

Situation-specific Development of Business Models within Software Ecosystems



Sebastian Gottschalk

Faculty of Computer Science, Electrical Engineering and Mathematics
Paderborn University

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doktor der Naturwissenschaften (Dr. rer. nat.)

December 2022

*Startups don't fail because they lack
a product; they fail because they
lack customers and a profitable
business model.*

- Steve Blank -

Acknowledgements

Although this thesis represents an independent work, its creation would be unthinkable without the support of numerous people. Therefore, at this point, I would like to thank all the people who, with their experience, time, or patience, have had a direct or indirect influence on the creation of this work.

First of all, I would like to thank Gregor Engels for being my doctoral supervisor. Gregor, thank you for providing me the freedom in following my research interest while guiding me throughout the whole journey. With that, you provided me with support in which I was able to grow not just professionally, but also personally. For that, I also want to thank my second examiner Dennis Kundisch. Dennis, thank you for the exciting interdisciplinary collaboration and for the valuable discussions over the last years. Furthermore, I would like to thank Roman Dumitrescu, Friedhelm Meyer auf der Heide, and Christoph Weskamp for being members of my examination committee.

However, research would remain impossible without the right environment. Here, I was glad to join the Software Innovation Campus Paderborn (SICP) during the start of my PhD. Therefore, I would like to thank especially Stefan Sauer, who showed me the PhD opportunity and brought me to the SICP when my previous startup journey failed. Moreover, I like to thank the organizing teams of the Collaborative Research Center (CRC) 901 of On-The-Fly Computing and the Software Campus (SC) initiative for their excellent programs in supporting PhD students following their research interests.

But the most important thing about the environment are the colleagues. Here, I would like to thank all my current and past colleagues from the SICP, CRC, SC, and our Database and Information Systems research group for the familiar atmosphere during the whole time. It was a pleasure to work as well as spend my free time with you. The same holds for the student assistants who support me in my research and the students whose theses I have supervised during that time. With your discussions and co-authorships, you all have also a share in this entire work.

Last but not least, I want to thank my family and friends. Here, my parents Dorle and Dieter together with my brother Fabian offered me lifelong support throughout the whole PhD way. Moreover, my friends supported me with the necessary breaks from research.

Abstract

The development of new business models is essential for startups to become successful, as well as for established companies to explore new business opportunities. However, developing such business models is a challenging activity. On the one hand, various tasks (e.g., conducting customer interviews) of business model development methods (BMDMs) need to be performed. On the other hand, different decisions (e.g., advertisements as a revenue stream) for the business models (BMs) need to be made. Both have to fit the changeable situation of the organization (e.g., availability of financial resources, mobile apps as application domain) in which the business model is developed to reduce the risk of developing ineffective business models with low market penetration. Therefore, the BMDMs and the BMs must be developed situation-specific. This situation-specific adaptation has already proven its value in Situational Method Engineering (SME), in which situation-specific software development methods are constructed from fragments of a method repository.

In this thesis, we conduct a design science research study to transfer the concept of SME to business model development. Our solution is a novel approach for the situation-specific development of business models with three stages. In the first stage, we create a method repository with method fragments for the BMDMs and a canvas model repository with modeling fragments for the BMs. Both repositories are filled by the knowledge of domain experts. Out of these repositories, in the second stage, situation-specific BMDMs for developing situation-specific BMs are composed by a method engineer based on the changeable situation of the organization and enacted by a business developer. The business developer collaborates with other stakeholders (e.g., software developer) during the enactment to create artifacts. Moreover, in the third stage, he receives IT support (e.g., design suggestions for the business model), provided by development support engineers, in different development steps. In particular, we support the crowd-based validation of prototypes for business models. In contrast to existing approaches, we point out the importance of the method engineer who composes a development method that fits the business developer's actual needs. We develop the whole approach as a modular concept and implement it as an extensible open-source tool. We apply it to the application area of software ecosystems, particularly mobile ecosystems. For these ecosystems, we provide an evaluation through a case study and a user study.

Zusammenfassung

Die Entwicklung neuer Geschäftsmodelle ist sowohl für den Aufbau erfolgreicher Startups als auch zur Erschließung neuer Geschäftsmöglichkeiten für bereits etablierte Unternehmen von entscheidender Bedeutung. Die Entwicklung solcher Geschäftsmodelle ist jedoch eine anspruchsvolle Aufgabe. Einerseits müssen verschiedene Aufgaben (z. B. Durchführung von Kundeninterviews) von Geschäftsmodellentwicklungsmethoden (GME Mn) durchgeführt werden. Andererseits müssen verschiedene Entscheidungen (z. B. Werbung als Einnahmequelle) für die Geschäftsmodelle (GMe) getroffen werden. Beide müssen an die veränderliche Situation der Organisation (z. B. Verfügbarkeit finanzieller Ressourcen, mobile Apps als Anwendungsbereich) angepasst werden, in der das Geschäftsmodell entwickelt wird, um das Risiko der Entwicklung ineffektiver Geschäftsmodelle mit geringer Marktdurchdringung zu verringern. Daher müssen die GME Mn und die GMe situationsspezifisch entwickelt werden. Diese situationsspezifische Anpassung hat sich bereits im Situational Method Engineering (SME) bewährt, bei dem aus Fragmenten eines Methodenrepositorys situationsspezifische Softwareentwicklungsmethoden konstruiert werden.

In dieser Arbeit nutzen wir Design Science Research, um das Konzept von SME auf die Geschäftsmodellentwicklung zu übertragen. Unsere Lösung ist ein neuartiger Ansatz für die situationsspezifische Entwicklung von Geschäftsmodellen mit drei Stufen. In der ersten Stufe erstellen wir ein Methodenrepository mit Methodenfragmenten für die GME Mn und ein Canvas-Modellrepository mit Modellierungsfragmenten für die GMe. Beide Repositories werden mit Wissen von Domänenexperten gefüllt. Aus diesen Repositories werden in der zweiten Phase situationsspezifische GME Mn für die Entwicklung situationsspezifischer GMe auf der Grundlage der veränderlichen Situation der Organisation von einem Methoden Engineer konstruiert und von einem Business Developer ausgeführt. Der Business Developer arbeitet bei der Ausführung mit anderen Stakeholdern (z. B. Softwareentwickler) zusammen, um Artefakte zu erstellen. Darüber hinaus erhält er in der dritten Stufe IT-Unterstützung (z. B. Designvorschläge für das Geschäftsmodell), die von Entwicklungsunterstützern für verschiedene Entwicklungsschritte bereitgestellt werden. Wir unterstützen insbesondere die crowd-basierte Validierung von Prototypen für Geschäftsmodelle. Im Gegensatz zu bestehenden Ansätzen betonen wir die Bedeutung des Methoden Engineers, der

eine Entwicklungsmethode zusammenstellt, die den tatsächlichen Bedürfnissen des Business Developers entspricht. Wir entwickeln den gesamten Ansatz als modulares Konzept und implementieren ihn als erweiterbares Open-Source-Tool. Wir wenden diesen auf den Bereich der Softwareökosysteme und speziell der mobilen Ökosysteme an. Für diese Ökosysteme liefern wir eine Evaluation auf Basis einer Fall- und einer Nutzerstudie.

Table of Contents

List of Figures	xv
List of Tables	xix
I Introduction, Foundations and Related Work	1
1 Introduction	3
1.1 Motivation and Problem Statement	3
1.2 Research Question and Research Approach	9
1.3 High-level Requirements and Solution Overview	14
1.4 Publication Overview	19
1.5 Thesis Structure	23
2 Foundations	25
2.1 Model Engineering	25
2.1.1 Metamodeling	26
2.1.2 Domain-specific Languages	27
2.1.3 Computer-Aided Modeling Tools	29
2.2 Situational Method Engineering	30
2.2.1 Concept of Method Engineering	30
2.2.2 Types of Situational Method Engineering Approaches	32
2.2.3 Computer-Aided Method Engineering Tools	35
2.3 Business Model Development	36
2.3.1 Business Model Modeling Languages	36
2.3.2 Business Model Development Methods	39
2.3.3 Business Model Development Tools	41
2.4 Summary	42

3	Related Work	43
3.1	Business Aspects in Situational Method Engineering	43
3.1.1	Related Approaches	43
3.1.2	Requirement Comparison	46
3.2	Situational Aspects in Business Model Development	49
3.2.1	Related Approaches	49
3.2.2	Requirement Comparison	52
3.3	Tools for Business Model Development	54
3.3.1	Related Approaches	54
3.3.2	Requirement Comparison	57
3.4	Summary	59
II	Solution Concept	61
4	Conceptual Overview	63
4.1	Overview of the Solution	63
4.1.1	Overview of Requirements	63
4.1.2	Overview of Stages	67
4.1.3	Overview of Roles	71
4.2	Application to Software Ecosystems	73
4.2.1	Introduction of Software Ecosystems	73
4.2.2	Usage of Solution	75
4.3	Summary	78
5	Knowledge Provision of Methods and Models	79
5.1	Requirements and Overview	79
5.2	Provision of Method Repository	81
5.2.1	Modeling of Method Elements	83
5.2.2	Modeling of Method Building Blocks	85
5.2.3	Modeling of Method Patterns	87
5.3	Provision of Canvas Model Repository	89
5.3.1	Modeling of Canvas Elements	90
5.3.2	Modeling of Canvas Building Blocks	91
5.3.3	Modeling of Canvas Models	93
5.4	Summary	95

6	Composition and Enactment of Development Methods	97
6.1	Requirements and Overview	97
6.2	Composition of Development Methods	100
6.2.1	Definition of Context	101
6.2.2	Situation-specific Composition of Methods	102
6.2.3	Domain-specific Composition of Models	105
6.3	Enactment of Development Methods	108
6.3.1	Execution of Development Process	109
6.3.2	Stakeholder Involvement in Artifact Development	111
6.3.3	Change of Context	112
6.4	Summary	114
7	Support of Development Steps	117
7.1	Requirements and Overview	117
7.2	Modularization of Development Support	119
7.2.1	Provision of Development Support	121
7.2.2	Composition of Development Support	126
7.2.3	Enactment of Development Support	130
7.3	Types of Support Modules	133
7.3.1	Integrated Canvas Module	134
7.3.2	Internal Hypothesis Modeling and Mapping Module	138
7.3.3	External Crowd-based Prototype Validation Platform	142
7.4	Summary	148
III	Implementation, Evaluation, Conclusion	149
8	Implementation	151
8.1	Modularized Architecture	151
8.1.1	Designed Architecture	151
8.1.2	Applied Technologies	154
8.2	Software Tool	163
8.2.1	Implemented Tool	163
8.2.2	Developed Modules	166
8.3	Summary	170

9	Evaluation	171
9.1	Case Study on OWL Live	171
9.1.1	Experimental Design	171
9.1.2	Execution	174
9.1.3	Analysis	181
9.1.4	Interpretation	183
9.2	User Study in Student Courses	184
9.2.1	Experimental Design	185
9.2.2	Execution	187
9.2.3	Analysis	196
9.2.4	Interpretation	198
9.3	Summary	201
10	Conclusion and Future Work	203
10.1	Contribution Summary	203
10.2	High-level Requirements Revisited	205
10.3	Future Work	208
	References	211
	Abbreviations	229
	Appendix A SLR on Business Model Decision Support Systems	233
	Appendix B Installation of the Situational Business Model Developer	241

List of Figures

1.1	Interplay of BMDMs, BMs, and BMDTs (based on [SL20])	4
1.2	Classification of <i>Business Model Development Methods</i> (based on [HBO94])	5
1.3	Classification of <i>Business Model Development Tools</i>	6
1.4	Development of a New Service for a Software Ecosystem	8
1.5	Three Cycles of Design Science Research (based on [Hev07])	10
1.6	Positioning of Research Contributions (based on [GH13])	12
1.7	Design Science Research Process (cycle from [KV08])	12
1.8	Solution Sketch of the Approach proposed in the Thesis	18
1.9	Overview of the Underlying Publications of the Thesis	20
1.10	Structure of the Thesis	23
2.1	Overview of the Meta-Object Facility (based on [BCW17])	27
2.2	Concept of Method Engineering (based on [MCF ⁺ 95])	32
2.3	Different Types of SME Approaches (based on [FB16])	33
2.4	Stages of Assembly-based SME (based [HSRÅR14])	34
2.5	Overview of the Business Model Canvas (structure from [OP10])	37
2.6	Overview of the Business Model Ontology (based on [Ost04])	38
2.7	Overview of a Development Method and a Tool (images from [GSE17, GFC14])	39
4.1	Mapping of High-Level Requirements to Solution Requirements	64
4.2	Overview of the Approach with Roles and Stages	69
4.3	Exemplary Execution of the Development Process	70
4.4	Overview of the Mobile Ecosystem with the Mobile ToDo Application . . .	75
4.5	Exemplary Stages for the Mobile App Developer	76
5.1	Overview of the Knowledge Provision of Methods and Models	82
5.2	Abstracted Phase of Providing the Method Repository	83
5.3	Metamodel of the Method Elements	84

5.4	Metamodel of the Method Building Blocks	86
5.5	Metamodel of the Method Patterns	87
5.6	Exemplary Visualisation of the Method Patterns	89
5.7	Abstracted Phase of Providing the Canvas Model Repository	90
5.8	Metamodel of the Canvas Elements	90
5.9	Metamodel of the Canvas Building Blocks	92
5.10	Exemplary Visualisation of the Canvas Building Block	93
5.11	Metamodel of the Canvas Models	94
5.12	Exemplary Model of the Business Model Canvas	95
6.1	Overview of the Composition and Enactment of Development Methods . . .	99
6.2	Abstracted Phase of Composition of Development Methods	100
6.3	Exemplary Definition of the Context	102
6.4	Exemplary Pattern-based Construction of Methods	104
6.5	Exemplary Consolidation of Models	107
6.6	Abstracted Phase of Enactment of Development Methods	108
6.7	Exemplary Transition from a Method Building Block to a Development Step	110
6.8	Exemplary Visualisation of Canvas Artifact	111
6.9	Exemplary Change of Context for a Pattern-based Constructed Method . . .	114
7.1	Overview of the Support of Development Steps	119
7.2	Abstracted Phase of Modularization of Development Support	120
7.3	Overview of the Support Modules	122
7.4	Extension of the Method Elements for the Modularization	123
7.5	Exemplary Method Elements for the Development Support	124
7.6	Extension of the Method Building Blocks for the Modularization	127
7.7	Exemplary Method Building Block for the Modularization	129
7.8	Extension of the Process Engine for the Modularization	131
7.9	Exemplary Usage of the Execution Manager for the Modularization	132
7.10	Exemplary Modules for Development Support	133
7.11	Exemplary Visual Notation for the Canvas Artifact	136
7.12	Exemplary Visual Notation for the HypoMoMap Artifact	140
7.13	DSR Process for the CPBV Platform	143
7.14	Solution Design for the CPBV Platform	147
8.1	Overview of the Modularized Architecture	152
8.2	Component Diagram of the Modularized Architecture	153
8.3	Main Parts of an Angular Application	155

8.4	Screenshots for Provision of Method Repository	164
8.5	Screenshots for Provision of Canvas Model Repository	165
8.6	Screenshots for Composition of Development Methods	165
8.7	Screenshots for Enactment of Development Methods	166
8.8	Screenshots for Modularization of Development Support	167
8.9	Screenshots of the Canvas Module	168
8.10	Screenshots of the Hypotheses Modelling and Mapping Module	169
8.11	Screenshots of the Crowd-based Prototype Validation Platform	170
9.1	Knowledge Provision: Examples of the Method Repository based on the GLR	176
9.2	Method Composition: BMDM for OWL Live based on the identified Context	178
9.3	Method Enactment: Example of the BMC for OWL Live	181
9.4	Knowledge Provision: Examples of the Canvas Model Repository based on the TD	190
9.5	Method Composition: Parts of the BMDMs for the Mobile Apps	192
9.6	Method Enactment: Parts of the BMC for the Mobile Apps	194
9.7	Development Support: Self-evaluation of the Platform	195
9.8	Evaluation of the Design Principles (DPs) on the CBPV platform	197
A.1	Extracted Number of Publications per Stage of the SLR	235
B.1	Tool Overview for the Method Engineer	241

List of Tables

3.1	Comparison of the Situational Approaches (SA) against the High-level Re- quirements (HR)	47
3.2	Comparison of the Business Approaches (BA) against the High-level Re- quirements (HR)	53
3.3	Comparison of the Business Tools (BT) against the High-level Requirements (HR)	58

Part I

Introduction, Foundations and Related Work

Chapter 1

Introduction

In this thesis, we show our approach for situation-specific business model development within software ecosystems. At the beginning of the thesis, we motivate our thesis topic and explain the underlying problem statement (1.1). For that, we present our research approach based on our stated research question (1.2). Next, we give an overview of the solution based on identified high-level requirements (1.3) and provide an overview of the publications on which our thesis is based (1.4). Finally, we present the structure of the thesis (1.5).

1.1 Motivation and Problem Statement

The development of effective business models, defined by Osterwalder et al. as “the rationale of how the organization creates, delivers, and captures value” [OP10], is an essential task for an organization to remain competitive and sustainably successful. Here, a study by CB Insights [CBI19] in 2019 analyzed 101 bankrupt startups and concluded that 42% of them failed due to a missing market need. But also for established companies, the GE Innovation Barometer [Gen18] in 2018 stated that 64% of the over 2000 business executives have the problem of developing effective business models for new ideas. By comparing the results with a previous study in 2015, the challenge is getting even more prominent (59% of over 3000 executives). An important reason for this is that customers want solutions for perceived needs rather than just products [Tee10]. This corresponds to the potential effect that the business model can often be more important than the latest technology of the product [Che10]. Here, especially Software Ecosystems (SEs), that can be defined as “a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs” [BBS10] tend to standardize more and more technological parts within Software Development Kits (SDKs) on their software platforms and make them available

for all developers. Therefore, external developers are highly dependent on developing and modifying their business models to stay competitive with their services against other developers. In particular, for mobile ecosystems as a subset of SEs, the AppAnnie’s State in Mobile 2021 study [App21] highlighted that these ecosystems provided 218 billion app downloads that led to 142 billion dollar revenue just in 2020. Nevertheless, app developers compete with their apps against millions of other apps over the users’ usage time.

The development of business models is a complex and creative activity that consists of different phases (e.g., discover, develop) where multiple tasks (e.g., conduct customer interviews, analyze competitors) of *Business Model Development Methods (BMDMs)* need to be conducted [GSE17]. Inside those tasks, communication and collaboration between different stakeholders (e.g., business developer, customer) is crucial or even required [EHB11, EBL16], different alternative decisions for the *Business Models (BMs)* have to be developed (e.g., subscription or advertisement as a revenue stream) [SEP⁺19, ADv13], and underlying hypotheses need to be validated (e.g., sales channel, product price) [McG10, BO20]. As shown in Figure 1.1, for *Business Model Development (BMD)*, the BMDMs and the BMs are both essential and highly interrelated. Moreover, they can be assisted using software-based *Business Model Development Tools (BMDTs)*.

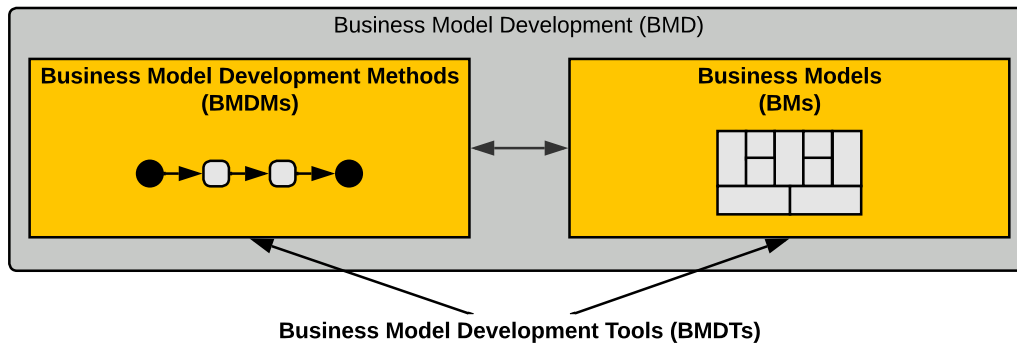


Fig. 1.1 Interplay of BMDMs, BMs, and BMDTs (based on [SL20])

To support the **BMD**, organizations often rely on light visualization tools like Kanban boards [GM03] for structuring the development steps of the BMDT or modeling artifacts like canvas models [OP10] for structuring the information of the BM. Due to the complexity of the BMD, there are several options for each development step. Consequently, a development method with inappropriate steps can lower the quality of the business model. Here, the guidance of domain experts supports the development by providing all stakeholders with the needed understanding of the development steps in the development method and presented information in the modeling artifacts [SL20]. In literature, different domain experts propose various methods to develop such business models in the form of development methods (e.g.,

[McG10, SEP⁺19]) and method repositories (e.g., [BO20, SP09]). Moreover, these experts provide knowledge in the form of taxonomies with decisions of possible (e.g., [HZFN16, AIB07]) and patterns of successful (e.g., [RHTK17, GFC14]) business models. However, the BMDMs should match the organization's current situation (e.g., financial resources, target market size), and the information within the BMs needs to match the application domain (e.g., mobile app, social network) of the product/service of the organization [STRV10, FWCG13]. This, in turn, raises the chance of developing an effective business model for the organization. For that, organizations (e.g., external developers of services) are supported with BMDMs and corresponding BMDTs to develop effective BMs.

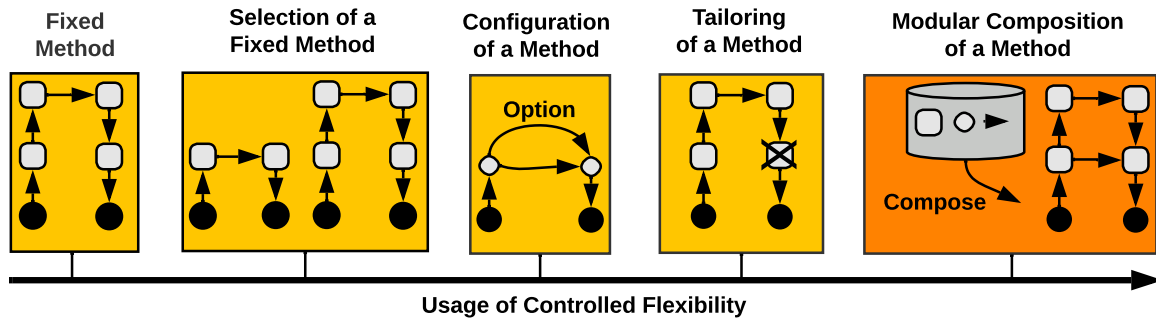


Fig. 1.2 Classification of *Business Model Development Methods* (based on [HBO94])

Although various **BMDMs** for developing **BM**s have been proposed, they do not cover the composition and modification of the method and the modeling artifacts to the existing and changeable context based on refactored expert knowledge. As shown in Figure 1.2, those approaches can be classified through their *Usage of Controlled Flexibility* [HBO94]. Thereby, controlled flexibility is defined as the combination of the free arrangement of the development steps (i.e., flexibility) and the restricted arrangement of the development method (i.e., control). Here, most approaches (e.g., [McG10]) provide *Fixed Methods* that do not focus on the organization's context and apply the same tasks to every situation. Out of the approaches, also a *Selection of Fixed Methods* (e.g., [McG10] or [Rie14]) can be made. Some approaches try to cover this lack of controlled flexibility by providing different *Configurations of a Method* (e.g., [SEP⁺19]) or basic *Tailoring of a Method* (e.g., [BO20]). Nevertheless, these one-size-fits-all methods cannot fit every existing and newly detected context in advance (e.g., availability of a low or high number of competitors). This context, in turn, can be considered with the *Modular Composition of a Method* out of refactored expert knowledge and changed during the development. Therefore, we consider the context-specific composition of the development method and the modeling artifacts based on the knowledge of domain experts by refining the concept of Situational Method Engineering (SME) [HSRÅR14] for BMDMs.

Originally, SME is used for the situation-specific composition and enactment of software development methods to structure the development of software projects.

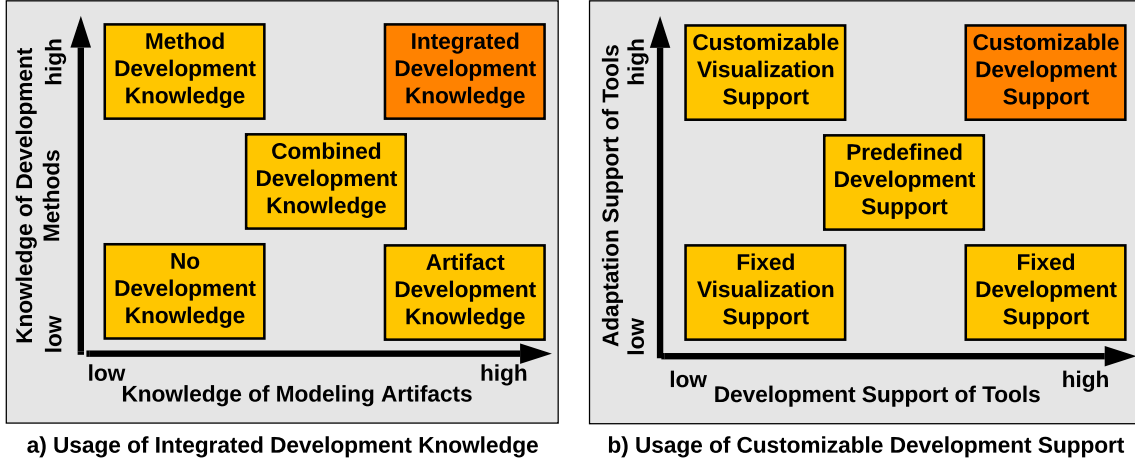


Fig. 1.3 Classification of *Business Model Development Tools*

To assist the development, **BMDTs** are developed in research and practice and provide different types of support like guidance in the development of the actual model or collaboration among the stakeholders [VCB⁺14]. While the tools in practice are often focused on the pure visualization of the business models [SSJ⁺19], tools in research are able to use existing knowledge and provide design and decision support for specific tasks [BdRHF20]. As shown in Figure 1.3, we classify selected approaches through their usage of integrated development knowledge and customizable development support.

As in Figure 1.3 a) *Usage of Integrated Development Knowledge*, we refer to the utilization of existing knowledge for the development method (e.g., knowledge about different development steps within a development method) and the modeling artifacts (e.g., knowledge about different decisions of the business model) to support the BMD. Here, most tools focus on the pure visualization of business models with *No Development Knowledge* about the methods and within the modeling artifacts (e.g., [FP10] with the visual structure of a business model). Some tools use *Method Development Knowledge* to structure the different phases and development steps with predefined knowledge (e.g., [BO20] with a fixed repository of different experiments for the validation phase). Other tools use *Artifact Development Knowledge* to guide the possible decisions within the modeling artifact with predefined knowledge (e.g., [LFBBM19] with a repository of different patterns for possible business models). Moreover, a few tools use the *Combined Development Knowledge* of fixed methods and model artifacts to support both the structuring of the development method and the modeling of the artifacts (e.g., [BM17] with the fixed phases for the development

method and domain-specific knowledge for the models). However, none of the identified tools uses the *Integrated Development Knowledge* of various freely-definable methods and models from different domain experts that are utilized in a reusable manner. Based on those reusable knowledge sources, the development can be tailored to the organization's situation and the application domain of the product/service with low additional effort in contrast to one-size-fits-all methods. For that, both the *Method Development Knowledge* and *Artifact Development Knowledge* need to be utilized to use them during the BMD. Therefore, we use the concept of Model Engineering (ME) [Béz06] within BMDTs. Here, ME is used to abstract specific information from the real world into models, making them reusable.

As in Figure 1.3 b) *Usage of Customizable Development Support*, we refer to the usage of software (e.g., specific software for visualizing the business model) to provide flexible support for different development steps (e.g., design support for developing the business model based on recommendations) during the development. Here, most tools have a *Fixed Visualization Support* that does not offer real development support but focuses on the visualization of those business models (e.g., [FP10] with the visual structure of a business model). Some tools provide *Customizable Visualization Support* for the business model (e.g., [FAM18] with different visual representations of the organization's data). Other tools use *Fixed Development Support* on specific development steps (e.g., [DLEL19] with the validation of the business model with the crowd). Moreover, a few tools provide *Predefined Development Support* to support a fixed set of development steps (e.g., [BM17] with predefined support for the configuration and the analysis of business models). However, none of the identified approaches uses *Customizable Development Support* for supporting different development steps with adjustable software parts. With this, the development can be supported flexibly on those needed steps. Therefore, we apply the concept of Modularization [Par72] to BMDTs. Here, Modularization is used to bundle different functionalities for a specific use case within a single dedicated piece of software.

Based on those challenges of current BMDMs and BMDTs, we provide a modular concept for the situation-specific BMD. This approach uses the knowledge of methods and modeling artifacts to compose a development method for a defined context. This development method is enacted, and development support in the form of adjustable software parts is used for specific development steps. During the enactment, the development method can be modified according to the changed context. We show the application of our approach based on developing business models for service providers in SEs. Our applied approach can be seen in Figure 1.4. Here, we have a *Software Platform* that a *Platform Provider* provides. Moreover, we have *Service Providers* who have developed some *Services* that the *End Users* execute. If a *New Service Provider* develops a *New Service* for the ecosystem, he also needs

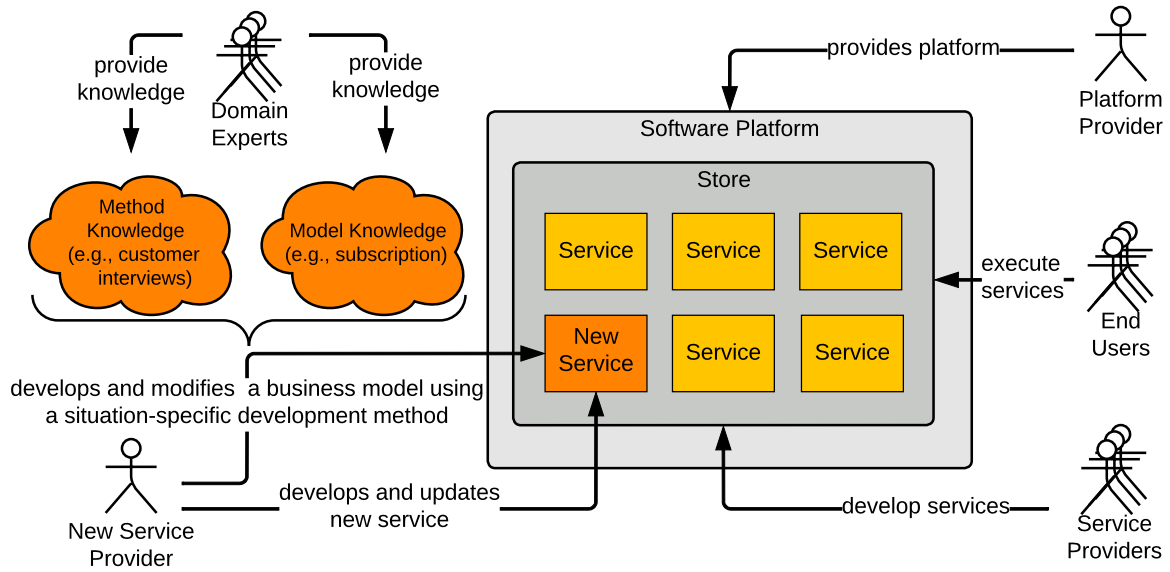


Fig. 1.4 Development of a New Service for a Software Ecosystem

to find a suitable business model. For that, in turn, he can use the existing *Method Knowledge* (e.g., *customer interviews as a development step*) for the development method and the *Model Knowledge* (e.g., *subscription as a revenue stream*) within the modeling artifacts from the *Domain Experts*. Out of that knowledge, the *New Service Provider* develops and modifies a business model using the development method. In particular, we focus on mobile ecosystems as one of the most established but also competitive markets within SEs.

Inside this thesis, we are using the running example of a mobile app developer who develops a mobile to-do app. Here, the app could provide various features like managing to-dos in general, advanced collaboration features, and the integration of machine learning techniques. The developed business model is based on the context in terms of the situation of the mobile app developer (e.g., availability of low or high financial resources) or the application domain (e.g., an app for productivity or collaboration). Out of this context, a development method is composed and enacted. During the composition, different tasks are assigned as development steps (e.g., choose problem analysis and target market analysis as development steps), and in the enactment, those tasks are conducted to develop a business model (e.g., choose subscription or advertisement as a revenue stream). Moreover, during the enactment, the development method might be modified due to a changing context which could lead to a change in the business model. Last, the conduction of specific development steps is supported by different software pieces (e.g., calculation of business outcome). Based on those identified challenges, we state the research question of our thesis to provide a situation-specific BMD.

1.2 Research Question and Research Approach

By analyzing the existing BMDMs and BMDTs, we identified three challenges to support the situation-specific development of BMs. These were the *Usage of Integrated Development Knowledge* with the lack of *Integrated Development Knowledge*, the *Usage of Controlled Flexibility* with the lack of *Modular Composition of a Method*, and the *Usage of Customizable Development Support* with the lack of *Customizable Development Support*. Out of those challenges, we derive our overall research question (RQ) for the situation-specific BMD as presented in this thesis:

- **Overall RQ:** How to enable the situation-specific development of business models for service providers within software ecosystems?

Based on the three challenges, we also divide our overall research question into three subquestions (RQ1, RQ2, RQ3) that will be answered within the thesis. The first one goes along with the *Usage of Integrated Development Knowledge*. Here, we want to use the knowledge from development methods and canvas modeling artifacts to guide the BMD. For that, we need to formalize and store the knowledge of different domain experts in a uniform and structured way. Therefore, the first question is:

- **RQ 1:** How to utilize the knowledge about development methods and canvas modeling artifacts to support the business model development?

The second one is related to the *Usage of Controlled Flexibility*. Here, we want to use the provided knowledge of the development methods and the modeling artifacts to allow the situation-specific composition of a BMDM. This composition, in turn, goes hand in hand with their enactment to support the stepwise development of a BM. Therefore, the second question is:

- **RQ 2:** How to handle the situation-specific composition and enactment of business model development methods by using utilized knowledge for the development methods and the canvas modeling artifacts?

The third focuses on the *Usage of Modular Development Support*. Here, the enacted BMDM contains various development steps. Those development steps can be conducted with the assistance of different software support. In order to provide those support at the right time, the support needs to be flexibly adjusted to those steps. Therefore, the third question is:

- **RQ 3:** How to support the development of artifacts within the business model development steps using flexibly customizable software support?

To answer those three questions and the overall RQ, we conducted a design science research (DSR) study [HMPR04] to create a situation-specific development approach for business models. We used DSR as it aims to solve a class of problems by developing a solution to a specific problem and then generalize that gained knowledge [GH13]. This, in turn, is based on developing and evaluating a corresponding IT artifact [HMPR04]. Here, we solved the problem of situation-specific development in the mobile app application area and ensured that the knowledge is generalizable to other application areas.

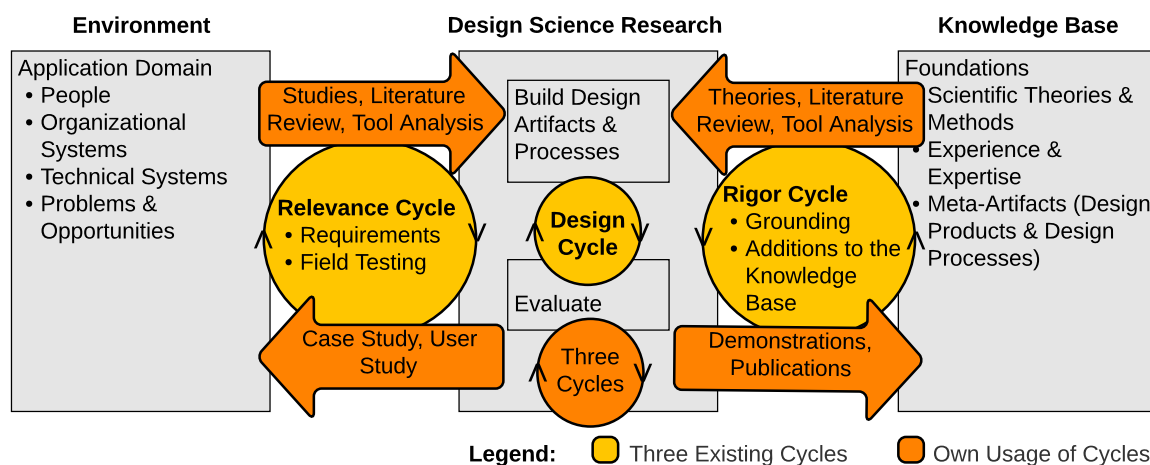


Fig. 1.5 Three Cycles of Design Science Research (based on [Hev07])

Typically *Design Science Research*, as shown in Figure 1.5, is originated between the *Environment* where it will be applied and the *Knowledge Base* where it is founded [Hev07]. Here, the *Environment* consists of the organization in terms of involved people, used (technical) systems, and corresponding problems and opportunities. Those real-world problems are usually brought as *Requirements* into *DSR* using the *Relevance Cycle* and need to be constantly evaluated with *Field Testing*. Moreover, the *Knowledge Base* consists of existing scientific theories and methods, formalized experience and expertise, and already designed meta-artifacts. Those foundations are used as *Grounding* of the *DSR* using the *Rigor Cycle*, and the obtained results need to be pushed back in *Addition to the Knowledge Base*. Both cycles are combined using the *Design Cycle* in *Design Science Research*, where artifacts are constantly designed and evaluated [HMPR04].

For this thesis, we initiated the *Relevance Cycle* with recent studies about the problems of BMD in companies using the GE Innovation Barometer 2018 [Gen18] and startups using the CB Insights 2019 report [CBI19]. We addressed the application domain of mobile apps because of their attractiveness but also competitiveness [App21]. Moreover, we initiated the *Rigor Cycle* based on the kernel theories of opportunity creation [ABA13] and boundary

objects [SG89]. The opportunity creation theory states that businesses are co-created under high uncertainty. Here, the development is an entrepreneurial process where companies create a business model based on their assumptions that need to be validated with the customers [Vog17]. Therefore, the process needs both parts of exploitation and exploration. The bounded object theory states that development is a complex activity where different understandings can occur. Here, the development is a heterogeneous task that requires the collaboration of different stakeholders with different knowledge [SL20]. Therefore, a common understanding between all stakeholders needs to be achieved. Moreover, we based both the *Requirements* of the *Relevance Cycle* and the *Grounding* of the *Rigor Cycle* on an exhaustive review of literature on BMD and a tool analysis of existing Business Model Decision Support Systems (BMDSSs). To connect the *Design Science Research* with the *Environment* and the *Knowledge Base*, we made use of the *Field Testing* of the *Relevance Cycle* and the *Additions to the Knowledge Base* for the *Knowledge Base*. For the *Field Testing*, we conducted a case study and a user study on developing business models for mobile apps. For the *Additions to the Knowledge Base*, we demonstrated our developed artifacts and published parts of our research. In the *Design Cycle* of *Design Science Research*, we conducted three cycles based on the cycle of Kuechler and Vaishnavi [KV08]. The cycle, see Figure 1.7, consists of the following five iteratively conducted steps. First, the (1) *Awareness of [The] Problem* is identified based on a real-world problem and a (2) *Suggestion* of a possible solution is provided. Next, the (3) *Development* of the artifact is done and an (4) *Evaluation* is conducted. Based on the evaluation results, a further iteration is conducted or the research contribution as a (5) *Conclusion* is provided.

An important aspect of DSR is the positioning of the research contributions. According to Gregor and Hevner [GH13], the positioning can be divided into different *Contribution Types* and be situated to the *Knowledge Contribution Framework* as shown in Figure 1.6. The *Contribution Types* are divided into their *Knowledge* attractiveness and completeness to reuse the gained knowledge in other application domains. Here, *Level 1* provides a situated implementation of the artifact (e.g., software tool), *Level 2* provides the knowledge as operational principles (e.g., method), and *Level 3* provides a well-developed design theory (e.g., mid-range theory). Moreover, the *Knowledge Contribution Framework* divides between a high or low *Solution Maturity* and a high or low *Application Domain Maturity*. Here, the DSR can be defined as *Invention*, where a new solution is developed for a new problem, *Improvement*, where a new solution solves a known problem better, *Exaptation*, where a known solution is extended to a new problem, and *Routine Design*, where a known solution is used for a known problem.

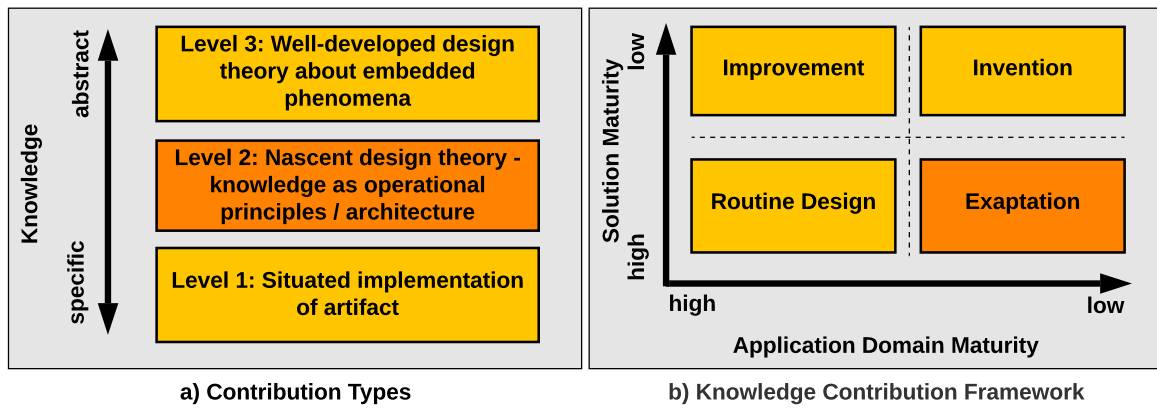


Fig. 1.6 Positioning of Research Contributions (based on [GH13])

In this thesis, we will develop different *Contribution Types*. Here, we will contribute our evaluated modularized concept (*Level 2*) and our implemented open-source software tool (*Level 1*). Moreover, we will create an *Exaptation* according to the *Knowledge Contribution Framework*. Here, we will refine parts of the concepts of Software Product Lines (SPLs) [ABKS13], SME [HSRÅR14], and modularization [Par72] to the application domain of BMD. Based on our overall RQ, the three cycles of DSR, and the positioning of the research contribution, we conducted three design cycles, as shown in Figure 1.7. In this thesis, we will present the results of the five steps of our third cycle.

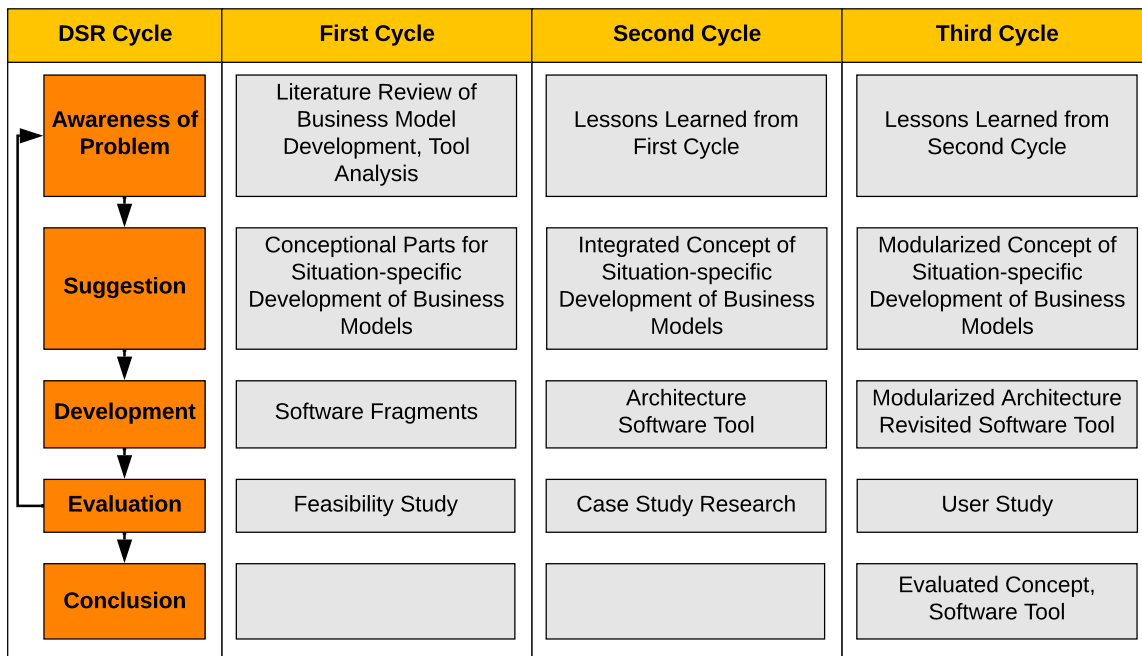


Fig. 1.7 Design Science Research Process (cycle from [KV08])

In the **First Cycle**, we got *Aware of [the] Problem* based on the studies on BMD [Gen18, CBI19] and the theories on opportunity creation [ABA13] and boundary objects [SG89]. That information was used for an exhaustive literature review on BMD and a tool analysis on BMDSSs. The tools are collected by a Systematic Literature Review (SLR) [Kit04]. Based on those, we derived high-level requirements (HRs) of our solution with a grounding in literature. In *Suggestion*, we provided conceptual parts for the situation-specific development of business models. Here, we focused on RQ 1 of formalizing and storing knowledge by applying the concept of SPLs [ABKS13] to business models. We modeled all possible business models as a feature model [CN09] based on the canvas model structure of the Business Model Canvas (BMC) [OP10]. Moreover, we developed a method to fill the feature model initially and change the selected features based on the conduction of experiments of a method repository. For the *Development*, we designed web-based tooling and implemented the conceptual parts as software fragments. As *Evaluation*, we conducted a feasibility study to show the applicability of an illustrative scenario. Here, we developed a potential business model for a mobile to-do app.

In the **Second Cycle**, we updated our *Awareness of [the] Problem* by taking the lessons learned from the last cycle. During the evaluation of the first cycle, we saw especially limitations in the method by considering an initial development of all possible business models and their following validation through experiments. Moreover, we saw the limitation of using just the BMC for structuring the knowledge. Based on those findings, we updated our HRs for the solution. In *Suggestion*, we provided an integrated concept for the situation-specific development of business models. Here, we focused on solving RQ 2 of composing and enacting methods and improved RQ 1 by applying the concept of SME [HSRÅR14]. We stored knowledge about possible method parts in a method repository and possible model parts in a model repository. Out of those, we provided a situation-specific composition of the method. Moreover, different stakeholders might collaborate to develop the business model during the enactment. For the *Development*, we designed an architecture and implemented the whole concept as a software tool. As *Evaluation*, we conducted a case study on developing potential business models for a local event platform.

In the **Third Cycle**, which results will be presented in this thesis, we updated our *Awareness of [the] Problem* by taking the lessons learned from the last cycle. During the evaluation of the second cycle, we saw limitations in the proposed tools of the method repository that can not directly interact with the models in the model repository. Moreover, current BMDSSs are not based on each other but are built from scratch. Based on those findings, we updated our HRs for the solution. In *Suggestion*, we provided the modularized concept of the situation-specific development of business models. Here, we focused on

solving RQ 3 of flexible development support and improved RQ 1 & 2 by applying the concept of modularization [Par72]. Here, we bundle that development support in the form of parts in software modules. For that, we provide different exemplary modules. In particular, we develop development support for crowd-based validation of prototypes for business models. For that, we used an additional DSR study that will be presented in the corresponding Section 7.3.3. For the *Development*, we designed a modularized architecture to allow the usage of such modules and revisited our software tool. As *Evaluation*, we conducted a user study in student courses on mobile and augmented reality (AR) / virtual reality (VR) applications. Finally, we draw a *Conclusion* with the evaluated concept and the software tool. With this conclusion, we provide the *Contribution Type* of operational principles of the concept (*Level 2*) and the situated implementation of the software tool (*Level 1*). Moreover, we provide the adaptation of existing concepts to the BMD according to the *Knowledge Contribution Framework* (*Exaptation*). In the following, we present the HRs out of that third cycle.

1.3 High-level Requirements and Solution Overview

To get aware of the problem during our DSR study, we reviewed BMD literature and analyzed BMDSSs. The SLR that built the foundation of our tool analysis is provided in Appendix A. Out of that, we derived the high-level requirements (HRs) behind our solution to answer our overall RQ, including the subquestions of RQ 1, RQ 2, and RQ3. Here, we based the choices of our requirements on the theories of opportunity creation [ABA13] and boundary objects [SG89]. Those HRs were updated through the design cycles, and in the following paragraphs, we provide the requirements of the third design cycle.

The development of business models is a complex and challenging task. This task can be supported with knowledge about tasks to be accomplished (i.e., methods) and decisions to be made (i.e., models) [SL20], where the development process can be as important as the model itself [JKS17]. In order to utilize that knowledge, well-defined syntax and semantics for the development methods and modeling artifacts are needed [VCB⁺14] that can also represent different levels of abstractions and content [MTA17]. Existing tools provide that knowledge based on metamodels and, partly, shared vocabulary [BM17]. Based on those metamodels, relevant data can be gathered from different repositories and consolidated for the development [Sch14]. Those tools use the knowledge for the visualization [FAM18] or transformation [AFM18] of business models. Therefore, an utilization of the knowledge for the development methods and modeling artifacts is needed.

- **HR 1: Knowledge Utilization:** The solution should allow the utilization of knowledge about business model development methods and canvas models.

The first part of BMD contains the tasks for the BMDM that need to be performed. Here, BMD is a complex and creative activity consisting of different phases where multiple tasks must be accomplished [GSE17]. Some development methods stay on a high-level description of the process consisting of iterations of exploration and exploitation [McG10]. Other methods provide deeper insights into the distinguishment of different phases [FWCG13] or the conduction of different experiments [BO20]. Existing tools are developed for different phases of BMD [BdRHF20]. Here, some approaches provide overall guidance through the phases [dRAH⁺16], deeper integration of specific phases like configuration or analysis [Sch14], or overall phase management on the analysis, design, implementation and integration [EBL16]. Therefore, comprehensive methods that can support all phases of the BMD with tasks to be made are needed.

- **HR 2: Method Comprehensiveness:** The solution should allow the comprehensive development of business model development methods for all phases.

The second part of BMD contains the decisions for the BM to be made. To reduce the complex phenomena of a business, conceptual models can be used [MTA17]. Those conceptual models are often based on visualizations where different representations for the business model itself [JKS17] and corresponding tasks during the development process [TA17] exist. However, the BMC is the most used visualization technique [OP10]. The BMC is also well-applied by software tools in practice [SSJ⁺19]. Existing tools provide a visualization of the changes in the business model [FP10], heatmaps based on influencing factors [BHH⁺18], or customization of the canvas components [BM17]. Therefore, visual representations of the business model with decisions to be made are needed.

- **HR 3: Model Visualisation:** The solution should allow visual representations of the business model.

Before beginning the BMD, it is essential to know the context in which the development takes place [STRV10]. Here, the context can consist of internal and external factors of the business [McG10]. Moreover, approaches can take prior experience in business modeling and industrial characteristics of the business into account [SSJ⁺19]. However, just a few studies investigate BMD concerning the specific context of the business [BSRP16]. Existing tools use the maturity of the user to support the development [FP16], provide different business model patterns based on the type of business [LFBBM19], or support ideation based on the context of the idea [Joh16]. Therefore, awareness of the context is essential during the development of the business model.

- **HR 4: Context Awareness:** The solution should be aware of the context in which the business model is developed.

After getting aware of the actual context, the business developer needs to find the best BMDM for the BM. For that, the business developer can select from a variety of different models and methods [SL20]. For example, this could be different experiments from a large repository [BO20] of experiments or patterns for a large database of patterns [RHTK17]. Existing tools solve those with different levels of guidance for different levels of maturity of the users [FP16]. Here, the method is guided by a wizard for the different phases [VOPN13], and the modeling is supported by a previous questionnaire [RRE⁺19]. Moreover, recommendations for different visualizations based on the development goal are provided [dRAH⁺16]. Therefore, an assistant for selecting the development method and canvas models for a specific context is important.

- **HR 5: Selection Assistance:** The solution should assist the development of business models with the selection of business model development methods and canvas models.

Based on the selected methods and models, the business developer starts to develop the business model. Here, such a development is a continuous activity [Tee10], where the business needs to react to changes in the business environment [OPT05]. This is because business models are developed under high uncertainty, where not everything can be anticipated in advance [McG10]. Therefore, experiments need to be continuously conducted [Che10]. Possible changes can be inside the target market, the value proposition, the value capture, or the value delivery [SLF17]. Existing tools provide continuity with a loop of experimentation [KB10] and the iterative validation of business model ideas [DLEL19]. Moreover, they provide iterative adjustments based on changing market conditions [RHRH⁺19]. Therefore, continuity in the changes of the BMDM and the BM during the development is needed.

- **HR 6: Development Continuity:** The solution should allow continuous changes in the business model and the business model development method during the whole development.

During the continuous development of the business model, different stakeholders are involved. Here, parts of the development consist of creative tasks where those different stakeholders need to collaborate [SL20]. For that, the stakeholders can interact during the different phases [AdR20], like in the phase of generating new ideas for a business model [EHB11]. In general, there is a division between the internal and external stakeholders of a company [SL20] and the synchronous and asynchronous collaboration [SSJ⁺19]. Existing tools provide basic functionalities for sharing business model ideas among different

stakeholders [VOPN13]. Moreover, they provide functionalities to discuss and rate business models, including their components [SBK18a]. Therefore, a collaboration of different stakeholders during the development is needed.

- **HR 7: Stakeholder Collaboration:** The solution should allow the collaboration of different stakeholders during the business model development.

During the development of the business model, different artifacts are created. Here, developing business models is a complex activity, where traceability of the artifacts can ensure the understanding of the reasons behind all changes over time [OPT05]. With that, it is possible to see the evolution of a business model [OPT05], analyze the reasons for failed experiments [McG10], and develop a change plan [BSRP16]. Existing tools visualize the evaluation of business models [FP14b], provide a tracing between current and target business models [AFM18], track the reasons for changes [SBK18a], and provide advanced reasoning to managers [HKB18]. Therefore, management, including the traceability and reasoning, of all (canvas) artifacts is needed.

- **HR 8: Artifact Management:** The solution should provide management of all (canvas) artifacts that are created during the business model development.

To support the BMD, including artifact management, different BMDTs have been introduced in research [BdRHF20] and practice [SSJ⁺19]. However, to efficiently support the development, those tools need to evolve into BMDSSs [OP13a]. Currently, there are different tools for different phases, different stakeholders, different units of analysis, and different values [BdRHF20]. Here, an ongoing question in research is the level of automation that can be used within the BMD [BdRHF20]. Existing tools provide different types of decision support like generation of ideas [Joh16], analysis of robustness [HBJdR17], simulation of alternatives [GFCL], and validation of ideas [DLEL19]. Our SLR of those tools can be seen in Appendix A. Therefore, support for decision-making during the development is needed.

- **HR 9: Decision Support:** The solution should provide support for decision-making during business model development.

Like for decision support, software tools can be used to reduce the complexity at all phases of BMD. Those BMDTs can be used to support both methods and models during the development [SL20]. In practice, various BMDTs have been introduced that mostly focus on filling out the BMC [SSJ⁺19]. Common tool functionalities are comparing different business models [OPT05] and sharing ideas between different stakeholders [BdRHF20]. Therefore, support in the form of a software tool is needed.

Comprehensiveness) and adequately visualize the used canvas models (i.e., *HR 3: Model Visualisation*).

Based on that utilized knowledge, the **(2) Composition and Enactment of Development Methods** composes and enacts a situation-specific development method. For that, the *Business Developer* informs the *Method Engineer* about the business context who uses that information to formalize the context (i.e., *HR 4: Context Awareness / 2.1*). Based on that formalized context and additional information, the *Method Engineer* composes the development method (i.e., *HR 5: Selection Assistance / 2.2*). This development method, in turn, is then enacted (2.3) by the *Business Developer*. During the enactment, the *Business Developer* conducts different development steps together with *Other Stakeholders* to create artifacts (2.4). Specific steps are assisted with provided development support. Moreover, depending on changes in the context, a modification of the development method might be needed (i.e., *HR 6: Development Continuity / change needed*), or the development could be finished (i.e., *development finished*).

To provide development support, the **(3) Support of Development Steps** assists the conduction of different development steps. For that, the *Meta-Development Support Engineer* needs to develop the structures for the development support in the software tool (3.1). Next, those structures are used by the *Development Support Engineer* to create different development support for specific development steps (3.2). During the conduction, different stakeholders should collaborate (i.e., *HR 7: Stakeholder Collaboration*) and the created artifacts should be managed (i.e., *HR 8: Artifact Management*). Moreover, the conduction might be assisted by decision support (i.e., *HR 9: Decision Support*). Finally, all stages are supported with a software tool (i.e., *HR 10: Tool Support*). Those stages are also proposed by different published publications.

1.4 Publication Overview

During the work on the thesis, we already published parts of our research, as shown in Figure 1.9. We divide those publications into the six categories of domain overview, solution overview, solution concept (first cycle), solution concept (second cycle), solution concept and modules (third cycle), associated publications, and follow-up works.

Inside the **Domain Overview**, we provide insights into the application domain in which the thesis takes place. Here, we analyze the *Business Models of Software Ecosystems* [GRE19a] by mapping the business model decisions of different store-oriented software ecosystems to the BMC. Out of these decisions, we create a variability model for the decisions of the store provider, the service providers, and the end-users. Moreover, we provide an

Domain Overview:	Business Models of Software Ecosystems [GRE19a]	Overview of Decision Support for Business Model Development [GY21]
Solution Overview:	Situation-specific Business Model Development for Service Providers [Got21b]	Decision-support Systems for Business Model Development [Got21a]
Solution Concept (First Cycle):	Intertwined Development of Business Models and Product Functions [GRE19b]	Business Model Development based on Product Line Engineering [GRE20]
Solution Concept (Second Cycle):	Situation-specific Business Model Development Methods [GYNE21b]	Domain-specific Business Knowledge Modeling [GKE21]
	Composition and Enactment of Business Model Development Methods [GYNE21a]	Tool-support of Situational Business Model Developer [GYNE22c]
	Continuous Situation-specific Business Model Development [GYNE22a]	
Solution Concept and Modules (Third Cycle):	Modularized Architecture for Business Model Development Tools [GYNE22b]	
	Hypothesis Valiation for Business Model and Product Features [GYE20]	Crowd-based Prototype Validation Platform for Services [GAYE21, GPYE22]
Associated Publications:	Model-based AR Product Configuration [GYSE20a, GYSE20b]	Detecting Process Incompatibilities Process-based DSS [KGE22]
Follow-Up Works:	Visualizing and Simulating Platform Ecosystems [VG21]	Business Model Ideation for Business Ecosystems [VGK+22, RBLL+22]
	Model-driven Continuous Experimentation [GYE22, GBW+22]	Situation-specific Design Thinking Processes [GYNE22d]

Fig. 1.9 Overview of the Underlying Publications of the Thesis

Overview of Decision Support for Business Model Development [GY21]. Here, we discuss the division between data-based and data-driven BMD and show how data-driven approaches use data, information, and knowledge from internal and external company sources during development.

Inside the **Solution Overview**, we show a high-level overview of our solution and the underlying research approach. Here, we attend a doctoral consortium on the topic of *Situation-specific Development for Service Providers* [Got21b]. This paper shows our DSR approach

and sketches a preliminary solution after the third cycle. Moreover, we attended a researcher career workshop with the research statement of *Decision Support Systems for Business Model Development* [Got21a]. This statement shows how we connect our two different DSR studies of the situation-specific BMD and the crowd-based prototype validation that are presented in Section 1.2 and Section 7.3.3.

In the **Solution Concept (First Cycle)**, we apply the concept of SPLs to BMD during the first design cycle. With the *Intertwined Development of Business Models and Product Functions* [GRE19b], we use so-called features models to structure the knowledge of business models and related product functions in a common structure. Out of those feature models, we extract both using an iterative and incremental selection approach. By using *Business Model Development based on Product Line Engineering* [GRE20], we provide a method to insert the features in the feature model based on an analysis of the business and select them based on an analysis of the customer. Moreover, the method adapts those selected features based on the conduction of experiments.

In the **Solution Concept (Second Cycle)**, we apply the concept of SME to BMD during the second design cycle. With the *Situation-specific Business Model Development Methods* [GYNE21b], we show how the method repository of SME can be translated to BMD. For this, we create a method repository for composing BMDMs for mobile app developers. With *Domain-specific Business Knowledge Modeling* [GKE21], we show how a canvas model repository for BMD based on feature models can be used. For this, we provide a method to consolidate the knowledge of different domain experts to support the design of BMs. Inside the *Composition and Enactment of Business Model Development Methods* [GYNE21a], we show the definition of a context in terms of situational factors and applications domain to compose a development method out of both repositories. Moreover, we use lightweight structuring techniques of Kanban boards and canvas models during enactment. The *Tool-Support of Situational Business Developer* [GYNE22c] is a presentation of a prototype of our corresponding software tool. For that, we present an architectural overview and the functionalities of knowledge provision, method composition, and method enactment. In *Continuous Situation-specific Business Model Development* [GYNE22a], we provide a journal article of our second design cycle consisting of the knowledge provision, the method composition, and the method enactment.

In the **Solution Concept and Modules (Third Cycle)**, the modularized architecture and development support modules of the third design cycle are shown. In *Modularized Architecture for Business Model development Tools* [GYNE22b], we present our modularized architecture to support different development steps of BMDTs on a software business and a free software conference to represent its extensible. The used support modules contain

different tools with atomic steps to conduct and meta artifacts with artifacts to create. Moreover, we worked on different development support modules in separate publications. Here, *Hypothesis Validation for Business Model and Product Functions* [GYE20] supports the iterative validation step for the hypotheses about the business models and the product functions. For that, we have developed a modeling language and a development approach that we translate into a support module of our approach. With the *Crowd-based Prototype Validation Platform for Services* [GAYE21, GPYE22], we have developed a platform to support the validation step using the feedback of crowd-workers. Here, we conducted three design cycles of DSR to derive the operational principles and situated implementation of such a platform. For that, we connect the platform to our approach as an external support.

The **Associated Publications** present work related to our approach but not directly integrated. In *Model-based AR Product Configuration* [GYSE20a, GYSE20b], we use the created and implemented concept of feature models to model AR products. Those AR products can be freely adjusted within the real environment using a mobile phone and might be used to show prototypes of physical products. In *Detecting Process Incompatibilities Process-based DSS* [KGE22], we provide an approach to detect incompatibilities during the composition and the enactment of the process. That approach provides a fine-granular definition of incompatibilities and might be used to detect incompatibility between different steps of our development support.

Last, the **Follow-Up Works** present work that is currently in an early development stage but can be included in our approach as presented in the future work in Section 10.3. Here, with *Visualizing and Simulating Platform Ecosystems* [VG21], we present the idea of modeling platform ecosystems based on intra- and inter-related dependencies of the platform actors. That work might be used in the future to simulate parts of the ecosystem's behavior. In *Business Model Ideation for Business Ecosystems* [VGK⁺22, RBL⁺22], we present a procedure for supporting the ideation of possible business ecosystems based on canvas models. That work might be used in the future to consider the BMD of the whole software ecosystem instead of a single service provider. In *Model-driven Continuous Experimentation* [GYE22, GBW⁺22], we present a model-driven approach to support software experimentation by connecting those models to component-based software architectures. That work might be used in the future to integrate the knowledge of the product into our approach. In *Situation-specific Design Thinking Processes* [GYNE22d], we investigate the extension of our situation-specific approach toward design thinking workshops. That work might be used in the future to use our approach in workshop settings for developing business models.

1.5 Thesis Structure

This thesis presents the third design cycle of our situation-specific BMD approach. The thesis is structured into ten chapters around three parts, as shown in Figure 1.10. The parts are part 1 of the introduction, foundations, and related work, part 2 of the solution concept, and part 3 of the implementation, evaluation, and conclusion.

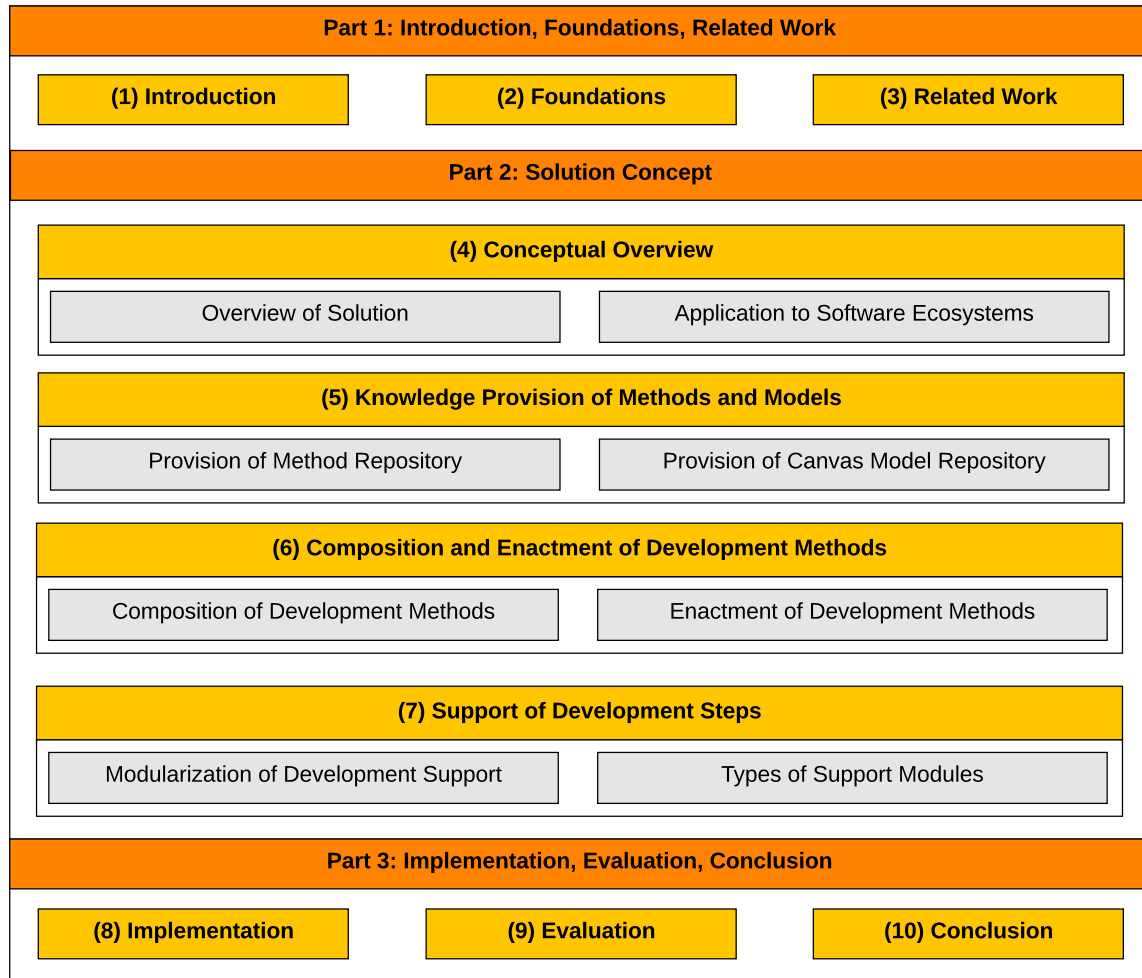


Fig. 1.10 Structure of the Thesis

In **Part 1: Introduction, Foundations, Related Work**, we start the thesis with a general overview. In *(1) Introduction*, we motivate our thesis topic, state our research question, and point out our research contributions. In *(2) Foundations*, we elaborate on the fundamental topics of model engineering, situational method engineering, and business model development on which the thesis is based. In *(3) Related Work*, we introduce related

approaches that use business aspects in situational method engineering and situational factors for business model development, together with tools for business model development.

In **Part 2: Solution Concept**, we present our concept for situation-specific business model development. In (4) *Conceptual Overview*, we show how the overall approach is working. For that, we give an *Overview of [the] Solution* of all stages and show the *Application to Software Ecosystems* as our application area. In (5) *Knowledge Provision of Methods and Models*, we show how corresponding repositories for methods and models can be derived. For that, we show the *Provision of [a] Method Repository* based on different method fragments. Moreover, present the *Provision of [a] Model Repository* based on different model fragments. In (6) *Composition and Enactment of the Development Methods*, we show the composition and enactment based on those repositories. For that, the *Composition of Development Methods* includes the definition of the context, the situation-specific composition of methods, and the domain-specific composition of models. Moreover, the *Enactment of Development Methods* consists of the execution of the development process, the stakeholder involvement inside those steps, and changes in the context. In (7) *Support of Development Steps*, we show how software tools support different development steps. For that, the *Modularization of Development Support* presents the provision of support modules together with their composition and enactment. Moreover, the *Types of Support Modules* show exemplary module types, including the crowd-validation of prototypes.

In **Part 3: Implementation, Evaluation, Conclusion**, we exploit the thesis by applying our approach. In (8) *Implementation*, we show our modularized architecture and our software tool. In (9) *Evaluation*, we evaluate our approach based on a case study on OWL Live and a user study in student courses. In (10) *Conclusion*, we summarize our contributions, revisit the high-level requirements and point out future work.

Chapter 2

Foundations

In the previous chapter, we motivated our approach and presented an overview of our solution. Based on that, this chapter shows the relevant foundations for this thesis. First, we explain model engineering to utilize the knowledge of development methods and canvas models within our approach (2.1). Next, we describe situational method engineering, which concept of composing and enacting development methods we refine in our approach (2.2). Last, we introduce business model development as our application domain (2.3). Finally, we summarize the used parts of the foundations (2.4).

2.1 Model Engineering

Model Engineering (ME) is a discipline of designing, developing, and using models as an abstraction of specific objects or systems within the real world [Béz05]. Models can be defined as “a complex structure that represents a design artifact, such as a relational schema, object-oriented interface, UML model, XML DTD, website schema, semantic network, complex document, or software configuration” [BHP00]. With this, models aim to express and communicate the essential aspects of the real world while avoiding irrelevant details.

In this section, we first give an overview of the metamodeling of models to abstract the essential information (2.1.1). Based on that overview, we show domain-specific languages for developing such models for a specific domain (2.1.2). To support the applicability of those domain-specific languages also for larger models, we present different categories of computer-aided modeling tools (2.1.3).

2.1.1 Metamodeling

Metamodeling is an activity that is often used with model-based and model-driven software development [Béz05]. The focus here is on developing a so-called metamodel from which a set of models can be derived. With this, certain real-world aspects can be abstracted through the models and interpreted according to the metamodels [BCW17]. For this purpose, a metamodel consists of constructs and rules from which the models can be created. In this context, a model created based on a metamodel is also referred to as an instance of this metamodel.

To develop metamodels, bottom-up or top-down approaches can be used. In bottom-up approaches, examples of the real world are observed, and out of the gained knowledge, constructs and rules are derived. This can be done directly out of the real world or by abstracting the real world into exemplary models and deriving the constructs from them. This has advantages if the initial requirements for the metamodels are not fixed. In top-down approaches, the already existing knowledge (and specific requirements of the use case) is used to derive the metamodel, which is then instantiated with models. This has advantages if the initial requirements are fixed and no exemplary models for all use cases exist [BCW17].

To support the metamodeling activity, the Object Management Group (OMG) has defined the Meta-Object Facility (MOF) [Obj16] standard. This standard, in turn, is used for the unified modeling of metamodels and meta-data repositories. For that, MOF defines a self-conforming M3 layer, as shown in Figure 2.1, from which an infinite amount of layers can be instantiated, while at least two are needed. With this, MOF provides high flexibility in defining metamodels. One prominent example of the usage of MOF is the Unified Modeling Language (UML) [Obj17], which is used for the specification and documentation of software parts or whole systems. The illustration of MOF with the four layers of M0, M1, M2, and M3, based on [BCW17], and the example of a mobile to-do app is depicted in Figure 2.1.

The layer **M0: Real World Objects** consists of the objects of the real world that need to be abstracted. For the mobile to-do app, that could be the technical perspective, like the source code or a running system of the app, but also the existing market or the running organization from a business perspective.

The layer **M1: Model** consists of models of the objects in the real world. Here, those models abstract the essential aspects of those objects for a specific use case. For the mobile to-do app, that could be, from a technical perspective, a UML class diagram to represent the structure of the source code with specific instances for those diagrams. Moreover, from a business perspective, it could be the representation of the BMC with filled-out information about the organization as a specific instance of the canvas.

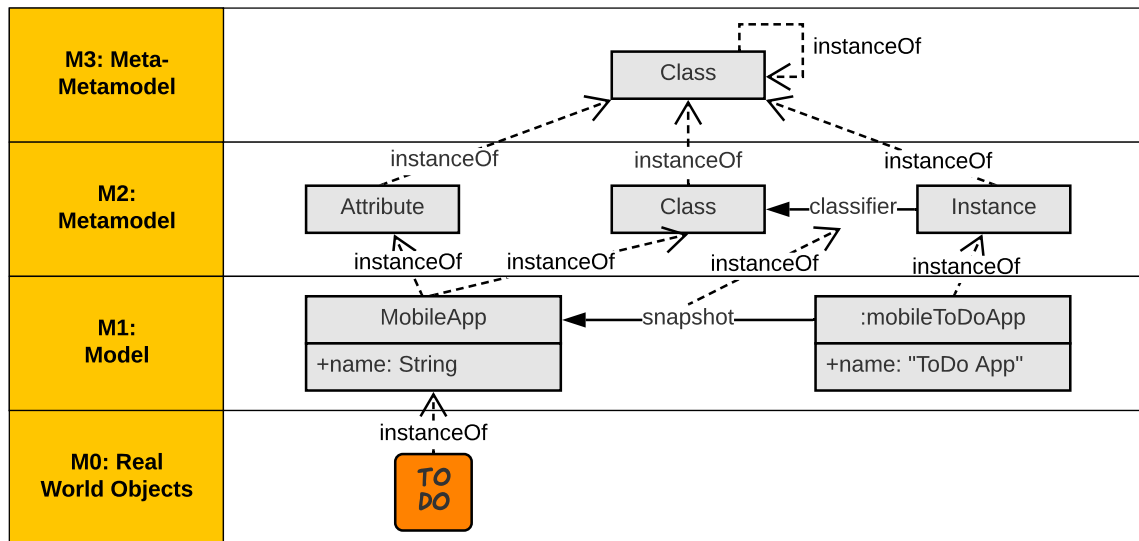


Fig. 2.1 Overview of the Meta-Object Facility (based on [BCW17])

The layer **M2: Metamodel** consists of the metamodels that built the specific models' foundations. Moreover, those layer provides the possibilities of specifying new Domain-specific Languages (DSLs) [BCW17], as explained in Section 2.1.2, and extension mechanisms of existing modeling languages. From the technical perspective of the mobile to-do app, that could be the UML [Obj17] that specifies the UML class diagram. From the business perspective of the mobile to-do app, that could be the Business Model Ontology (BMO) [Ost04], as explained for BMD in Section 2.3.1, which specifies the BMC.

The layer **M3: Meta-Metamodel** consists of MOF as the foundation for the specification of metamodels. Here, MOF is a self-describing language that can define its own language constructs. Both the UML and the BMO can be specified through MOF. With this, the MOF is a DSL that can be used to define metamodels.

2.1.2 Domain-specific Languages

Domain-specific Languages (DSLs) are modeling languages that are designed to fulfill the needs of a specific application domain. For this purpose, a reduced language vocabulary that is already established in the domain is often used [BCW17]. This, in turn, has the advantage of increasing the effectiveness with optimization for the domain and increasing the efficiency by reducing the language constructs. Moreover, DSLs often come with a simplified visual notation to support the understandability and usage by non-experts of the domain [Fra13]. In contrast, General Purpose Languages (GPL), like UML [Obj16], provide support for a larger

and more complex set of domains [Völ13]. The development of a DSL can be supported by a clear definition of the modeling language and a development method.

For the **Definition of a Modeling Language**, particularly a DSL, the definition of syntax, semantics, and notation, together with mechanics (or pragmatics) for later usage, is needed [KK02]. *Syntax* describes the structure of the different elements of the language and how those elements can be combined. Those combinations are often specified by using metamodels. *Semantics* describes a modeling language's meaning consisting of a semantic domain and a semantic mapping. Here, the semantic domain describes the concrete meaning using concepts like ontologies or taxonomies, while the semantic mapping connects those meanings to the actual syntax. *Notation* describes the visualization of the language. This can be done by using graphical symbols for the different elements of the syntax. Last, *Mechanics* (or pragmatics) describes how the modeling language is used. Here, the mechanics can be defined for a specific modeling language or inherited from a meta-language that also provides syntax, semantics, and notation for different related languages [KK02].

For the **Development of a Modeling Language**, particularly a DSL, the development method of Frank [Fra13] provides seven-step guidance in structured development. In the beginning, the (1) *Clarification of the Scope and Purpose* is used to justify the rationale of the language and its motivation. Here, it is important to determine the scope, elaborate on the expected benefits and economic outcomes, and look at the long-term perspective and feasibility. Next, the (2) *Analysis of Generic Requirements* analyzes the existing generic requirements for the modeling language. For that, catalogs of existing requirements should be analyzed to determine if they can be used or need to be modified and prioritized. During the (3) *Analysis of Specific Requirements*, the domain-specific details of the language need to be clarified. For that, scenarios can be developed and refined over time from which the requirements are derived and prioritized. For the (4) *Language Specification*, the semantics and syntax of the DSL need to be defined. For that, a glossary of the terms for the DSL should be developed over time as a foundation to develop a new or extend an existing metamodel. Based on that, the (5) *Design of Graphical Notation* develops the notation for the defined syntax to ensure simple usage. For that, existing guidelines and guiding questions can be used. Additionally, the optional (6) *Development of Modeling Tool* supports the usage of the language with a software tool. For that, a new tool can be created, or an existing tool can be extended. Last, the (7) *Evaluation and Refinement* should continuously optimize the DSL. For that, the DSL should be checked against different test cases and current practices. Overall, using existing development methods increases the quality of the DSL.

Based on the language definition and the development method, different DSLs can be defined. DSLs can be used to model a design artifact's structure (i.e., product) or behavior

(i.e., process). One example of modeling the structure is so-called feature models (FMs) that are used for modeling different software variants within SPL with a common graphical representation [ABKS13]. With this, FMs allow a lightweight visualization of the different possible configurations. One example to model the behavior is the Business Process Model and Notation (BPMN), which is used to model the activity flows of business processes by providing a graphical representation [Obj10]. With this, BPMN allows a lightweight visualization of the process flow. Both modelings might be interrelated, like the BMDMs and the BMs in BMD, as presented in Section 1.1. Moreover, different computer-aided modeling tools, like the BMDTs in BMD, can be used to support the modeling process.

2.1.3 Computer-Aided Modeling Tools

To support ME, various computer-aided modeling tools, in short modeling tools, have been developed over the years. Those modeling tools are provided with an extensive and varying set of features that can support different phases of ME. Moreover, those tools can differ in various aspects, like the license (e.g., open-source vs. closed-source), the scope (e.g., entire framework for all phases vs. individual development steps), or the platform [BCW17]. We categorize those modeling tools into the overlapping categories of visualization tools, general-purpose modeling tools, and domain-specific modeling tools.

The so-called **Visual Modeling Tools** are often used as lightweight modeling tools for different modeling languages and other visual graphics. Those tools provide basic functionalities like the creation, reading, update, or deletion of models, the versioning or recovery of models, and the collaboration on or sharing of models with other stakeholders. However, those tools often focus on the free arrangement of symbols for the notation while not considering the syntax and semantics of the underlying languages. This means that graphical symbols of different modeling languages can be freely combined in a single visualization without restricting constraints. Examples of those visual modeling tools are the desktop application Microsoft Visio¹ or the cloud solution of LucidChart².

The so-called **General-Purpose Modeling Tools** are often used as middleweight modeling tools for different languages. Besides the basic functionalities of the *Visual Modeling Tools*, they focus on integrating metamodels to support the modeling process. However, those tools often focus on the syntax and notation of the models while working less on the ontological semantics of the underlying modeling language. This means that out of a modeling

¹Website of Microsoft Visio: <https://www.microsoft.com/en-us/microsoft-365/visio/>

²Website of LucidChart: <https://www.lucidchart.com/>

language, various visual notations according to the syntax can be created. Examples are the non-extended versions of Enterprise Architect³ and Eclipse⁴.

The so-called **Domain-specific Modeling Tools** are often used as heavyweight modeling tools for a single DSL. Besides the basic functionalities of the *Visual Modeling Tools* and the metamodel integration of the *General-purpose Modeling Tools*, they focus on integrating the domain-specific aspects according to the semantic domain. With this, they can take the syntax, the ontological semantics, and the notation of the modeling language into account. This means that the ontological semantics can restrict the overall syntax and the arrangement of symbols for visual notations by, for example, using a predefined vocabulary. The domain-specific modeling tools can be delivered as standalone solutions or extensions for existing modeling tools. Examples for BPMN are the cloud solution of Camunda⁵ or the extension of the BPMN2 Modeler⁶ for Eclipse.

2.2 Situational Method Engineering

Situational Method Engineering (SME), which originates in software development, is a discipline for creating development methods that fit the specific project's situation where they are applied [HSRÅR14]. A method can be defined as "an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products" [Bri96]. With this, methods aim to represent and communicate a structured development towards a predefined purpose.

In this section, we first give an overview of the concept behind method engineering (2.2.1). Based on that, we show different types of SME approaches to design methods, including the method composition and enactment (2.2.2). To support the applicability, we, lastly, present different software tools to support the process of (S)ME (2.2.3).

2.2.1 Concept of Method Engineering

Method engineering is the research field of designing, constructing, and adapting development methods to create information systems [Bri96]. For this purpose, a new method is designed out of parts of existing methods. This, in turn, has the advantage of increasing the effectiveness with optimization for the specific information system and increasing the

³Website of Enterprise Architect: <https://www.sparxsystems.de/>

⁴Website of Eclipse: <https://www.eclipse.org/eclipseide/>

⁵Website of Camunda: <https://camunda.com/>

⁶Website of the BPMN2 Modeler: <https://www.eclipse.org/bpmn2-modeler/>

efficiency by conducting just the necessary development steps. For that, method engineering often uses an existing and continuously updated method base from which different methods can be constructed [HSGP10]. The process of method engineering can be supported by a clear definition of the method and a development approach.

For the **Definition of a Method**, the method can be decomposed into different parts. That decomposition can be done into method fragments, method chunks, and method components [HSRÅR14], together with method patterns [FB16]. A *Method Fragment* is a reusable atomic part of a method that can have a process- (called work unit), product- (called work product), or producer-focus. According to the triangle of producer, work unit, and work product [HSGP10], the producer performs the work unit to produce a work product. The work product, in turn, is created, evaluated, or iterated by the work unit. This triangle is also used in Software & Systems Process Engineering (SPEM) [Obj08], a software engineering DSL, which can be used to model the method modeling activities. The focus of fragments can be further refined based on the application of the method. A *Method Chunk* is the direct combination of a producer- and product-fragment into a method part. While this reduces the complexity of the method construction due to decreased amount of parts, it also reduces the flexibility of the constructed methods. A *Method Component*, in turn, consists of input and output work products together with a work unit to transform the inputs into the outputs. With this, single fragments can be combined into larger development steps [HSRÅR14]. A *Method Pattern* denotes a structured sequence of method components. By using patterns, the method components can be combined with the constructed development method [FB16]. To identify the different method parts, Ralyte [Ral04] presents the two principles of existing method re-engineering and ad-hoc construction. *Existing Method Re-engineering* splits existing (non-modular) methods into their different atomic parts based on decomposition and exploration. Here, the decomposition takes the whole method and transforms it into method parts by identifying product and process models. Moreover, the exploration identifies additional method parts by analyzing different use cases of the given method. This has advantages if multiple methods already exist for the application area. *Ad-hoc Construction* creates the method parts from scratch by analyzing new application areas where existing methods have not been applied or insufficiently solve the given use cases. This has the advantage of not getting biased by existing methods. Moreover, the development of a method can be supported with an underlying meta-method. Here, Sauer [Sau11] proposes MetaME, which uses a product and a process dimension. For the *Product Dimension*, different artifacts are derived from software engineering concepts. For the *Process Dimension*, those artifacts are modified due to different software development tasks.

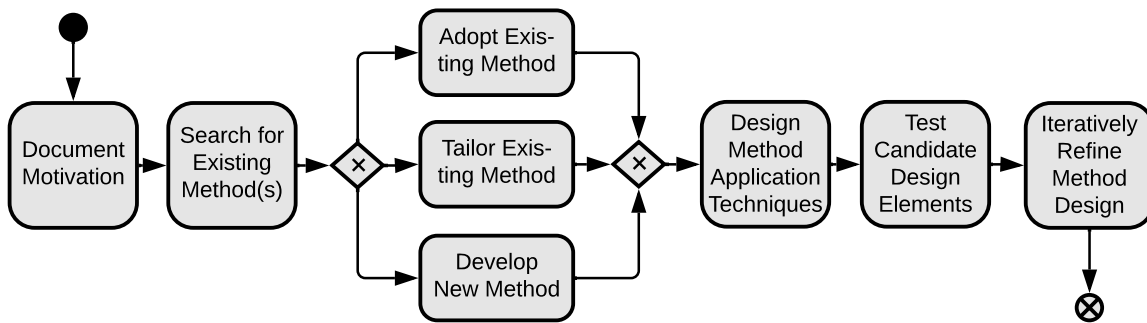


Fig. 2.2 Concept of Method Engineering (based on [MCF⁺95])

For the **Development of a Method**, the approach of Mayer et al. [MCF⁺95], as depicted in Figure 2.2, consists of six steps. In the beginning, it is important to (1) *Document [the] Motivation* behind the method development. This can be done by, for example, isolating the basic intuitions for the method, identifying potential stakeholders, and analyzing the shortcomings of existing methods. After that, a (2) *Search for Existing Method(s)* needs to be conducted. Here, the goal is to find methods that already fulfill parts of the previous motivation for a new method. Based on the results, it is possible to adopt an existing method, tailor it, or develop a new one. During the (3.1) *Adopt[ion] [of an] Existing Method*, an existing method can be reused because it fulfills all requirements of the motivation. During the (3.2) *Tailor[ing] [of an] Existing Method*, an existing method is modified so that it fulfills the requirements behind the motivation. During the (3.3) *Develop[ment] [of a] New Method*, the method is created from scratch out of the requirements behind the motivation. After that, the (4) *Design [of] Method Application Techniques* is needed to define under which different circumstances the method should be used. Here, it is crucial in which scenarios and under which conditions the proposed development method should be applied. This application technique should then be validated by (5) *Test [the] Candidate Design Elements*. Here, the proposed method is applied to test its applicability. The results of those tests are used to (6) *Iteratively Refine [the] Method Design*. The usage of structured development methods increases the quality developed method. Different types of SME approaches have been proposed to develop individual methods for specific projects.

2.2.2 Types of Situational Method Engineering Approaches

SME develops a method directly for a specific project based on its situational context. The goal is to manage a set of projects with specific methods that are created from the same set of method parts. This creation, in turn, is often done by the role of a method engineer, while the role of a project manager does the management. To apply SME, various approaches

have been developed over time. By analyzing the *Trade-Off of Effort vs. Flexibility* and the *Flexibility of Tailoring*, Fazal-Baqaie [FB16] classifies those approaches into the three categories of *Configuration-based*, *Assembly-based*, and *Creation-based* approaches. As shown in Figure 2.3, those categories are originated between the usage of a complete *Fixed* defined method and a totally *Free* definable method.

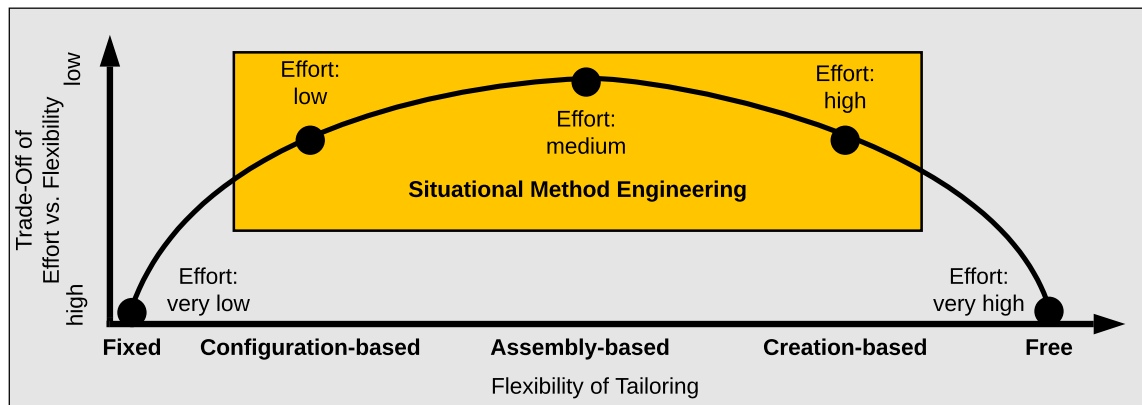


Fig. 2.3 Different Types of SME Approaches (based on [FB16])

The **Configuration-based Approaches** are based on a set of configuration points where the method engineer can configure the situation of his project. Out of that configuration, an automated process creates a tailored method for the project manager without requiring additional changes. With this approach, the tailoring of a situation-specific method needs little effort. Those approaches have the advantage that direct tailoring ensures a high quality of the method by restricting the possible configurations. However, due to the fixed configuration points, the methods cannot be customized by the method engineer, and the methods can not be directly applied to unknown situations.

The **Assembly-based Approaches** are based on a repository of different method parts from existing methods that can be continuously updated. The method engineer assembles a new method from this repository that the project manager uses. During the usage, the project manager informs the method engineer about the occurring quality problems of the method that the method engineer needs to be changed. With this, the tailoring can be done with a medium effort. Those approaches have the advantage that the method engineer can customize the methods, and method parts for unknown situations can be added directly to the repository. However, they have higher upfront costs for providing the repository and ensuring a high quality of the tailored method.

A high-level overview of the steps for assembly-based SME is shown in Figure 2.4 (adapted from [HSRÅR14]). Here, we have the roles of the *Method Engineer* and the *Project*

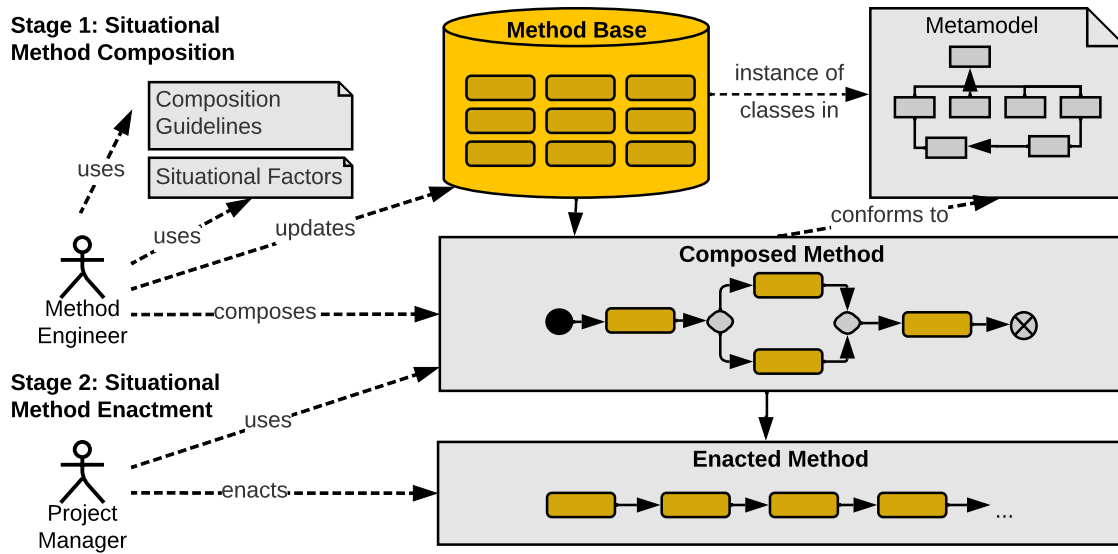


Fig. 2.4 Stages of Assembly-based SME (based [HSRÅR14])

Manager, together with the stages of situational method composition and situational method enactment. In **Stage 1: Situational Method Composition**, the *Method Engineer* needs a *Method Base* consisting of different method parts that can be created by the engineer or gathered from an external source. Here, the *Method Engineer* uses the *Situational Factors* of the project and additional *Composition Guidelines* to construct a *Composed Method* that conforms to the underlying *Metamodel*. In **Stage 2: Situational Method Enactment**, the *Project Manager* uses the method as *Enacted Method* to develop the software within the project. Here, he can also report quality problems with the method.

The **Creation-based Approaches** are based on a metamodel with general composition constraints but no explicit repository with predefined method parts. With the support of that metamodel, the method engineer needs to formalize the needed method from scratch for each project of the project manager. For that, the method engineer can use his experience from past projects. With this, the method creation can be done with high effort. Those approaches have the advantage of providing high flexibility in the construction of the method and their adoption in unknown situations. However, they have high runtime costs by developing the method from scratch and issues providing a high quality of the created method.

To assist the development of (situation-specific) methods using the configuration-based, assembly-based, and creation-based SME approaches, computer-aided method engineering tools can be used.

2.2.3 Computer-Aided Method Engineering Tools

To support SME, different computer-aided method engineering tools, in short method tools, have been developed over the last years. Those tools can have various functions like representing different methods, administrating the method parts, selecting the method parts, and assembling the methods [BH95]. We categorize those method tools into the overlapping categories of method visualization tools, method modeling tools, and method engineering tools. Those tools can also be seen as a subset of the modeling tools in Section 2.1.3.

The so-called **Method Visualization Tools**, which can be seen as a subset of the *Model Visualisation Tools*, are used to define a fully customizable method freely. For that, those tools provide a graphical interface to visualize the tailored method and modify it. However, those tools do not provide quality assurance of the developed method by using an underlying syntax. This means that the method engineer is fully responsible for constructing the tailored method out of a repository of graphical symbols. Examples are BPMN symbol repositories within the already mentioned tools of Microsoft Visio and Lucidchart.

The so-called **Method Modeling Tools**, which can be represented within the *General-Purpose Modeling Tools*, are used to define a method from scratch using an underlying syntax. Those tools often provide modeling support based on the metamodels of existing method modeling languages to allow a conformance checking of the developed method. However, those tools provide just a limited form of quality assurance due to the lack of a semantic understanding of the usage of the method. This means that the method engineer is fully responsible for tailoring the method according to the correct semantics. Examples are the MetaEdit+ Domain-Specific Modeling Environment⁷ [ALB⁺09] and the BPMN2 Modeler for Eclipse⁸

The so-called **Method Engineering Tools**, which can be represented within the *Domain-Specific Modeling Tools*, are used to define a method from an existing repository of method parts. For that, those tools often support managing such a repository, which also represents parts of the semantic domain. With this, the methods can be tailored directly according to the syntax, the semantic domain, and the visual notation. This means that the method engineer has a good guidance tailoring method with high quality. Examples are the generic ADOxx Metamodeling Platform⁹ [FK15] and the specific Method Engineering with Method Services and Method Patterns [FB16].

⁷Website of the MetaEdit+ Domain-Specific Modeling Environment: <https://www.metacase.com/>

⁸Website of the BPMN2 Modeler: <https://www.eclipse.org/bpmn2-modeler/>

⁹Website of the ADOxx Metamodeling Platform: <https://www.adoxx.org/>

2.3 Business Model Development

Business Model Development (BMD) is a discipline for creating new or innovating existing business models under high uncertainty [ADv13]. A business model can be defined as “the rationale of how the organization creates, delivers, and captures value” [OP10]. With this, business models aim to provide a detailed representation of the desired business strategy of the organization and an abstraction of the value creation mechanism of the underlying business processes of the organization [ADEHA08].

In this section, we first give an overview of the modeling languages for the visualization of business models (2.3.1). Based on that, we show development methods to develop such business models (2.3.2). To support the applicability of business model development, we show different software tools to support the development (2.3.3).

2.3.1 Business Model Modeling Languages

Business Model Modeling Languages (BMMLs), which could be created using model engineering as explained in Section 2.1.2, are DSLs that are used to support a common understanding and the communication of business models [OPT05]. For that, those BMMLs can range from the pure visualization of the business model using a graphical notation to modeling support using syntax and semantics. For the visualization, Taeuscher and Abdelkafi [TA17] analyzed various visual business model representations from their cognitive perspective and grouped them into three categories of element view, transactional view, and casual view. Based on this categorization, they derived a recommendation of which graphical visualizations can be used for which phase of BMD. This has the advantage of reducing the user’s cognitive load of the representation. For modeling support, John et al. analyzed different BMMLs regarding their syntax, semantics, and pragmatics [JKS17]. From the visualization, they divide those BMMLs into connection-based and geometric-based languages together with their hybrid combination.

The **Connection-based Languages** use freely-definable structures based on graphical symbols and their connections. With this, those languages can be used to model relationships within and between the business models of different organizations (e.g., multiple service providers and their relationships within a software ecosystem). For that, there exists purely visualization languages (e.g., ValueMap [All00], Value Stream Map [PHS08]) where those symbols can be arranged without any guidance and modeling languages (e.g., e3-Value [GA01], Strategic Business Model Ontology [SYT09]) where a syntax in the form of a metamodel and ontological semantics of the business model domain is used. The most prominent example of the connection-based approach is the e3-Value language [GA01,

AG03], which can be used to model and analyze different actors within a value network. For that, the approach allows the modeling of the different actors and market segments. Those actors and market segments have interfaces with different ports (e.g., subscriptions) to exchange certain objects (e.g., money). Moreover, those initiated models can be used to calculate the business outcome of different predefined scenarios.

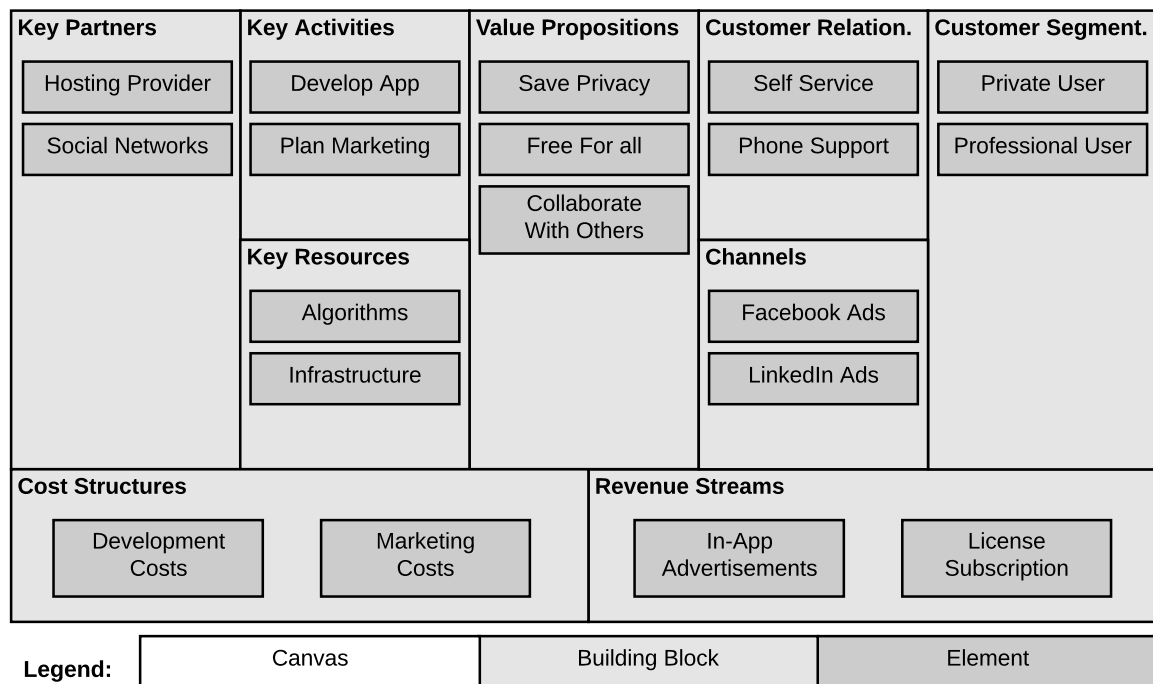


Fig. 2.5 Overview of the Business Model Canvas (structure from [OP10])

The **Geometric-based Languages** use fixed structures based on solid geometric boxes where different information can be defined. With that, those languages can be used to structure the information for the business model of a single organization (e.g., service provider in a software ecosystem). While the representation of the boxes is the graphical notation, the arrangement can be interpreted as syntax and their labeling as ontological semantics. The most prominent example and de-facto standard in business modeling is the Business Model Canvas (BMC) [OP10] as shown in Figure 2.5. The BMC as a *Canvas* divides the business model up into the nine *Building Blocks* of the *Value Propositions*, the *Customer Segments*, the *Customer Relationships*, the *Channels*, the *Key Partners*, the *Key Activities*, the *Key Resources*, the *Revenue Streams*, and the *Cost Structures*. Those building blocks are filled out with *Elements* about the business. In Figure 2.5, an example for filling out the BMC for a mobile todo app is given. Here, the *Private User* is an element of the *Customer Segments*, the *Save Privacy* is an element of the *Value Propositions*, and the *In-App Advertisements* is an element of the *Revenue Streams*. Additionally, the ontological semantics is improved by

Osterwalder by providing guiding questions (e.g., For whom are we creating value? Who are our most important customers? for *Customer Segments*) and example elements (e.g., Mass Market, Niche Market, Segmented, Diversified, Multi-sided Platform for *Customer Segments*) [OP10]. Moreover, the relationship between the different elements can be displayed by using different colors [FP10]. Due to the success of the BMC, other canvas models like the Value Proposition Canvas (VPC) [OPB⁺14] for developing a value proposition of the customer and the Platform Canvas [SSSV19] for developing platform business models have been proposed.

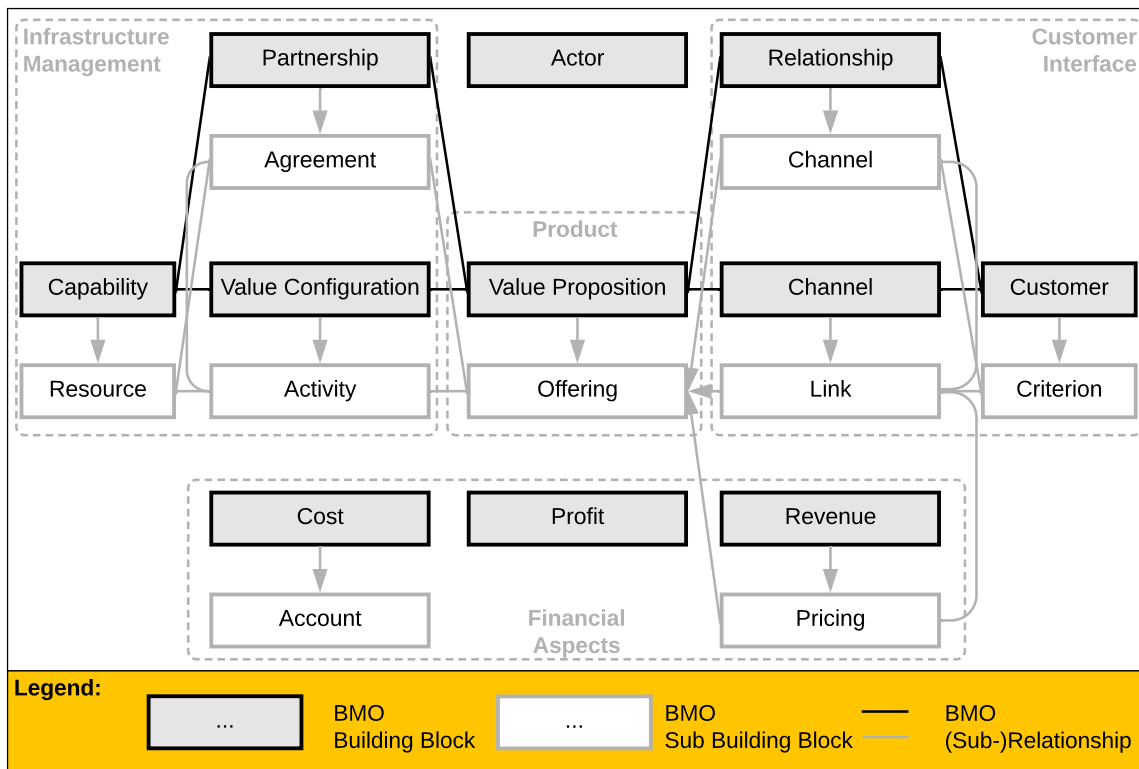


Fig. 2.6 Overview of the Business Model Ontology (based on [Ost04])

While the BMC uses fixed boxes to reduce the cognitive load, the underlying thesis of Osterwalder [Ost04] proposed an enhanced connection-based Business Model Ontology (BMO). As shown in the overview in Figure 2.6, the BMO consists of additional *BMO Building Blocks* (e.g., Actor), decomposed *BMO Sub Building Blocks* (e.g., Offering), and *BMO (Sub-)Relationships* (e.g., Activity to Offering) around the four categories of the *Product*, the *Customer Interface*, the *Infrastructure Management*, and the *Financial Aspects*. With this enhanced ontological semantic, it is also possible to provide enhanced development support for the BMD. For example, based on the BMO, system dynamics can be integrated to support dynamic business modeling [CN18] for the calculation of possible business outcomes, or

the business model can be mapped to an enterprise architecture [MIN⁺12] for aligning the business with the IT. This has the advantage of deeper integrating the business modeling to related application areas and supporting information exchange.

2.3.2 Business Model Development Methods

Business Model Development Methods (BMDMs), which could be created using method engineering as explained in Section 2.2.1, are used to guide the development of new or innovation of existing business models. Here, BMD is a complex and creative activity that consists of different phases where multiple tasks need to be conducted. Inside those tasks, communication and collaboration between different stakeholders are needed [EHB11], different alternative business models have to be developed [SEP⁺19], and uncertainties need to be reduced [McG10].

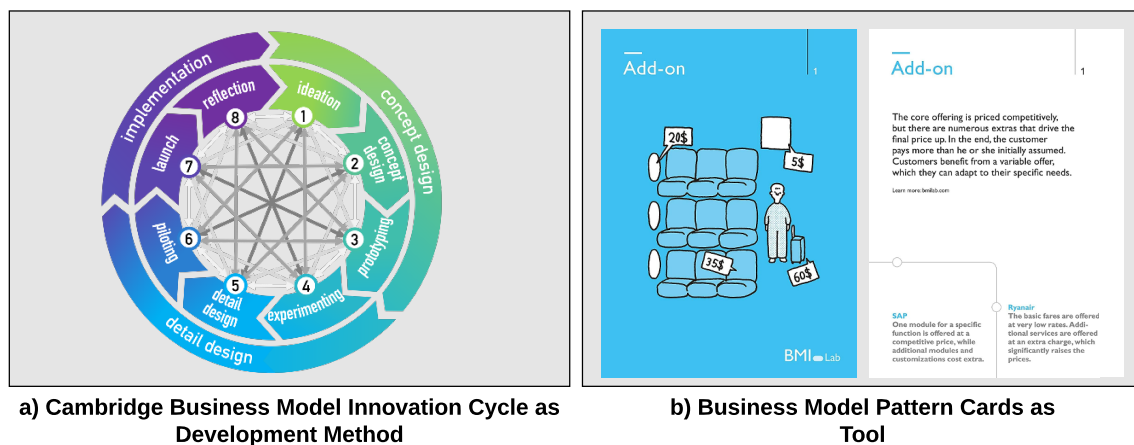


Fig. 2.7 Overview of a Development Method and a Tool (images from [GSE17, GFC14])

The **Development Method** structures the whole activity of BMD. For that, there exist methods that stay on a higher description level with a loop of continuous experimentation (e.g., Discovery-driven Planning [McG10], Try-And-Error Learning [STRV10]) and methods that provide detailed descriptions of different phases and corresponding tasks (e.g., Cambridge Business Model Innovation Process [GSE17], Customer Development [Bla20]). One exemplary method is the Cambridge Business Model Innovation Process, as depicted in Figure 2.7 (a), by Geissdoerfer et al. [GSE17]. The authors developed a business model innovation (BMI) process for the transformation of a single organization. For that, they divide that process into three phases with eight steps. In the beginning, the (1) *Ideation* has the tasks of formulating a vision of the business, defining needed stakeholders, ideating on the value, analyzing the sustainability, and selecting and evaluating ideas. Based on that, the

(2) *Concept Design* integrates those ideas, discusses current trends, and defines the value creation, capture, and delivery. Based on those results, the (3) *Virtual Prototyping* has the steps of benchmarking the ideas with industry and generic business models, building the prototype, and evaluating and selecting the prototype. To evaluate the prototype, the (4) *Experimenting* identifies the key variables of success, designing the experiments, running the experiments, and analyzing the learned lessons. After that, the (5) *Detail Design* provides a refinement by defining all business elements in detail, summarizing a specific overview, and providing all information in a transformation tool. To take action, the (6) *Piloting* plans and implements parts of the transformation, analyzes the implementation, adjusts according to the results, documents the transformation, and identifies possible failures. After piloting parts of the transformation, the (7) *Launch* realizes and implements the whole planning and scaling everything up. In the end, the (8) *Reflection* monitors the transformed business, reflects those changes, adjusts and scales needed changes, and starts a new innovation process. As shown in Figure 2.7 (a), the whole method consists of a non-linear process with the possibility of choosing the next step according to the results of the last step.

The development can be supported with different **Additional Tools**. Within this thesis, we focus on the usage of patterns and taxonomies together with software-based tools. *Patterns* describe reusable combinations of business model elements for a certain outcome. For that, those patterns abstract knowledge that repeatedly occurs in multiple organizations. With this knowledge, the pattern can be recombined to support the generation of new business models. As an example, Gassmann et al. [GFC14] introduced a set of business model pattern cards, as shown in Figure 2.7 (b), which they derived from the analysis of existing companies. Here, each card consists of a name (e.g., Add-on), a visual marker (e.g., airline seating), a short description (e.g., the core of offering...), and exemplary companies (e.g., SAP, Ryanair). Moreover, they published more details about each pattern inside a handbook [GFC14]. *Taxonomies* describe a schema to classify different business model elements. For that, those taxonomies abstract knowledge that exists in at least one organization. With this knowledge, it is possible to compare different organizations or discover new elements to support the idea generation of possible business models. As an example, Hartmann et al. [HZFN16] developed a taxonomy for data-driven business models from a diverse set of sources. Here, their taxonomy uses the dimensions of the data source, the key activity, the offering, the target customer, and the revenue model, which dimensions are further refined like the revenue model into an asset sale, lending/renting/leasing, licensing, usage fee, subscription fee, and advertisement. Moreover, those dimensions are clustered into different types like a free data collector and aggregator or analytics-as-a-service. For *Software-based Tools*, BMDTs exist that partially provide support for patterns and taxonomies.

2.3.3 Business Model Development Tools

To support the BMD, different Business Model Development Tools (BMDTs) have been developed in research and practice over the last years. Here, those tools in practice focus on the visualization of and collaboration on business models using different aspects like modeling (e.g., customization, development, commenting and linking, assessment, navigation, and filter), collaboration (e.g., communication, synchronization, user and role management, repository and conflict management), and technical implementation (e.g., architecture, data exchange) [SSJ⁺19]. Moreover, those tools in research develop solutions for supporting the development in terms of design and decision-making using aspects like user and task characteristics (e.g., different maturity of users), the generation of business models (e.g., pattern matching), or the validation of business models (e.g., crowd-validation) [BdRHF20]. However, there is a trend for adding more and more decision support to those tools [OP13b]. A list of existing BMDSSs derived using an SLR can be seen in Appendix A. Based on our findings, we categorize those BMDTs into the overlapping categories of visualization support tools, design support tools, and decision support tools.

The so-called **Visualization Support Tools**, which are often developed and used in practice, mainly build upon the structure of the BMC or similar *Geometric-based Languages*. For that, those tools visualize the business model, show modification, provide versioning, and enable collaborative development. However, while the focus is on simple collaboration based on a graphical notation, those tools often lack a clear syntax and semantics behind the developed business models. This means that the developed business models of an ideation process can not be completely interpreted and further processed by the software. Examples are the online versions of Strategyzer¹⁰ and Canvanizer¹¹.

The so-called **Design Support Tools**, which are often developed in research, but to some degree, also used in practice, often use the simple structure of the BMC together with an invisible structure (e.g., metamodel) with additional constraints to support the design. For that, those tools support the design phase of the business model with additional guidance like recommending business model elements or checking existing business model configurations. However, those tools often focus on using the existing syntax and ontological semantics only during the design of the business model. This means that those tools do not include other phases of the BMD, like the validation with experiments or the calculation of business outcomes. Examples are the Smart Business Model Developer [LFCJ⁺18], with the usage of patterns in the design, or the Question-based Business Model Configurator [RRE⁺19], with the usage of taxonomies in the design.

¹⁰Website of the Strategyzer App: <https://www.strategyzer.com/app>

¹¹Website of Canvanizer: <https://canvanizer.com/>

The so-called **Decision Support Tools**, which are currently often developed just in research, provide additional decision support to the BMD. Depending on the use case, those tools support different phases of the BMD, like the initiation of the business model from existing data sources, the innovation of business models through transformations, or the validation of business models with the crowd's support. For that, those tools need a well-defined syntax and ontological semantics, for example, on ontologies with connections between business model elements. Examples are the Business Model Analyzer [AFM18] with the support for transforming the business model and the Hybrid Intelligence Decision Support System [DLEL19] with the support to validate business models with the crowd.

2.4 Summary

Within this chapter, we have provided the foundations behind our situation-specific BMD approach. For that, we have given an overview of the topics of model engineering, situational method engineering, and business model development.

For **Model Engineering**, we have shown the creation of metamodels, the development of DSLs, and the existing software tools for model engineering. In our approach, we want to apply those foundations by providing a DSL and metamodels to formalize the knowledge of the development methods and the business models. For that DSL, we need to specify syntax, ontological semantics, and a visual notation.

For **Situational Method Engineering**, we have introduced the concept of method engineering, shown different types of SME, and explained existing software tools for SME. In our approach, we want to use Assembly-based SME to provide repositories for the development methods and business models together with the composition and enactment of the method. For that, we need to provide metamodels for both repositories together with a situation-specific construction process that can be changed during the enactment.

For **Business Model Development**, we have presented different modeling languages and development methods together with supporting software tools. In our approach, we want to formalize the knowledge of different BMDMs as development methods and the knowledge of Geometric-based BMMLs as business models. For that, we need to consider specialties of the semantic domain of BMD during the formalization process.

Based on those foundations, the next chapter shows the related work of our approach concerning our derived HRs. Here, we will analyze the business aspects of SME approaches, the situational aspects of BMD approaches, and the existing BMDTs.

Chapter 3

Related Work

In the previous chapter, we provided the different disciplines of our thesis foundations. Based on that, we give an overview of approaches in those disciplines that are related to our research question and the derived high-level requirements. For that, we first show how business aspects are currently integrated into approaches for situational method engineering (3.1). After that, we analyze how situational aspects are supported in approaches for business model development (3.2). Next, we investigate how current tools for business model development solve those challenges of situation-specific development (3.3). Last, we summarize our gained results on the related work (3.4).

3.1 Business Aspects in Situational Method Engineering

Various approaches for SME have been introduced in the literature, where some of them cover different business aspects during their development. In the following, we describe related approaches (3.1.1) and compare them against our HRs (3.1.2).

3.1.1 Related Approaches

During the last years, different approaches using SME have been developed that also cover business aspects within their methods. Here, those approaches have a wide range from general frameworks to support the (method) modeling over approaches with a customizable method repository for software engineering to approaches with fixed repositories that contain business-related situational factors, tasks, and artifacts. In the following, we describe selected situational approaches (SA) related to our research.

The **SA 1: ADOxx Meta Modelling Platform**¹ [FK15] is a platform that supports the design of domain-specific modeling methods. For that, ADOxx introduces the ADOxx metamodel from which user-specific metamodels for different modeling methods could be derived. Those modeling methods are specified by a modeling technique that consists of a modeling language (i.e., notation, syntax, semantic) and a modeling procedure (i.e., steps, results), and by mechanisms & algorithms that are refined to generic, specific, and hybrid once. Out of those developed metamodels, specific models could be instantiated. For that, ADOxx provides an architecture with the layers of a repository to cover a database and the core modeling techniques, components to allow different functionalities like the analysis, transformation, or import/export of models, and the user interface to provide interactions with the user. Moreover, based on the platform, different (method) modeling languages like UML or BPMN have been realized.

The **SA 2: MetaEdit+ Domain-Specific Modeling Environment**² [TK09] is a tool suite that supports collaborative (method) modeling based on DSLs. It consists of the MetaEdit+ Workbench to design a modeling language and the MetaEdit+ modeler to use a modeling language. The MetaEdit+ Workbench allows the definition of the general concept, the inclusion of different rules as modeling constraints, the association of language concepts to visual notations as symbols, and the creation of source code with the definition of generators. The MetaEdit+ Modeler uses the defined modeling language and allows the collaborative graphical development of models. The tool can be fully integrated into existing toolchains and supports modifying the (method) modeling language due to changing requirements.

The **SA 3: Method Engineering with Method Services and Method Patterns** [FB16] is an approach to support the project-specific composition, enactment, and quality assurance of software engineering methods. For that, the approach consists of different roles and stages of the method content definition, method tailoring, and method enactment. The method content definition is conducted by the senior method engineer, who defines method elements, method building blocks, and method patterns. The method tailoring is conducted by the project method engineer who characterizes the project, composes the project-specific method, and assures the quality of that method. The method enactment is conducted by the project team, who enacts the method and reflects the quality of it. Moreover, the stage of method content definition and method tailoring are implemented using the Eclipse Modeling Framework, while the enactment is based on the WSO2 Business Process Server. The approach has been already transferred to software modernization in [Gri16]. Here, the author presents an SME framework to guide the development of situation-specific transformation methods. For

¹Website of ADOxx Meta Modelling Platform: <https://www.adoxx.org/>

²Website of MetaEdit+: <https://www.metacase.com/>

that, the method repository is based on the domain of model-driven engineering, while the composition focuses on the identification of concepts within the existing legacy system.

The **SA 4: Method Engineering for Blockchain Use Cases** [FLR⁺18] is an approach that uses action design research and SME to construct a development method for blockchain use cases. Based on existing literature, including general method requirements, they designed the development method and evaluated it in various workshops with different organizations. The development method consists of the six phases of understanding the technology, getting creative-unbiased, glancing in the market, getting creative-informed, structuring ideas, and prototyping. For every phase, they identified the involved roles (e.g., moderator and participants in structure ideas), the possible techniques (e.g., cluster ideas, discuss common features of each cluster), the used tools (e.g., mind map, business model canvas), and the output (e.g., structured and prioritized blockchain use case clusters, number of MVPs suitable for the prototyping activity).

The **SA 5: Method for Engineering Gamified Software** [MHW18] is an approach that provides a comprehensive method for developing gamified software. Using DSR, they conducted a literature review and expert interviews to develop a knowledge base, including design principles for such a method. Out of the knowledge base, they created a method base from which they iteratively select method fragments, assemble them into a development method, and evaluate that method in expert interviews and a case study. The development method consists of the seven phases of project preparation, analysis of context and users, ideation, design, design implementation, evaluation, and monitoring. During the method, they consider the context of the project and the user together with business-related tasks (e.g., create personas in analysis) and artifacts (e.g., prototype in design).

The **SA 6: Situation-specific Construction of Internet-of-Things Development Methods** [GT18] is an approach that provides situation-specific development methods for IoT cases. The authors analyzed different IoT development methods in the literature to develop a corresponding method based on method fragments. Here, the developed method base also contains business-related situational factors (e.g., business regulations), tasks (e.g., producing an IoT Canvas), and artifacts (e.g., IoT Canvas). Out of that method base, they allow the construction of development methods for IoT cases by selecting the method fragments, assembling them into a method, and validating the method.

The **SA 7: Domain-specific Modeling Method for Supporting the Generation of Business Plans** [WF20] is a concept to generate textual business plans out of the BMC. They developed a metamodel of the modeling language and a modeling process. For the metamodel, they extended the nine building blocks of the BMC (e.g., revenue streams) with additional relationships (e.g., pays) between them. For the modeling process, they defined the

steps of creating ideas within a simplified transaction model, creating a snapshot of the ideas as a BMC, filling out additional information and preferences for the business plan document models together with generating, editing, and exporting the business plan. To support the usage, they implemented the whole concept on top of ADOxx.

The **SA 8: Situational Approach to Data-driven Service Innovation** [vvR19] proposes an approach to assembly data-driven service development methods based on the context for which the service is created. For that, the authors focused on the development of a metamodel for the method fragments based on existing literature. Here, each method fragment contains a descriptor of the reuse context and the reuse intention, an interface of the situation and the intention, and a body with multiple activities, results, techniques, and roles. Moreover, they outlined an assembly process with the steps of ideation of a method, selection of method fragments, the realization of the method, and use of the service.

The **SA 9: Multi-concern Method for Identifying Business Services** [ATO⁺20] proposes an approach to develop methods for identifying and defining business services within organizations. For that, the authors conducted a literature review to identify and match such methods against a predefined set of requirements. They extracted method chunks from those methods and used an assembly-based technique to combine them into a new development method. As input and output of those method chunks, they used business-related modeling techniques like business process and feature models.

The **SA 10: Design Thinking Process Model based on Method Engineering** [TM11] proposes a formal model to guide design thinking processes. Based on observations in an educational context, the authors used method engineering to extract a process model for design thinking. The process model consists of six phases named to understand, observe, point of view, ideation, prototyping, and test. Within the phases, they suggested conducting business-related tasks (e.g., clustering ideas in ideation) and creating business-related artifacts (e.g., user journey in point of view).

3.1.2 Requirement Comparison

To analyze the existing SA, we compare those approaches against our HRs, as shown in Table 3.1. Here, we must mention that those approaches are not designed primarily for developing business models, and therefore not all high requirements are fully comparable. Moreover, ADOxx and MetaCase provide just general frameworks where possible BMD techniques would need to be implemented on top. Nevertheless, those approaches show that business aspects are already covered by different of them. For the analysis, we divide that comparison into our three stages of knowledge provision of method and models, composition and enactment of development methods, and support of development steps.

Tool	HR: 1	HR: 2	HR: 3	HR: 4	HR: 5	HR: 6	HR: 7	HR: 8	HR: 9	HR: 10
SA: 1										
SA: 2										
SA: 3										
SA: 4										
SA: 5										
SA: 6										
SA: 7										
SA: 8										
SA: 9										
SA: 10										
Legend	No Fulfillment (○)		Partial Fulfillment (◐)			Complete Fulfillment (●)				

Table 3.1 Comparison of the Situational Approaches (SA) against the High-level Requirements (HR)

The aim of **(1) Knowledge Provision of Methods and Models** is to utilize the existing knowledge to make it (re)useable within the BMD. Here, we have the three requirements of knowledge utilization, method comprehensiveness, and model visualization, together with the cross-cutting requirement of tool support. For *HR 1: Knowledge Utilization*, we want those approaches to provide knowledge of BMDMs and BMs based on unified interpretable structures. Here, some approaches provide tables to group knowledge into categories (e.g., SA 5), simplified metamodels to compose developed methods out of a method repository (e.g., SA 8), and general frameworks that could handle the knowledge utilization by defining DSLs for it (i.e., SA 1, SA 2). For the *HR 2: Method Comprehensiveness*, we want those approaches to cover all phases of different BMDMs. Here, some approaches cover the development of a single development method (e.g., SA 4), some approaches cover the situation-specific construction of such methods for a specific case (e.g., SA 9), and a few approaches provide general approaches to the creation situation-specific method bases for different cases (e.g., SA 3). For *HR 3: Model Visualisation*, we want those approaches to cover different visualizations for the BMs. Here, a few approaches use no visualization for their modeling artifacts (e.g., SA 4), some approaches refer to external models that could be used within the development

method (e.g., SA 9), and some use explicit defined (canvas) models (e.g., SA 7). For *HR 10: Tool Support*, we want those approaches to provide an open-source implementation of their solution that can be extended. Here, a lot of approaches provide just conceptual models without concrete software (e.g., SA 5), some approaches show tools but do not make them accessible (e.g., SA 3), and the frameworks make their software accessible (i.e., SA 1, SA 2). However, based on that analysis, no approach formalizes the methods consisting of development steps and modeling artifacts consisting of different canvas visualizations together with an accessible tool whose source can be downloaded and modified. Here, the two overall frameworks would cover most requirements based on DSLs that need to be developed.

The aim of **(2) Composition and Enactment of Development Methods** is to compose development methods out of the knowledge and enact those methods as processes. Here, we have the three requirements of context awareness, selection assistance, and development continuity. For *HR 4: Context Awareness*, we want those approaches to provide a flexible definition and usage of different contextual factors. Here, some approaches provide just methods without context (e.g., SA 4), some approaches use the implicit definition of the context (e.g., SA 7), and some approaches provide an explicit definition in the form of situational factors (e.g., SA 6). For *HR 5: Selection Assistance*, we want those approaches to provide selection support for the BMDMs and the BMs from existing knowledge. Here, some approaches provide no knowledge or no assistance in selecting the knowledge (e.g., SA 4), some approaches provide a simple selection of methods fragments (e.g., SA 8), some approaches provide a configuration of the models (e.g., SA 3), and some approaches provide a modular composition of the method (e.g., SA 3). For *HR 6: Development Continuity*, we want those approaches to provide a runtime modification of the BMDM and the BM. Here, some approaches provide just a single executable development method (e.g., SA 6), some approaches provide a continuous reflection of the output of the method (e.g., SA 9), and the frameworks could provide a flexible definition of continuity within their toolkits (i.e., SA 1, SA 2). However, based on that analysis, no approaches provide a flexible configuration of methods and models based on existing knowledge that can be easily modified during runtime.

The aim of **(3) Support of Development Steps** is to support certain development steps. Here, we have the three requirements of stakeholder collaboration, artifact management, and decision support. For *HR 7: Stakeholder Collaboration*, we want those approaches to support the collaborative development of the BM by different stakeholders. Here, some approaches are designed for single stakeholders (e.g., SA 4), some approaches cover the explicit definition of the involved stakeholders (e.g., SA 5), and some approaches allow the direct interaction of multiple stakeholders (e.g., SA 2). For *HR 8: Artifact Management*, we

want those approaches to provide management of the created artifact, including the tracing and reasoning of changes. Here, two approaches provide manual management of artifacts (e.g., SA 4), some approaches some basic management by definitions of the artifacts (e.g., SA 3), and some provide full management of changes in those artifacts (e.g., SA 1). For *HR 9: Decision Support*, we want those approaches to provide flexible decision support for different development steps. Here, the analyzed approaches focus not on providing decision support (e.g., SA 10), while it could be implemented on top of the frameworks (i.e., SA 1, SA 2). However, based on that analysis, no approach provides collaborative development of different artifacts with modularized decision-support that can be flexibly applied to all steps.

3.2 Situational Aspects in Business Model Development

Various approaches for BMD have been introduced in the literature, where some of them cover different situational aspects during the development. In the following, we describe related approaches (3.2.1) and compare them against our HRs (3.2.2).

3.2.1 Related Approaches

Over the years, different approaches have been developed to support the whole process of BMD or single phases like the design or the validation. Here, those approaches have a wide range from handbooks that explain the BMD in different situations, repositories of method parts and patterns that can be applied to different situations, and the outcome calculation of different business model configurations. In the following, we describe selected business approaches (BA) related to our research.

The **BA 1: Business Model Generation** [OP10] is a handbook from the creators of the BMC and offers practical guidance for developing business models. For that, they use the visualization of the BMC, together with detailed explanations of the building blocks, including knowledge of examples (e.g., Mass Market, Niche Market for Customer Segments), guiding questions (e.g., How do we raise awareness about our company's products and services? in Channels), and patterns (e.g., Multi-Sided Market). For the method, they use a process of the five phases of mobilizing the business model design project, understanding the different elements for the design, designing the different options and selecting the best, implementing a prototype of the business model in the field, and managing the adaption of the business model due to market reactions, each with different tools and techniques. Moreover, they offer decision support for validating the business model by asking control questions for a SWOT analysis of the building blocks.

The **BA 2: Business Model Innovation through Trial-and-Error Learning** [STRV10] is a framework for BMD based on a case study. For that, the authors work on a dynamic view of the business model over time that needs to be continuously updated due to internal organizational and external market changes (e.g., innovations, competitors, regulations). Here, the innovation of a business model is divided into the two phases of exploitation, based on organizational search, and experimentation, based on try-and-error experimentation. Out of these two phases, they designed a process with the stages of creating an initial business design with testing using the exploitation, developing the business model through experimentation, scaling the refined business model based on exploitation, and growing the business model in the organization sustainable using a combination of exploitation and exploration.

The **BA 3: Business Model: A Discovery Driven Approach** [McG10] is an approach that uses continuous experimentation to deal with highly uncertain, complex, and fast-moving environments. For that, the authors divide the business model into the unit of a business and the key metrics. The unit of a business is the thing (e.g., product, service) for which the customer pays directly (e.g., charging the product) or indirectly (e.g., advertisements). The key metrics are dedicated to selling those units and contain the most critical constraints or rating-limiting steps of the value chain. Here, experimentation for the whole business model can support the validation of the articulated underlying assumptions. For that, a sequence of experiments (e.g., market study, feasibility study) is used to test the main assumptions of the business.

The **BA 4: Startup Owner's Manual** [Bla20] is a book that provides a step-by-step introduction to the development of a business model and a new product for an organization. For that, they provide the two phases of customer discovery and customer validation that are needed within customer development to search for new business opportunities. For customer discovery, they point out the activities of identifying hypotheses, testing the problem, testing the solution, and verifying the results. For customer validation, they specify the activities of preparing sales, selling the product, developing the positioning, and verifying the metrics. Their approach uses the visual structure of the BMC that is continuously changed. Moreover, to support the development, the authors provide various checklists regarding the development steps and the business model itself.

The **BA 5: Cambridge Business Model Innovation Process** [GBH16] is a framework to guide the BMI of an organization by focusing on the gap between the design of the business model and its actual implementation. For that, the authors conducted a literature review and expert interviews to develop a non-linear process of the three phases of concept design, detail design, and implementation together with the eight stages of ideation, concept

design, virtual prototyping, experimenting, detail design, piloting, launch, and adjustments & diversification (see Section 2.3.2 for details). For each stage, they provide necessary activities (e.g., stakeholder definition in ideation) and potential challenges (e.g., communication failures in concept design).

The **BA 6: Testing Business Ideas** [BO20] is a book that provides a repository that can be used to validate existing business ideas. For that, they propose an iterative process to design a prototype of the business and test it with experiments. Here, every experiment consists of a name, a description, tasks to accomplish, needed capabilities of the organization, and an estimation of cost, evidence strength, setup time, and runtime. Moreover, they provide different experiment sequences (e.g., customer interview before paper prototype) that can be used for different contexts (e.g., b2b hardware business, highly regulated field). During the process, they use various canvas models to describe and sort the different business ideas.

The **BA 7: Business Model Navigator** [GFC14] is a book together with a set of pattern cards that support the ideation of new business models. For that, the authors argue that most of the new business models are based on the recombination of existing business models. For that, they analyzed various organizations to extract patterns of them. As a process, they choose the four phases of initiation, ideation, integration, and implementation. Here, those patterns could be used in the ideation based on the definition of the existing business models, the actors, and the contextual factors in the initiation (see Section 2.3.2 for more about the business model patterns). They use a geometric-based language called the BMI Magic Triangle for the visualization of the business model.

The **BA 8: Business Model Patterns Database** [RHTK17] is a tool to support the design of business models based on recombination. For that, different approaches in the past have developed slightly different concepts for those patterns together with an overlapping set of patterns among those approaches. Here, the authors conducted a literature review on existing patterns for BMD and a taxonomy development to create a comprehensive business model pattern base. That pattern base is grouped into the five categories of overachieving, value proposition, value delivery, value creation, and value capture with the twelve dimensions of hierarchical impact, degree of digitization, product type, the strategy of differentiation, target customers, value-delivery process, sourcing, third parties involved, value-creation process, revenue model, pricing strategy, and direct profit effect. Each dimension provides different patterns with its name, description, selected examples, and sources. Moreover, they provide a simplified process to use those patterns inside the phases of initiation, ideation, integration, and implementation.

The **BA 9: Simulating Business Models using System Dynamics** [RVS15] is an approach to simulate the dynamic behavior of business models by combining the BMC and

system dynamics. For that, the authors enrich the metamodel of the canvas with flows and converters. For the flows, they use means to define necessary assets, the value that is proposed for the services & products, and the cash that is generated. For the converter, they allow the transformation of flows between different means, different values, and means and values. The enriched model, in turn, is transformed into a system dynamics diagram. Moreover, they propose a process with the phase of configuring the lab, designing experiments, and assessing results together with getting a business idea, designing a canvas model, defining scenarios, configuring parameters, simulating, observing results, and evaluating & redesigning. To allow the simulation, they use an external software tool called iThink.

The **BA 10: Dynamic Business Modelling Framework** [CN18] is an approach that provides a dynamic view of the business model by combining a canvas visualization with system dynamics. This, in turn, allows a business model simulation with which the organization can experiment how the business model reacts to strategic and organizational changes. For the canvas visualization, they use the seven building blocks of key partners, strategic resources, value proposition & key performance indicators, key processes, customer segments, cost structure, and revenue streams. They use strategic resources, flow variables, and strategic levers from system dynamics for simulation. Those dynamic concepts, in turn, can be combined into causal loops and initiated with concrete parameters to calculate different business model outcomes over time. This calculation, in turn, can be done with traditional software tools for system dynamics.

3.2.2 Requirement Comparison

To analyze the existing BA, we compare those tools against our HRs, as shown in Table 3.2. Here, we need to mention that most of those approaches are not designed primarily for using the support of an IT artifact. Nevertheless, the knowledge of those approaches can mostly be integrated into our approach. For the analysis, we divide that comparison into our three stages of knowledge provision of method and models, composition and enactment of development methods, and support of development steps.

The **(1) Knowledge Provision of Methods and Models** is related to the requirements of knowledge utilization, method comprehensiveness, model visualization, and tool support. For *HR 1: Knowledge Utilization*, some approaches propose conceptual models without knowledge (e.g., BA 2), while other approaches provide knowledge about the development methods (e.g., BA 6) as well as knowledge for the modeling artifacts (e.g., BA 8). For the *HR 2: Method Comprehensiveness*, some approaches support a dedicated type of modeling (e.g., BA 9), some approaches support specific phases of the development (e.g., BA 7), and some approaches provide support for all phases of BMD (e.g., BA 4). For *HR 3: Model*





































































































Tool	HR: 1	HR: 2	HR: 3	HR: 4	HR: 5	HR: 6	HR: 7	HR: 8	HR: 9	HR: 10
BA: 1										
BA: 2										
BA: 3										
BA: 4										
BA: 5										
BA: 6										
BA: 7										
BA: 8										
BA: 9										
BA: 10										
Legend	No Fulfillment (○)		Partial Fulfillment (◐)		Complete Fulfillment (●)					

Table 3.2 Comparison of the Business Approaches (BA) against the High-level Requirements (HR)

Visualisation, a few approaches use no visualization for their modeling artifacts (e.g., BA 2), some approaches use single visualizations of parts of the business model (e.g., BA 7), and some use different (canvas) models (e.g., BA 4). For *HR 10: Tool Support*, most approaches provide just conceptual models without concrete software (e.g., BA 2), while some approaches are based on already existing software (e.g., BA 9). However, based on that analysis, no approach formalizes both the methods consisting of development steps and models consisting of different canvas visualizations together with an accessible tool that can be downloaded and modified.

The **(2) Composition and Enactment of Development Methods** is related to the requirements of context awareness, selection assistance, and development continuity. For *HR 4: Context Awareness*, all approaches provide some implicit definition of the context for the development methods (e.g., BA 6) or the modeling artifacts (e.g., BA 7). For *HR 5: Selection Assistance*, some approaches provide no knowledge or no assistance in selecting the knowledge (e.g., BA 2), while others provide that selection for the development methods (e.g., BA 6) or the modeling artifacts (e.g., BA 8). For *HR 6: Development Continuity*, some approaches provide just single steps of the development (e.g., BA 8), some approaches

provide the continuous development of the business model (e.g., BA 9), and some approaches provide the implicit change of the development method (e.g., BA 1). However, based on that analysis, no approaches provide a flexible configuration of methods and models based on existing knowledge that can be easily modified during runtime.

The **(3) Support of Development Steps** is related to the requirements of stakeholder collaboration, artifact management, and decision support. For *HR 7: Stakeholder Collaboration*, some approaches are designed just for single stakeholders (e.g., BA 9), some approaches cover the needed collaboration of involved stakeholders (e.g., BA 7), and some approaches focus on the direct interaction of multiple stakeholders (e.g., BA 1). For *HR 8: Artifact Management*, half of the approaches provide no explicit artifacts (e.g., BA 2), while the other use manual management of the defined artifacts (e.g., BA 3). For *HR 9: Decision Support*, half of the approaches focus not on providing decision support (e.g., BA 2), while other approaches provide checklists (e.g., BA 2) or simple calculations (e.g., BA 9). However, based on that analysis, no approach provides collaborative development of different artifacts with modularized decision-support that can be flexibly applied to all steps.

3.3 Tools for Business Model Development

The development can be supported by various BMDTs that are developed in research and practice. In the following, we describe related tools (3.3.1) and compare our HRs against their functionalities (3.3.2).

3.3.1 Related Approaches

The BMD can be supported with different BMDTs. Here, those tools cover different aspects like the web-based visualization of business models developed by companies in practice, conceptual frameworks to develop software for business model management, or design support that can be used in the web browser developed by universities in research. In the following, we describe business tools (BT) related to our research selected from our SLR in Appendix A.

The **BT 1: Strategyzer Innovation Software**³ is a software tool for the collaborative development of business models. The tool consists of different canvas models (e.g., VPC, BMC), where the building blocks are enriched with additional explanations. On those canvas models, the stakeholders can collaborate by using sticky notes. Those sticky notes, in turn, can be directly connected to the hypothesis that needs to be validated, experiments that need

³Online Version of the Strategyzer Innovation Software: <https://platform.strategyzer.com/>

to be conducted, evidence that is gained, or insights that are given. Moreover, out of the sticky notes and guided parametrization, the profit of the business model can be estimated. Out of the business model, a process for testing can be derived. For that, a Kanban board is used to structure the current state of the experiments that must be conducted.

The **BT 2: Canvanizer**⁴ is a visualization tool for collaborative ideation on business models. For that, the tool consists of various defined canvas models (e.g., BMC) but other visual geometric-based models (e.g., empty 3x3-matrix) where the stakeholders can collaborate by placing sticky notes on the canvas. Inside the tool, features support the general working on the canvas (e.g., filter functions) and creativity mechanism (e.g., timer). Moreover, the tool has a focus mode where single building blocks are highlighted and explained. After the ideation of the business models, the tool allows various export functionalities, including a PDF export of the canvas or a read-only view for sharing the canvas over the internet.

BT 3: OctoProz [VOPN13, VPO⁺13] is a creativity support system for the process-oriented development of business models. For that, it can be used within the two contexts of developing a new business model or analyzing and refining an existing one. Here, the tool provides an initial configuration of the business model, which can be further refined due to the collaboration of different stakeholders. As a business model structure, they use a visual model around the two components of the process (connected to finance, resources, and values) and the central functions (connected to fixed costs and resources). For the evaluation of the business models, the tool provides decision support by a syntax check on the created model, comments and ratings by the stakeholders, and simple financial assessments.

The **BT 4: Business Model Decision Support System** [DHOB13] is sheet-based tool support for the development of software-as-a-service business models. For that, the tool's focus is the modularity of the different system components with the overall goal to estimate the consequences of different business model decisions. Here, each sheet corresponds to a single component which can be grouped into a configuration module, a market analysis module, a business model design module, and a decision analysis module. Inside those components, it is possible to define different evaluation criteria like design issues or success factors, which are used to run different analyses like conjoint analysis in the market analysis and multi-criteria analysis in the decision analysis.

The **BT 5: Business Model Developer** [BM14, BM17] is a software tool focusing on developing domain-specific business models. The tool uses a customizable BMC as a metamodel. Based on that metamodel, domain-specific knowledge can be defined as a shared vocabulary consisting of elements and their connections. During the BMD, the tool provides a configurator to develop a business model out of that knowledge and checks the syntax and

⁴Online Version of Canvanizer: <https://canvanizer.com/>

semantics of the model against the metamodel and the vocabulary. Based on the structures, it also allows the comparison of different business models. Moreover, based on the structured model, the tool allows the conduction of various analyses like cluster analysis or financial calculations.

BT 6: Virtual Business Model Innovation [EBL16] is a framework to support the development of software tools that manage and design business models. For that, they identified the four phases of environment analysis, business model design, business model implementation, and business model management, which are connected to knowledge in the form of shared material (e.g., guidelines to develop business models) and stakeholders within a community (e.g., project team). For the environmental analysis, they suggested using an existing repository of industry benchmarks and market analysis as knowledge together with characterizing the context of the industry, the market, and the customer needs. For the business model design, they suggested using visual models for collaboration. For the business model implementation, they suggested using the feedback of external users and domain experts. For the business model management, the continuity is gathered by the continuous feedback of users and refinement of domain experts.

The **BT 7: Framework for Analysis of Business Model Management** [TSLE17] is a generic process to cover all phases of business model management. For that, they divide the management into the four phases of analysis, design, implementation, and control, which all should be supported by software tools based on a conceptualization, and, therefore, visualization of the business model. For the analysis, they mentioned collecting relevant information for the environment, the customer, and the competitor in a structured way for its use in the design phase. For the design, they suggest an iterative design of the business model centered around the customer, together with the continuous validation of assumptions. For the implementation, they pointed out to find the proper organizational structure according to the stakeholders that are needed for the implementation. For the control, they suggest continuity in the improvement of the business model. For the software support, they gathered the results that the tools should be situation-specific and iterative using pre-structured processes.

The **BT 8: Green Business Model Editor** [SBK18a, SBK18b, KBS19] is a software prototype for enabling the reflection of sustainability during BMD. For that, the tool uses a customizable version of the BMC that is extended with building blocks to sustainability aspects. On the canvas, it allows the collaboration of different stakeholders together with their communication. Here, it is possible to collaborate on different views of the business model and share own developed business models. Moreover, the elements of the business model could be pre-selected from an existing library. During the whole development, the tool

supports the documentation and reasoning of changes. Moreover, to support the decision-making, the software tool allows the conduction of trade-off analyses.

The **BT 9: Envision Platform** [dRAH⁺16], which was migrated to BusinessMakeOver⁵, is a platform that collects various (non-software) tools that support small and mid-size enterprises in the development of new and innovation of existing business models. For that, the platform contains a guided process structured by the defined goals (e.g., implement a new business idea), from which a guided process containing different process steps is derived. Each step is linked to a tooling page (e.g., Business Model Canvas), where the tool is described in detail, together with a link to download the tool or online resources. With the support of those tools, different artifacts of the process can be created mostly manually.

The **BT 10: Smart Business Model Developer** [LFCJ⁺18], which was migrated to Venturely⁶, is an online tool to support the development of business models using business patterns. For that, the tool uses the BMC and a large repository of different so-called pattern packs (e.g., circular economy pack) for different application domains (e.g., recycling) that can be used as design support. Based on that, the tool provides basic decision support for financial calculations, the detection and comparison against business models of existing similar companies, and the testing of hypotheses. Moreover, it has the functionality of a guided process from the ideation of a business to its actual implementation. However, in the different steps, they provide just simple PDF files with empty visual models and additional explanations (e.g., trend matrix) that can be uploaded manually as artifacts to the process steps.

3.3.2 Requirement Comparison

To analyze the existing BT, we compare those tools against our HRs, as shown in Table 3.3. For that, we divide that comparison into our three stages of knowledge provision of method and models, composition and enactment of development methods, and support of development steps.

The **(1) Knowledge Provision of Methods and Models** is related to the requirements of knowledge utilization, method comprehensiveness, model visualization, and tool support. For *HR 1: Knowledge Utilization*, some tools provide just a simple representation of the business model without existing knowledge (e.g., *BT 2*), some tools use semi-formal method knowledge about different method steps (e.g., *BT 9*), some tools use formal model knowledge about elements and patterns (e.g., *BT 5*), and some tools combine a pre-structured method with existing modeling knowledge (e.g., *BT 2*). For the *HR 2: Method Comprehensiveness*,

⁵Online Version of BusinessMakeOver: <https://businessmakeover.eu/>

⁶Online Version of Venturely: <https://app.venturely.io/>





































































































Tool	HR: 1	HR: 2	HR: 3	HR: 4	HR: 5	HR: 6	HR: 7	HR: 8	HR: 9	HR: 10
BT: 1										
BT: 2										
BT: 3										
BT: 4										
BT: 5										
BT: 6										
BT: 7										
BT: 8										
BT: 9										
BT: 10										
Legend	No Fulfillment (○)			Partial Fulfillment (◐)			Complete Fulfillment (●)			

Table 3.3 Comparison of the Business Tools (BT) against the High-level Requirements (HR)

some tools cover just the single design step (e.g., *BT 2*), some tools cover multiple steps of development (e.g., *BT 5*), and some tools reflect all steps of the BMD (e.g., *BT 9*). For *HR 3: Model Visualisation*, a single tool uses no visualization (e.g., *BT 4*), some tools use fixed single visualization models (e.g., *BT 3*), and the majority of tools uses multiple or customizable canvas models (e.g., *BT 5*). For *HR 10: Tool Support*, some tools provide just a framework without concrete software (e.g., *BT 6*), some tools show tools but do not make them accessible (e.g., *BT 8*), and some tools make them also accessible (e.g., *BT 2*). However, based on that analysis, no tool formalizes the methods consisting of development steps and models consisting of different canvas visualizations together with an accessible tool from which the source code can be downloaded and modified.

The **(2) Composition and Enactment of Development Methods** is related to the requirements of context awareness, selection assistance, and development continuity. For *HR 4: Context Awareness*, some tools provide no flexible awareness according to the context (e.g., *BT 2*), some tools use general factors that need to be considered (e.g., *BT 6*), some tools cover choosing parts of the method knowledge based on goals as factors (e.g., *BT 9*), and some tools cover the choice of parts of the model knowledge based on an application

domain (e.g., *BT 10*). For *HR 5: Selection Assistance*, some tools provide no knowledge or no assistance in selecting the knowledge (e.g., *BT 2*), some tools provide a selection of the methods (e.g., *BT 9*), some tools provide a configuration on the models (e.g., *BT 3*), and some tools combine a fixed method with the configuration of the business model (e.g., *BT 10*). For *HR 6: Development Continuity*, some tools provide an one-time process (e.g., *BT 4*), some tools provide continuous changes in the business model (e.g., *BT 2*), and some tools provide a continuous evolvement of the business model (e.g., *BT 1*). However, based on that analysis, no tools provide a flexible configuration of methods and models based on existing knowledge that can be easily modified during runtime.

The **(3) Support of Development Steps** is related to the requirements of stakeholder collaboration, artifact management, and decision support. For *HR 7: Stakeholder Collaboration*, some tools are designed just for single stakeholders (e.g., *BT 4*), while other tools focus on the collaboration mechanisms between different stakeholders (e.g., *BT 1*). For *HR 8: Artifact Management*, two tools provide just manual management of artifacts (e.g., *BT 4*), while the other tools provide full management of changes of those artifacts (e.g., *BT 9*). For *HR 9: Decision Support*, one tool provides no decision support (e.g., *BT 2*), most tools have decision support for single steps (e.g., *BT 8*), and a few tools have a certain focus on decision support of different steps (e.g., *BT 5*). However, based on that analysis, no tool provides collaborative development of different artifacts with modularized decision-support that can be applied to all steps.

3.4 Summary

In this chapter, we have provided an overview of different approaches related to our research question. For that, we have categorized those approaches into business aspects in situational method engineering, situational aspects in business model development, and tools for business model development.

For the **Business Aspects of Situational Method Engineering**, we saw that most approaches just partly cover the business model. Here, two approaches (i.e., *SA 1*, *SA 2*) provide general frameworks that could be applied to our approach by defining DSLs. One approach (i.e., *SA 3*) provides a holistic approach but just covers the method repository and doesn't provide a modular solution for all stages. However, those approaches have certain constraints that reduce the simplification and applicability of our approach. Moreover, other approaches provide just conceptual models with predefined knowledge for the method repository (i.e., *SA 4*, *SA 5*, *SA 6*, *SA 8*, *SA 9*, *SA 10*) or focus just on the single aspect of generating business plans (i.e., *SA 7*). However, no approach could be directly used to

support the composition and enactment of development methods from predefined knowledge of development methods and modeling artifacts.

For the **Situational Aspects of Business Model Development**, we saw that most approaches have no IT artifact as support. Here, comprehensive books (i.e., *BA 1*, *BA 4*) provide a lot of textual information about the development method and the modeling artifacts. Other conceptual models focus on the high-level conduction of experiments (i.e., *BA 2*, *BA 3*) or the provision of a non-linear process with different phases (i.e., *BA 5*). Moreover, repositories provide textual information for the different experiments of the development method (i.e., *BA 6*) or patterns for the modeling artifacts (i.e., *BA 7*, *BA 8*). Last, two approaches simulate different parts of the business model based on standard techniques (i.e., *BA 9*, *BA 10*). However, no approach supports the composition and enactment of development methods from predefined knowledge of development methods and modeling artifacts.

For the **Tools for Business Model Development**, we saw that most approaches focus on visualizing parts of the business model or fixed development phases. Here, practical tools focus on the pure visualization of the business model (i.e., *BT 2*) together with their evaluation based on experiments (i.e., *BT 1*). Other tools focus on the different phases (i.e., *BT 4*) of the development and visualize parts as modeling artifacts (i.e., *BT 3*, *BT 5*). Some tools provide high-level descriptions of frameworks to innovate the business model (i.e., *BT 6*, *BT 7*), combine the BMC with pattern support (i.e., *BT 8*, *BT 10*), or provide guided processes for certain business goals (i.e., *BT 9*). However, no tool supports the composition and enactment of development methods from predefined knowledge of development methods and modeling artifacts.

Based on those related works, the next chapter proposes our solution for fulfilling all HRs. For that, we map those generic HRs to specific solution requirements together with showing an overview of the provided stages and involved stakeholders. Moreover, we present the application of our approach to SEs.

Part II

Solution Concept

Chapter 4

Conceptual Overview

In the previous chapter, we analyzed related approaches to our RQ that don't cover all of our required HRs. Based on that, in this chapter, we introduce the conceptual overview of our solution to fulfill those HRs. For that, we first give an overview of our proposed solution (4.1). After that, we present our application area on SEs (4.2). Finally, we summarize our concept and the application area (4.3).

4.1 Overview of the Solution

Within this thesis, we propose the approach of situation-specific BMD within the application area of SEs. The underlying requirements of our solution are derived from the HRs based on a literature review and tool analysis on BMD. Out of those requirements, we design our solution by refining the concept of SME for BMD under the usage of formalization mechanisms from ME.

In this section, we first define the solution requirements that are the basis of our approach (4.1.1) by analyzing the derived HRs. Based on that, we show an overview of our solution with the corresponding stages based on those requirements (4.1.2). Last, we explain the involved roles in our approach (4.1.3).

4.1.1 Overview of Requirements

To design our approach for situation-specific BMD, we conduct a literature review of BMD and a tool analysis of BMDSSs, as described in Section 1.3, to derive the HRs for such a solution. Out of them, we create the nine solution requirements (SRs) as explained in the following paragraphs. A mapping of the HRs to our SRs is shown in Figure 4.1.

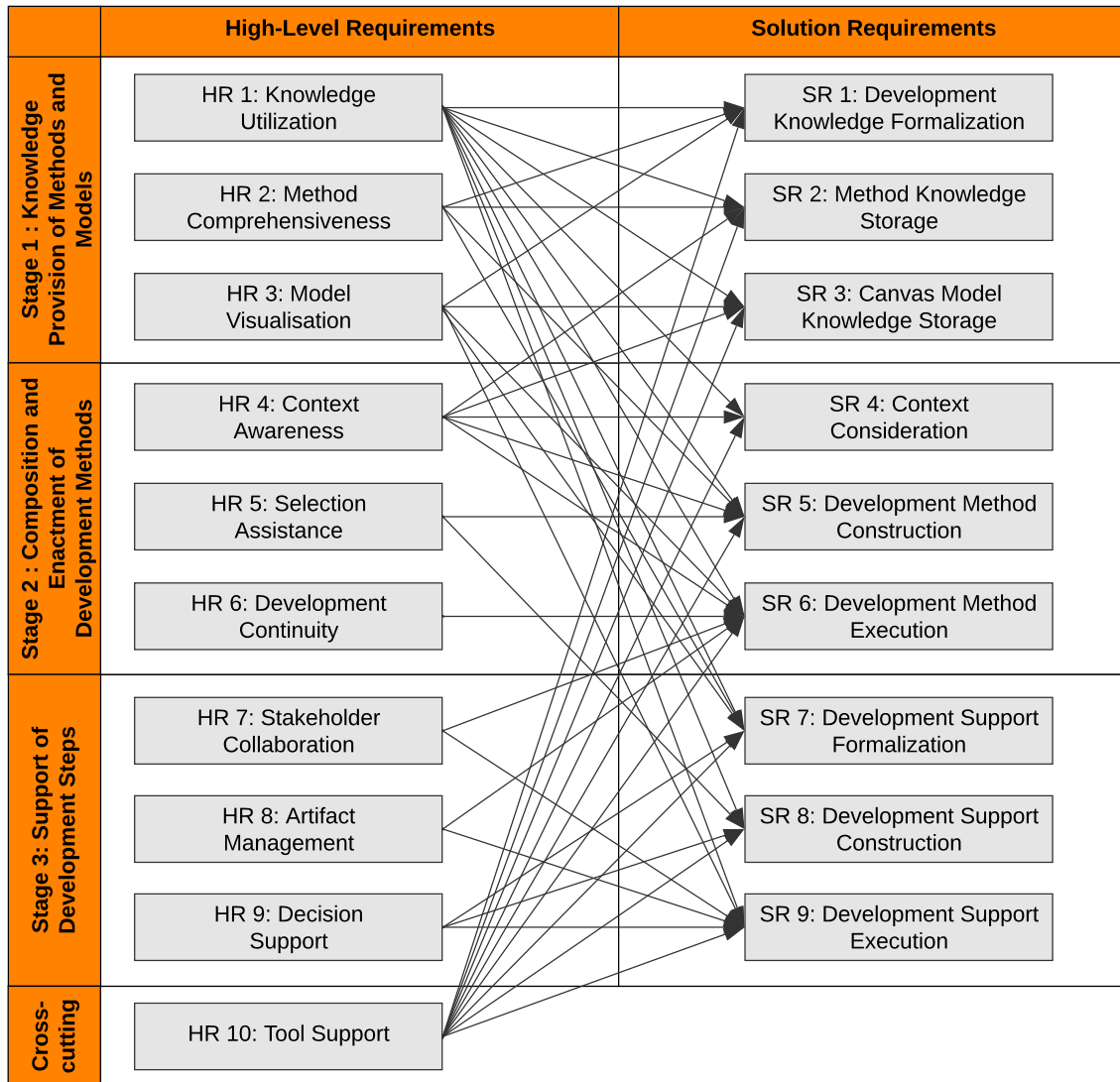


Fig. 4.1 Mapping of High-Level Requirements to Solution Requirements

The development of business models is a complex activity that can be supported with knowledge about tasks to be accomplished inside different proposed methods and decisions to be made inside different proposed models. However, that knowledge in terms of methods and models needs to be utilized to make it accessible to the business developer (i.e., *HR 1: Knowledge Provision*). Those methods that are proposed by different domain experts can support different phases of the BMD (i.e., *HR 2: Method Comprehensiveness*). Here, those methods can be split into atomic parts to reuse them in different existing methods or combine them into new ones. Those models that are proposed by different domain experts can support the visualization of decisions during the BMD (i.e., *HR 3: Model Visualization*). Here, those models can be split into atomic parts to reuse them in different models. Therefore,

our solution should provide a formalization mechanism for the knowledge of methods and models.

- **SR 1: Development Knowledge Formalization:** The solution should provide a formalization mechanism for the atomic parts of the methods and the models from different sources.

The methods, which provide the tasks that the business developer should conduct, are mostly developed by different domain experts for general purposes or specific types of organizations. Here, the method knowledge should be provided in an accessible form (i.e., *HR 1: Knowledge Utilization*) for the business developer and support all phases of the BMD (i.e., *HR 2: Method Comprehensiveness*). After formalizing the atomic parts of those methods, the solution should store those parts in an accessible form to structure the BMD. Parts can be related to situation-specific aspects of the organization for which the business model is developed (i.e., *HR 4: Context Awareness*).

- **SR 2: Method Knowledge Storage:** The solution should provide a storage mechanism for the situation-specific atomic parts for all phases of business model development methods.

The canvas models, which provide the possible decisions of the organization that the business developer should consider, are mostly developed by different domain experts for general purposes or specific types of services. Here, the model knowledge should be provided in an accessible form (i.e., *HR 1: Knowledge Utilization*) for the business developer and visualized by canvas models that are often used (i.e., *HR 2: Model Visualization*). After formalizing the atomic parts of those models, the solution should store them in an accessible form to support the BMD. Parts can be related to domain-specific aspects of the service for which the business model is developed (i.e., *HR 4: Context Awareness*).

- **SR 3: Canvas Model Knowledge Storage:** The solution should provide a storage mechanism for the domain-specific atomic parts of the developed modeling artifacts based on canvas models.

The development of the business model takes place in a highly uncertain environment where different assumptions have to be validated over time. That uncertainty also includes the context in terms of the organization's situation and the service's application domain, which can change over time (i.e., *HR 4: Context Awareness*). Therefore, the solution should provide a mechanism to continuously update the context in which the business model is developed. That context can be gathered from the utilized knowledge of the domain experts (i.e., *HR 1: Knowledge Utilization*).

- **SR 4: Context Consideration:** The solution should provide an adaptation mechanism for the consideration of the changeable situation of the organization and the application domain of the service.

Instead of using a fixed development method of a single domain expert, it is also possible to construct a customized development method. Here, the development method can be constructed from the atomic parts of the methods, and specific development steps can be supported by the atomic parts of the canvas models (i.e., *HR 1: Knowledge Utilization*). Due to the high amount of possible development knowledge, the solution should provide assistance in constructing a development method (i.e., *HR 5: Selection Assistance*) that covers all phases of the BMD (i.e., *HR 2: Method Comprehensiveness*). That assistance, in turn, can be based on the situation of the organization to select the suitable method parts and the application domain of the service to select the suitable canvas model parts (i.e., *HR 4: Context Awareness*).

- **SR 5: Development Method Construction:** The solution should provide an assembly mechanism for the context-specific construction of the development method based on the selection of method parts and canvas model parts.

To support the usage of such a fixed or constructed development method, it should be executable within a software tool. Here, our solution should provide execution of the steps of the development method guided by the existing knowledge (i.e., *HR 1: Knowledge Utilization*). During the execution, the business developer and other involved stakeholders might collaborate (i.e., *HR 7: Stakeholder Collaboration*) to create different (canvas) artifacts (i.e., *HR 8: Artifact Management*). Those artifacts might partly be based on the visual representations of the models (i.e., *HR 3: Model Visualization*). Moreover, the BMD is a continuous activity (i.e., *HR 5: Development Continuity*) which includes that tasks and decisions need to be adapted due to a changing context (i.e., *HR 4: Context Awareness*).

- **SR 6: Development Method Execution:** The solution should provide an execution mechanism for the continuous usage of the development method to collaboratively develop (canvas) artifacts during the development steps.

Various software tools have been developed in the past to assist the different phases or steps of the development method. That support has a wide range from the visualization of the business model, over the design of parts of the business model, to decision-making during the development steps (i.e., *HR 9: Decision Support*) that different domain experts developed with the support of software developers. Here, the knowledge about that support can be

split up into atomic parts to provide flexible usage during the execution of the development method (i.e., *HR 1: Knowledge Utilization*). Therefore, our solution should provide a formalization mechanism for the knowledge of development support for the supported development steps (i.e., *HR 2: Method Comprehensiveness*) and visualized models (i.e., *HR 3: Model Visualization*).

- **SR 7: Development Support Formalization:** The solution should provide a formalization mechanism for the atomic parts of the development support from different sources.

The development support, which assists the business developer and the other stakeholders during the BMD, can be gathered for different phases or steps of development (i.e., *HR 9: Decision Support*). Here, that knowledge about the different types of support can be utilized to have a common understanding between the business developer and all other stakeholders (i.e., *HR 1: Knowledge Utilization*). After formalizing the atomic parts of that development, the solution should provide an assistant to construct the development support for specific development steps (i.e., *HR 5: Selection Assistance*).

- **SR 8: Development Support Construction:** The solution should provide a construction mechanism for the atomic parts of the development support for specific development steps.

The development support needs to be applied to the different phases or development steps of the BMD. Here, those different support possibilities of visualization, design, and decision (i.e., *HR 9: Decision Support*) can be guided by the existing knowledge (i.e., *HR 1: Knowledge Utilization*). Here, our solution should provide an application of the development support in the development steps so that the business developer and the stakeholders might collaborate (i.e., *HR 7: Stakeholder Collaboration*) to create different artifacts (i.e., *HR 8: Artifact Management*). Those artifacts can partly be based on the visual representations of the models (i.e., *HR 3: Model Visualization*).

- **SR 9: Development Support Execution:** The solution should provide an execution mechanism for applying the development support to develop (canvas) artifacts collaboratively during the development steps.

4.1.2 Overview of Stages

Based on our derived SRs, we present an overview of our situation-specific BMD approach, as shown in Figure 4.2. For that, we refine the concept of Assembly-based SME (cf. Section

2.2.2), which contains the composition and enactment of the development method. Here, we separate the creation of the method base from the composition of the method to allow the flexible usage of different knowledge sources. Moreover, we enhance the enactment by providing flexible development support for the development steps. This, in turn, leads to the three stages of (1) *Knowledge Provision of Methods and Models*, (2) *Composition and Enactment of Development Methods*, and (3) *Support of Development Steps*.

Inside those stages, we have various people with different roles that work together. Here, we have the already introduced domain experts that provide the knowledge of methods and models, the business developer of the organization who wants to develop a business model for his service, and other stakeholders (e.g., software developer, early adopter) who are involved in different development steps. Moreover, from SME, we use the method engineer who formalizes the knowledge and composes the development methods and the meta-method engineer (i.e., us in this thesis) who provides the metamodels for the knowledge. For providing the assistants in the development steps, we identify the development support engineer who provides the specific development support. Last, a meta-development support engineer is needed (i.e., us in this thesis) who enables the development support within the software tool. Therefore, we need to consider the seven roles of the *Meta-Method Engineer*, the *Domain Expert*, the *Method Engineer*, the *Business Developer*, other *Stakeholders*, the *Meta-Development Support Engineer*, and the *Development Support Engineer*. In the following, we present our overview of those three stages based on the used foundations.

The **(1) Knowledge Provision of Methods and Models** is used to utilize the development knowledge about methods to use and models to rely on within the BMD. Here, the *Meta-Method Engineer* (i.e., us in this thesis) creates meta models for the methods and canvas models (1.1). For that, we develop DSLs with syntax, semantics, and visual notation to structure the *Method Repository* and the *Canvas Model Repository* (cf. Section 2.1.2 / i.e., *SR 1: Development Knowledge Formalization*). On the one hand, the *Method Repository* stores method parts for all phases of BMD, including situation-specific aspects (cf. Section 2.3.2 / i.e., *SR 2: Method Knowledge Storage*). On the other hand, the *Canvas Model Repository* stores model parts for visualizing free-definable canvas models with their elements, including domain-specific aspects (cf. Section 2.3.1 / i.e., *SR 2: Canvas Model Knowledge Storage*). Next, different *Domain Experts* explain their domain knowledge in terms of existing BMDMs and possible BMs (cf. Section 2.3) to the *Method-Engineer* (1.2). The *Method Engineer*, in turn, formalizes the development knowledge of method and model fragments according to the meta-models to make them usable within the repositories (1.3).

The **(2) Composition and Enactment of Development Methods** constructs the *Development Method* and executes it as *Development Process*. Here, we apply the two steps of

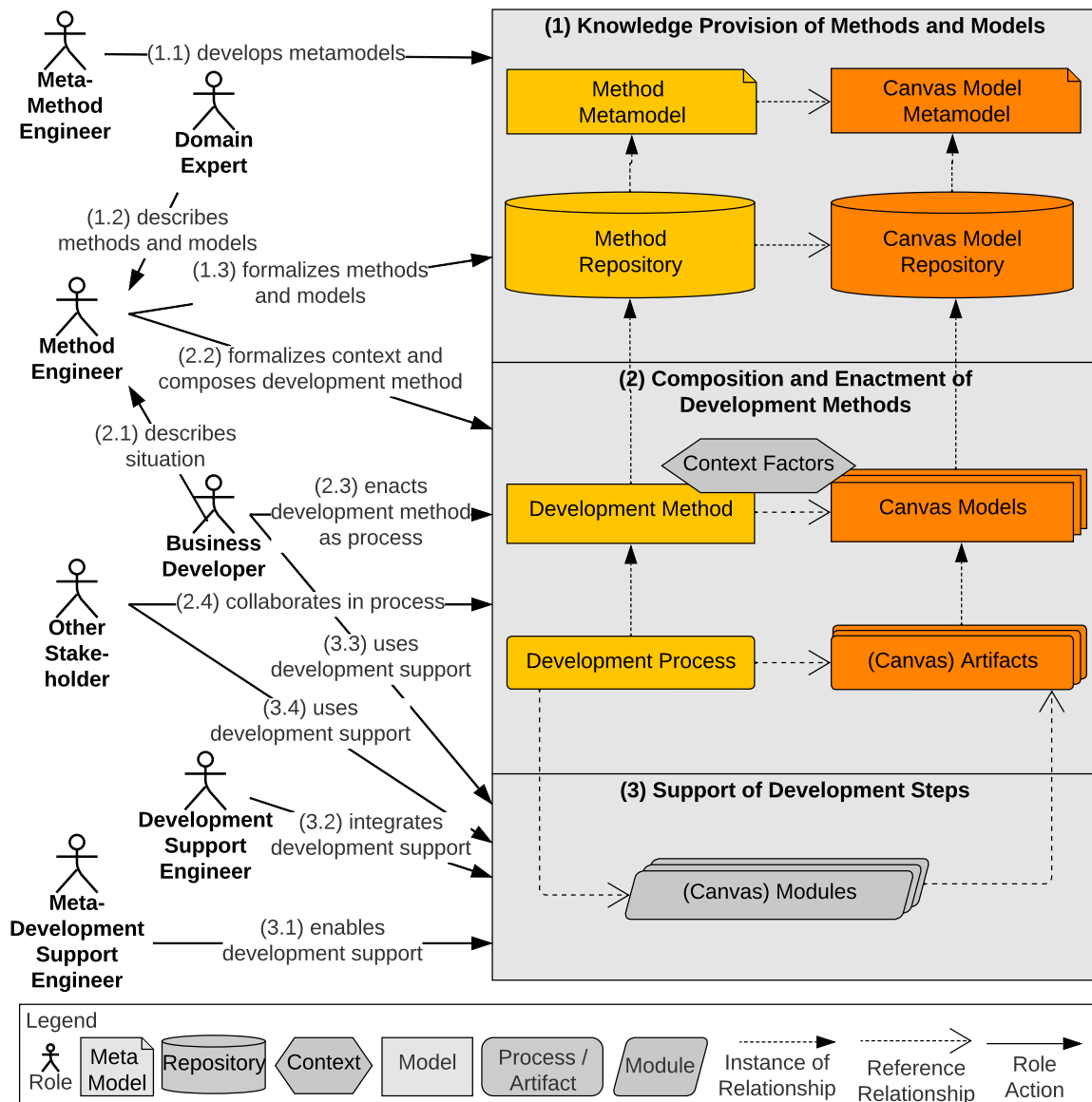


Fig. 4.2 Overview of the Approach with Roles and Stages

Assembly-based SME, where the development method is composed by the *Method Engineer*, but the enactment is done by the *Business Developer* instead of the project manager (cf. Section 2.2.2). In the beginning, the *Business Developer* explains the current context in which the business model should be developed to the *Method Engineer* (2.1). The *Method Engineer* formalizes these *Context Factors* as the situation of the *Development Method* and application domain of the *Canvas Models* (i.e., SR 4: *Context Consideration*) together with composing the *Development Method* (2.1). For that, the *Method Engineer* constructs the *Development Method* out of the *Method Repository* and connects specific development steps with canvas

artifacts to the *Canvas Models* in the *Canvas Model Repository* (i.e., *SR 5: Development Method Construction*). Based on that, the *Business Developer* executes the constructed *Development Method* as a *Development Process* and uses the connected *Canvas Models* as *Canvas Artifacts* (2.3) (cf. Section 2.3.2 / i.e., *SR 6: Development Method Execution*). Here, the *Business Developer* conducts the single development steps and modifies the *Artifacts*, including *Canvas Artifacts* (cf. Section 2.3.1). Moreover, other *Stakeholders* might contribute to different development steps and modify (*Canvas*) *Artifacts* during the execution of the steps (2.4). An exemplary development process is shown in Figure 4.3. Here, the *Business Developer* interacts with the other *Stakeholders* (i.e., *Customer*, *Software Developer*) to develop a business model. For that, he enacts the development process and conducts several development steps. For example, the development step of the *Interview Customer* is used to create a *Customer Information* artifact and the *Create Business Model* is used to create a *Business Model Canvas* artifact.

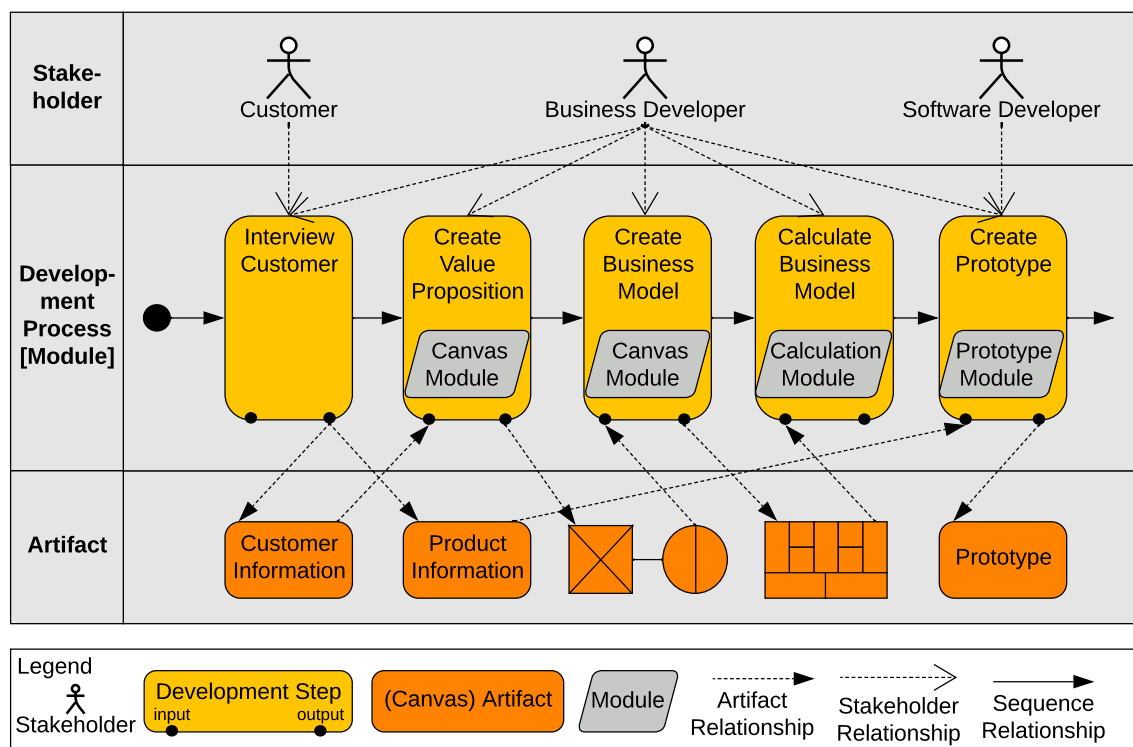


Fig. 4.3 Exemplary Execution of the Development Process

The **(3) Support of Development Steps** is used to provide assistance for the development steps with the flexible usage of (*Canvas*) *Tools*. Here, we support the visualization, the design, and the decision during the steps (cf. Section 2.3.3), in which underlying knowledge needs to be formalized for a common understanding (i.e., *SR 7: Development Support*

Formalization). The *Meta-Development Support Engineer* (i.e., us in this thesis) enables the development support by providing interfaces and hooks in the software tool for specific support modules for such assistants (3.1). Based on that, the *Development Support Engineers* add their development support to specific development steps in the *Method Repository* (3.2). During the execution of the composed development method, the *Business Developer* uses those (*Canvas*) *Modules* to create and modify the (*Canvas*) *Artifacts* (3.3). During that, he might collaborate with the *Other Stakeholders* (3.4). Exemplary software support for different development steps is shown in Figure 4.3. Here, the *Canvas Module* is used to create the *Value Proposition Canvas* artifact and the *Business Model Canvas* artifact, the *Calculation Module* is used to create a calculation of the business outcome, and the *Prototype Module* is used to create a prototype of the mobile app.

4.1.3 Overview of Roles

For our situation-specific BMD approach, we have identified seven different roles. Those are the meta-method engineer, the domain expert, the method engineer, the business developer, other stakeholders, the meta-development support engineer, and the development support engineer.

The **Meta-Method Engineer** is responsible for identifying important knowledge parts in existing resources for BMD, abstracting their syntax, semantic mapping, and visual notation, and formalizing them through metamodels with graphical support where the knowledge can be represented through. That knowledge, in turn, might come from various sources like existing development processes (e.g., [STRV10]), method repositories (e.g., [BO20]), modeling languages (e.g., [Ost04]), or pattern repositories (e.g., [GFC14]). In this thesis, we take the responsibility of the meta-method engineer by analyzing different sources of development methods and canvas models to develop the metamodels for the method repository and the canvas model repository.

The **Domain Expert** is responsible for providing the knowledge to support the development of the business model in terms of development methods and canvas models. Here, those knowledge is used to make the ontological semantics of the method repository and the canvas model repository accessible. By referring to Lethbridge et al. [LSS05] that knowledge can be divided into first, second, and third degrees. In the first degree, the domain expert stands in direct contact with the method engineer to share the knowledge. In the second degree, the domain expert shares the knowledge directly within the tool. In the third degree, the domain expert refers to already shared knowledge in the past. In this thesis, we are using knowledge of third degree by conducting a Grey Literature Review (GLR) [GFM19] to fill the method

repository (see knowledge provision of Section 9.1) and a Taxonomy Development (TD) [NVM13] to fill the canvas model repository (see knowledge provision of Section 9.2).

The **Method Engineer** is responsible for storing the knowledge of methods and models within the repositories and composing the development method based on constantly changeable, described context. To access the knowledge from the domain experts, the method engineer can interview them (first degree), provide the tool for adding them (second degree), or analyze existing sources of them (third degree) [LSS05]. To compose the development method for the business developer, the method engineer can analyze the described context for the situational factors, the application domains, and additional construction guidelines [HSRÅR14]. Moreover, we add development support to specific development steps. Out of that, he composes the development method out of both repositories and adapts it due to context changes. In the optimal case, the method engineer and the business developer roles are filled with the same person to eliminate communication between both roles and reduce the overhead of adapting the development method for changing situations.

The **Business Developer** is responsible for enacting the composed development methods and developing the business models. Here, the business developer needs a deep understanding of the running organization and its different available resources [OP10]. For enacting the development method, the business developer explains the context of the existing organization and the planned service to the method engineer. The business developer executes the development method as a process and conducts the development steps to develop the business model. During the development steps, the business developer collaborates with other stakeholders to creatively create different artifacts [EHB11]. In contrast to the project manager who enacts the method for software development, we have the additional responsibility of the business developer to conduct the development steps.

The other **Stakeholders** are responsible for supporting the conduction of the different business model development steps. Here, stakeholders with different skills can interact during the phases, like working on new ideas, developing prototypes, or running marketing campaigns [AdR20]. Those stakeholders, in turn, can be divided into internal stakeholders of the organization (e.g., software developer, marketing manager) and external stakeholders outside the organization (e.g., investor, early adopter) [SL20]. Within our approach, those stakeholders can be flexibly defined in the method repository.

The **Meta-Development Support Engineer** is responsible for identifying needed interfaces for integrating existing development support into the solution of the software tool as support modules. He analyses different BMDTs and BMDSSs to abstract the needed interfaces and hooks together with implements and documents them. Those tools can be gathered from different literature reviews of software tools in research (see Appendix A for

BMDSSs) and practice (see [SSJ⁺19] for BMDTs). In this thesis, we take the responsibility of the meta-development support engineer by analyzing different decision support systems and developing a metamodel to integrate the development support.

The **Development Support Engineer** is responsible for developing the development support as support modules and integrating them into the software tool. For that, he uses the provided interface and hooks of the meta-development support engineer to create new development support from scratch or to integrate existing development support. Those support might have various goals like visualization, design or decision support [SSJ⁺19, BdRHF20]. In this thesis, we take the responsibility of the development support engineer by providing the first support modules for the development support in the phases of design and validation support.

4.2 Application to Software Ecosystems

This thesis uses SEs as the application area of our approach. Here, we, in particular, focus on mobile ecosystems as one of the most competitive but also interesting ecosystems for third-party developers. This focus, in turn, ensures a direct transfer of our thesis results to a high number of potential users.

In this section, we first introduce the application area of SEs and reason our decision for mobile ecosystems (4.2.1). Based on that, we show an exemplary execution of the stages of our solution for the scenario of a mobile todo app developer (4.2.2).

4.2.1 Introduction of Software Ecosystems

We choose SEs, particularly mobile ecosystems, as our application area in our thesis. As already stated, we use the definition of Bosch et al., who define SEs as "a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs" [BBS10]. Here, the ecosystem providers of the software platforms provide SDKs to allow external developers to extend their own platforms. With this, those providers are participating in the value-creating transactions between the external developers (i.e., service providers) and the users. Here, one special group of SEs are store-oriented SEs, where the developers place their extensions within a store. Here, the providers often take transaction fees for all transactions made with their stores.

Based on that opportunity, more and more organizations build ecosystems around their existing (software) products to participate in the value-creating transactions. These are, for

example, Sony with their PlayStation Store¹, Adobe with their App Marketplace², or SAP with their SAP Store³. Based on an analysis of existing SEs, a recent study by Jazayeri et al. developed the three architectural patterns of resale software ecosystems, partner-based ecosystems, and OSS-based ecosystems [JZE⁺18]. Here, resale ecosystems provide a large number of extensions by different independent external developers. After their creation, the extensions were sold to many users within the SE. Next, partner-based ecosystems are used for complex SE in new sectors, where the external developers and ecosystem providers build new extensions based on partnership agreements. Here, different openness policies support providers in protecting the intellectual property within their SE. Last, in OSS-based ecosystems, the software platform is released under open source by the ecosystem provider. Here, the external developers are mostly not financially motivated to develop extensions but aim to gain reputations or extend the ecosystem for their own purposes. Based on those patterns, resale software ecosystems are our solution's most interesting type of ecosystem. This is reasoned by the fact that those external developers in the SEs are self-responsible for selling their extensions against many competitive extensions. Moreover, due to the size of those ecosystems, a lot of knowledge about successful business models is available, which can be utilized by our approach. Here, mobile ecosystems are a subset of those SEs.

Mobile ecosystems that are provided by Apple around iOS⁴ and Google around Android⁵, are one of the most evolved ecosystems. While Apple bundled its operating system with its hardware, Google licensed its system to various hardware manufacturers. Here, those two ecosystems provide external developers with access to billions of users [App21]. However, those developers also compete with their applications against millions of other applications of hundreds of thousands of other developers. Moreover, both mobile ecosystems have an ongoing growth trend, and the ecosystems also extend to different other application areas like the television or car segment with Apple CarPlay⁶ or Google TV⁷. In contrast, those ecosystems provide simple store concepts with limited visibility boost for new applications. Here, developers are self-responsible for making their applications successful. Therefore, those developers need an effective business model for their applications.

Here, as shown in Figure 4.4, both *Mobile Ecosystem Providers* provide an *Application Store* (i.e., AppStore, PlayStore) within their *Mobile Operating Systems*. Within those stores, they provide features like a *Search* or *Rating/Review* for the different existing *Applications*.

¹Website of the Sony's PlayStation Store: <https://store.playstation.com/>

²Website of Adobe's App Marketplace: <https://exchange.adobe.com/>

³Website of SAP Store: <https://store.sap.com/>

⁴Website of Apples iOS: <https://www.apple.com/ios>

⁵Website of Googles Android: <https://www.android.com/>

⁶Website of Apple CarPlay: <https://www.apple.com/ios/carplay/>

⁷Website of Google TV: <https://tv.google/>

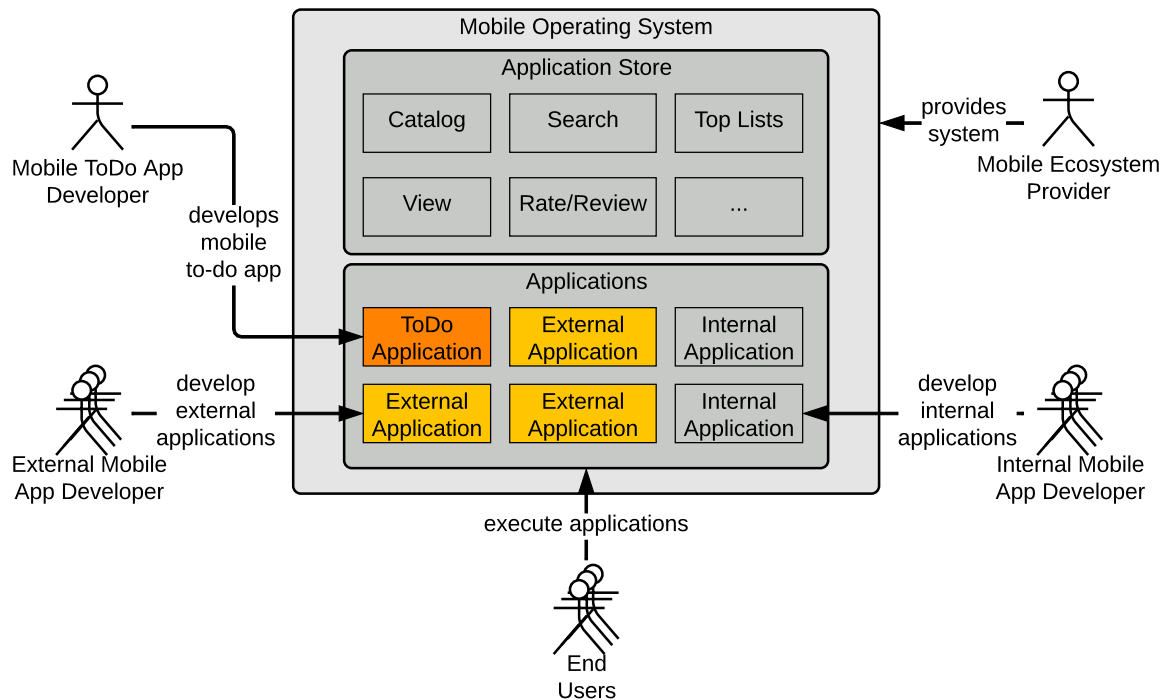


Fig. 4.4 Overview of the Mobile Ecosystem with the Mobile ToDo Application

Those applications are developed by different *Internal Mobile App Developers* and *External Mobile App Developers*. Moreover, they can be executed by the already existing *End-Users* of the ecosystem. Here, for example, a new *Mobile ToDo App Developer* publishes a new *ToDo Application* in the store. With this, he directly needs to provide advantages against similar *External Applications* of other *External Mobile App Developers*. In the following, we show how such a developer can use our approach to develop a business model for his application.

4.2.2 Usage of Solution

Our scenario for showing the usage of our solution is based on the feasibility study of our first design cycle presented in Section 1.2. In our scenario, the *Mobile ToDo App Developer* wants to develop a new *ToDo Application* and place it within the *Application Store*. By doing that, he has different options: First, he can try to target a large group of the *End-Users* by developing a generic todo app. With this, he might have the largest target group by also the competition of *Internal Applications* (e.g., Apple Reminders, Google Tasks) and *External Applications* (e.g., Microsoft Todo, Todoist). Second, he targets a smaller group by providing special features (e.g., time management, todo splits) and a corresponding value proposition to the *End-Users*. With this, he might have a smaller overall market but also less competition

against other *External Applications*. However, for both options, he needs to develop also a proper business model for his organization to create, deliver, and capture the possible value for the target group. Here, he uses our situation-specific BMD approach to develop the potential business model. The development method's focus should be the business model. For that, as presented in Figure 4.5, he conducts all three stages of our approach.

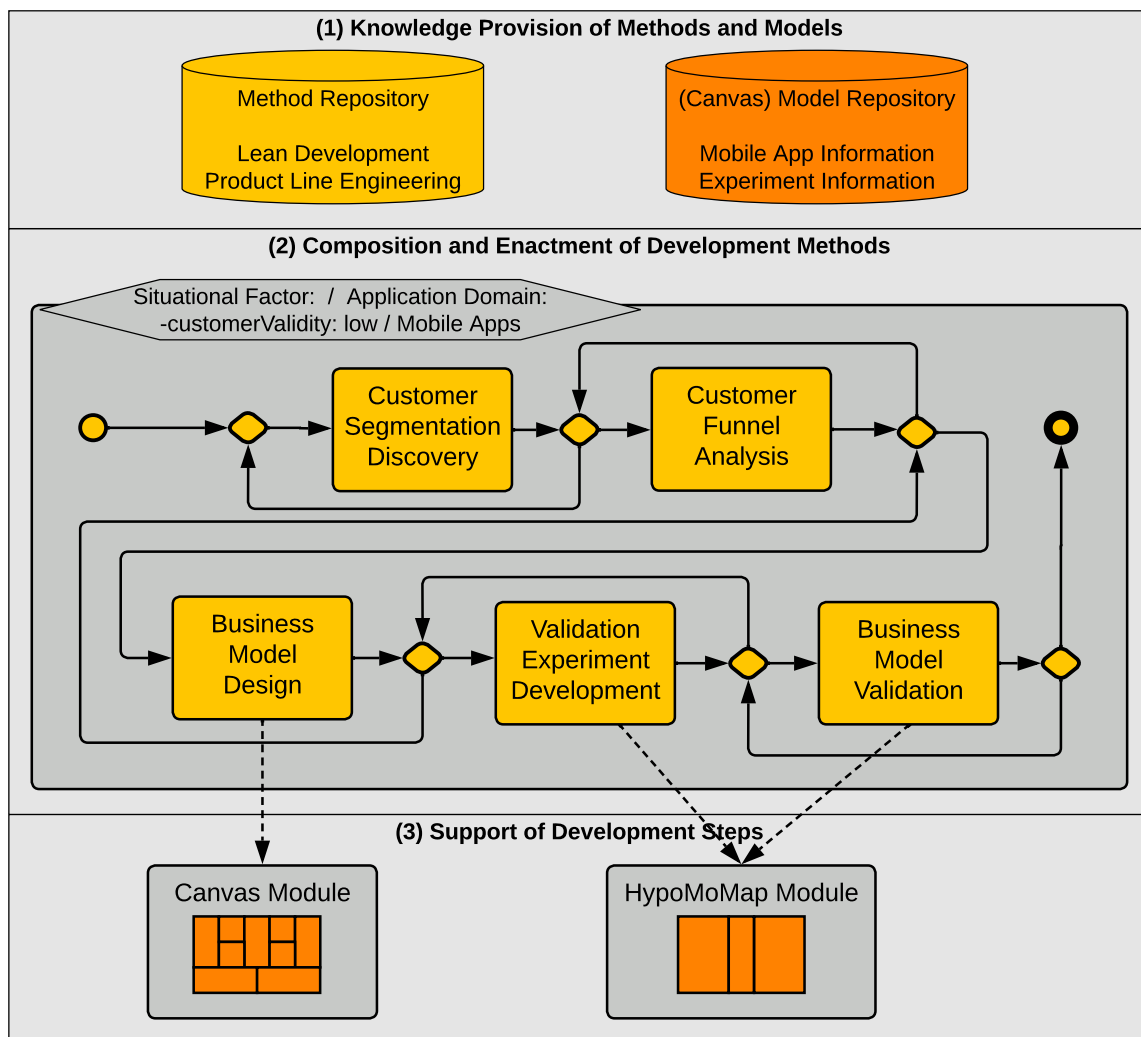


Fig. 4.5 Exemplary Stages for the Mobile App Developer

In the **(1) Knowledge Provision of Methods and Models**, the mobile app developer needs to gather the knowledge for the development. Here, the business model development is supported by the knowledge of development methods and modeling artifacts from different *Domain Experts*. In this scenario, we are used third-degree knowledge by utilizing already existing information. For the *Method Repository*, we analyzed the handbooks of Product Line Engineering [ABKS13] and LEAN Development [Rie14] to transfer those

development methods to BMD. Here, we extract different method fragments (e.g., the task to segment the customer groups and the stakeholders of customers) together with information on how to structure those fragments (e.g., customer segmentation before business model design). For the *(Canvas) Model Repository*, we analyzed existing mobile applications in the mobile ecosystems [GRE19b] and the handbook for different experiments for business model validation [BO20] to use that information for the modeling artifacts. Here, we extract different (canvas) model fragments (e.g., subscription as revenue streams) together with information on how to structure those fragments (e.g., refine subscription to monthly and yearly subscriptions). After this knowledge has been gathered, it can be reused for composing different BMDMs to develop BMs.

In the **(2) Composition and Enactment of Development Methods**, the mobile app developer has to construct and execute a development method. Here, at first, he needs to clarify the purpose and scope of his development. For that, he chooses different context factors (e.g., *customerValidity:low*, *Mobile Apps*) for the repositories together with additional constraints for the development method (e.g., focus on business model validation). Out of the method fragment and structure information, he constructs the iterative development method with stage gates of conducting a *Customer Segment Discovery* to identify different target groups, followed by a *Customer Funnel Analysis* to analyze the channels where those customers will come from. Next, he makes the *Business Model Design* by using the BMC. Out of that, he uses a *Validation Experiment Development* to identify experiments for testing the BM, which is used within the *Business Model Validation* by conducting those experiments. After that, he enacts the development method to conduct the different steps to create certain artifacts. For example, for the customer segments, he might identify the fitness improver, who will get special features for meal and gym times, life improver, who we will propose to save a work-life balance, or business improver, who will use the proposition to optimize workflows to get more done at the same time. Out of that, he derives the funnel that a broader range of customer segments can be reached with Facebook advertisements, while dedicated customer segments can be found in specialized online communities. After that, he uses the information to design different business models with elements like online communities for customer acquisition and calorie trackers as value offerings for fitness improvers, social media ads for customer acquisition and optimized schedules as value offerings for life improvers, and customer support as relationship and workflow tracking as value offerings for business improvers. In the end, he needs to evaluate those business models by using different experiments like Facebook advertisements to test different conversing rates of landing pages or smoke tests for different features.

In the **(3) Support of Development Steps**, the mobile app developer can gather recommendations for single development steps. Here, that development support needs to be connected to the fragments in the *Method Repository* before the composition of the development method. For the *Business Model Design*, he uses the *Canvas Module* to receive recommendations (e.g., possible revenue streams) for the design of the business model. Here those recommendations are visualized based on the BMC. This is done based on the *Mobile App Information* in the *(Canvas) Model Repository* and is explained in detail in Section 7.3.1. For the *Validation Experiment Development* and the *Business Model Validation*, he uses the *Hypothesis Modeling and Mapping (HypoMoMap) Module* to gather recommendations on which experiment he should conduct. Here, those recommendations give the developer suggestions on how to validate different assumptions about the business model with a single experiment. This is done based on the *Experiment Information* in the *(Canvas) Model Repository* and is explained in detail in Section 7.3.2. By using those modules, he receives additional support in developing an effective business model.

4.3 Summary

In this chapter, we have provided the conceptual solution for our situation-specific BMD approach. For that, we have given an overview of our solution with the different roles and stages together with software ecosystems as the application area.

For the **Solution Overview**, we have analyzed the extracted HRs and mapped them to the requirements of our solution. Based on that, we have developed an overview with seven roles and three stages. Successive, we describe the roles of the meta-method engineer, the domain expert, the method engineer, other stakeholders, the meta-development support engineer, and the development support engineer, together with the stages of knowledge provision of methods and models, composition and enactment of development methods, and support of development steps.

For the **Application Area**, we have chosen SEs and, in particular, mobile ecosystems. Here, we have introduced the application area together with our reasoning for mobile ecosystems based on their ongoing growth trend. For developers in those ecosystems, we showed the use of our approach's three stages to develop possible business models.

Based on that conceptual overview, in the following chapters, we will provide detailed explanations for each stage. Those stages are the knowledge provision of methods and models to fill the method and canvas model repository, the composition and enactment of development methods to construct and execute context-specific development processes, and the support of development steps to assist the development in specific steps.

Chapter 5

Knowledge Provision of Methods and Models

In the previous chapter, we gave a conceptual overview of our overall solution. Based on that, this chapter shows the first stage of knowledge provision of methods and models of our solution concept. For that, we first refine our SRs and give an overview of the stage (5.1). Based on that, we describe the provision of the method repository (5.2) and the canvas model repository (5.3). Finally, we summarize our procedure within the stage (5.4).

5.1 Requirements and Overview

The knowledge provision of methods and models is the first stage of our approach, which aims to store the formalized knowledge that is later reused within the BMD. For that, we refine the SRs, which were derived in Section 4.1.1, of *SR 1: Development Knowledge Formalization*, *SR 2: Method Knowledge Storage*, and *SR 3: Canvas Model Knowledge Storage* into detailed Knowledge Provision Requirements (KPR) together with providing an overview of both knowledge repositories.

The *Method Repository* is used to store the knowledge from various existing BMDMs in a unified way to make them accessible within the composition of a new method. Here, the repository needs to formalize the necessary method knowledge on the stakeholders, phases, tasks, artifacts, and tools, including the organization's situation, together with information on how to construct the method out of the repository. To increase the acceptance by the users, the knowledge should be expressed by a visualization together with an understandable and extensible method repository. Therefore, our KPRs are:

- **KPR 1: Situation-specific Provision of Method Knowledge:** The solution should provide the storing of situation-specific method knowledge, including the possible situational factors, the supported phases, the involved stakeholders, the accomplished tasks, the developed artifacts, and the used tools.
- **KPR 2: Structuring of Method Knowledge:** The solution should provide a structuring of the method knowledge to allow the flexible composition of comprehensive methods.
- **KPR 3: Visual Representation of Method Knowledge:** The solution should provide a visual representation of the method knowledge to support the business developer's acceptance.
- **KPR 4: Understandability of Method Repository:** The solution should provide appropriate explanations to allow a unified understanding of the method knowledge among the business developer and the other stakeholders.
- **KPR 5: Extensibility of Method Repository:** The solution should provide the extensibility of the method repository for new knowledge on business model development methods, including changing situations.

The *Canvas Model Repository* is used to store knowledge from existing taxonomies and patterns in a unified way to make them accessible within the development of new canvas model artifacts. Here, the repository needs to formalize the necessary canvas model knowledge on the items to place on the canvas models, their relationships, and exemplary instances of patterns and organizations, including the application domain of the service with structures to compose the canvas models of the repository. To increase the acceptance by the users, the model knowledge should be expressed by the widely used visualizations of canvas models together with an understandable and extensible model repository. Therefore, our KPRs are:

- **KPR 6: Domain-specific Provision of Canvas Model Knowledge:** The solution should provide the storing of domain-specific canvas model knowledge, including the applied application domains, the used items, their possible relationships, and exemplary instances.
- **KPR 7: Hierarchical Structuring of Canvas Model Knowledge:** The solution should provide a hierarchical structuring of canvas model knowledge to provide a refinement of the knowledge within the canvas models.

- **KPR 8: Visual Representation of Canvas Model Knowledge:** The solution should provide visual representations in the form of canvas models to support the acceptance of the business developer and the involved stakeholders.
- **KPR 9: Understandability of Canvas Model Repository** The solution should provide appropriate explanations to allow a unified understanding of the canvas model knowledge among the business developer and the other stakeholders.
- **KPR 10: Extensibility of Canvas Model Repository:** The solution should provide the extensibility of the canvas model repository for new knowledge on taxonomies and patterns, including new canvas representations.

Out of the KPRs of the *Method Repository* and the *Canvas Model Repository*, we develop an overview of the first stage, as shown in Figure 5.1. The provision of the *Method Repository* is explained in Section 5.2. Here, the repository consists of the method fragments of different atomic *Method Elements* (e.g., *Developer* for *Stakeholder*) that combined *Method Building Blocks* (e.g., *Customer Interview*) and arranged through the optional *Method Patterns* (e.g., *Init Development*). The provision of the *Canvas Model Repository* is explained in Section 5.3. Here, the repository consists of the model fragments of different atomic *Canvas Elements* (e.g., *Save Privacy* for *Item*) that are structured through *Canvas Building Blocks* (e.g., *Business Model*) and visually represented through *Canvas Models* (e.g., *Business Model Canvas*). In the following, we present the provision of both repositories.

5.2 Provision of Method Repository

The *Method Repository* contains the reusable knowledge needed to compose the development method for the BMD. For that, the repository allows the utilization and storage of knowledge that could be gathered from single development methods (e.g., [McG10, SEP⁺19]) or repositories of methods (e.g., [BO20, SP09]). The abstracted phase for providing the *Method Repository* can be seen in Figure 5.2. Here, in the beginning, the *Meta-Method Engineer* needs to specify the *Method Metamodel* once for the repository. Based on that, different *Domain Experts* explain their *Method Knowledge* to the *Method Engineer*. The *Method Engineer*, in turn, models the *Method Repository* with the different method fragments of the *Method Elements*, the *Method Building Blocks*, and the optional *Method Patterns*. The output of this step is the *Filled Method Repository* with expert knowledge.

In this section, we show the utilization of the method fragments together with our created metamodels. For that, we first explain the atomic parts of the method in the form of *Method*

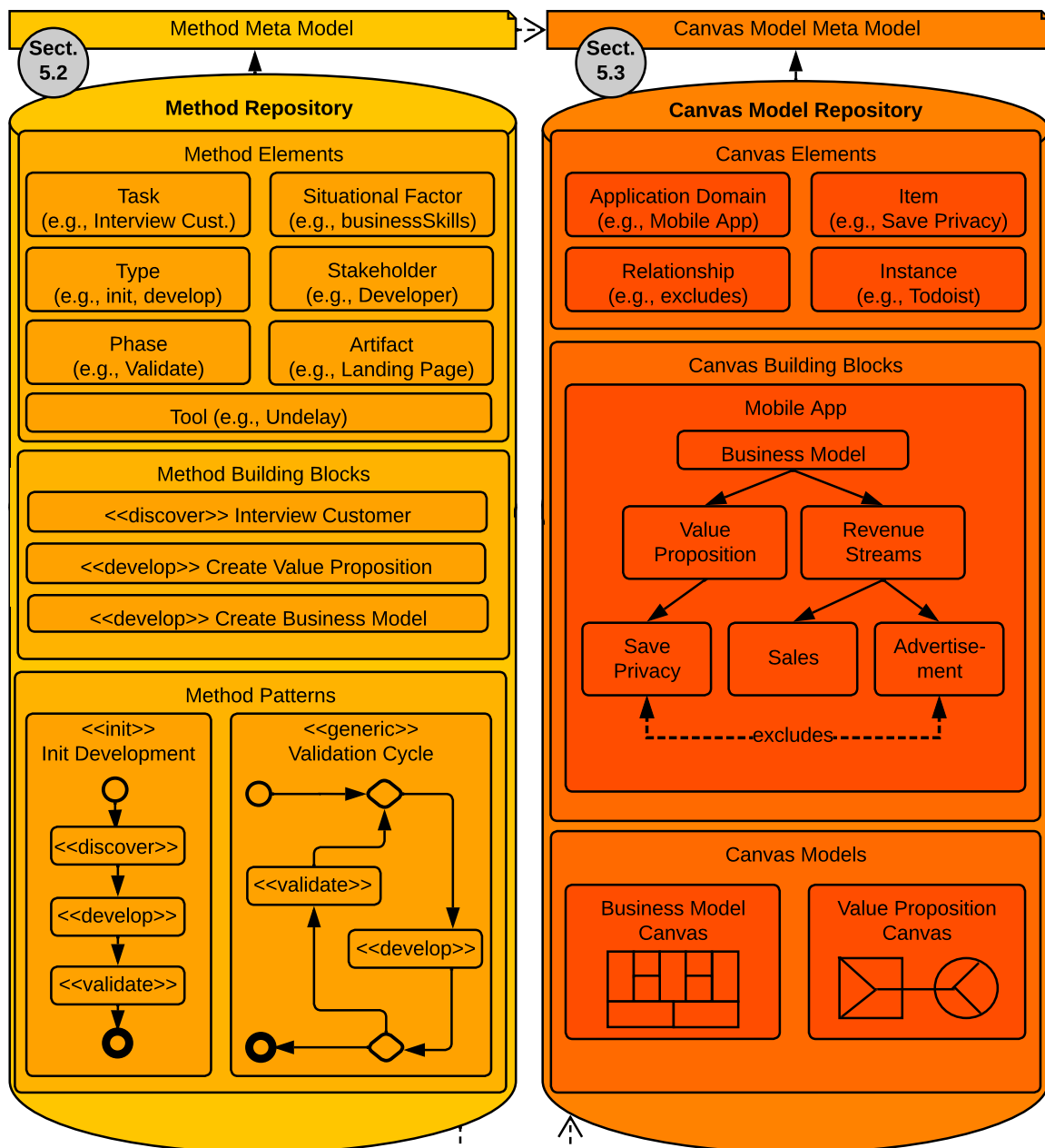


Fig. 5.1 Overview of the Knowledge Provision of Methods and Models

Elements (5.2.1). Based on that, we show the combination of different elements to *Method Building Blocks* (5.2.2). Last, we present the optional arrangement of those building blocks by using *Method Patterns* (5.2.3).

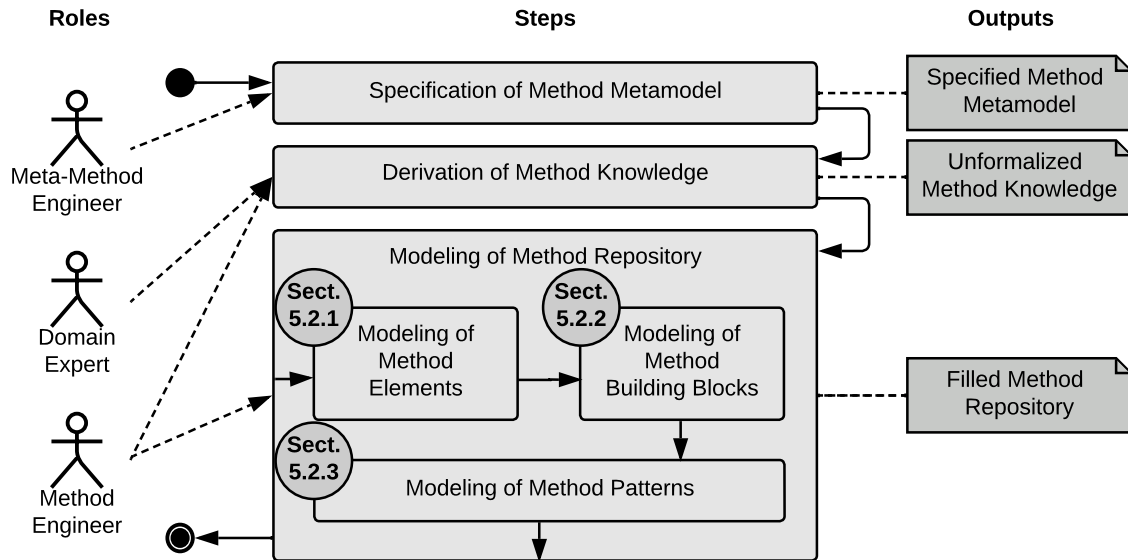


Fig. 5.2 Abstracted Phase of Providing the Method Repository

5.2.1 Modeling of Method Elements

The *Method Elements*, which metamodel is shown in Figure 5.3, are atomic parts of the development methods that have a name and a description. Moreover, those elements are divided into tasks, phases, types, stakeholders, artifacts, tools, and situational factors.

The **Tasks** are the main activities that need to be performed during the BMD. Here, various tasks on different granularity levels (e.g., *Interview Customers*, *Create Business Model*) can be defined. Moreover, those tasks are refined through an ordered list of *TaskSteps* (e.g., *Create Questionnaire*, *Organize Interviews*) that provides guidance during the conduction of the tasks.

The **Phases** are used to provide a phase-based composition of the development method. For that, those phases should be comprehensive to support the whole process of BMD (e.g., *Design* or *Validate* the business model). Moreover, those phases are ordered to structure the different development steps of the BMD (e.g., *Design* before *Development*). Section 6.2.2 explains the phase-based composition of the development methods.

The **Types** are used to create a pattern-based composition of the development method. For that, they are used as guidance in the optional method patterns, see Section 5.2.3, to connect them to building blocks or other patterns (e.g., *discover*, *develop*). Moreover, some *Types* provide logic to those patterns in the composition stage. Here, patterns with the *InitialisationType* (e.g., *initialisation*) serve as a starting point during the composition, and building blocks and patterns with the *GenericType* (e.g., *generic*) can be used in every pattern

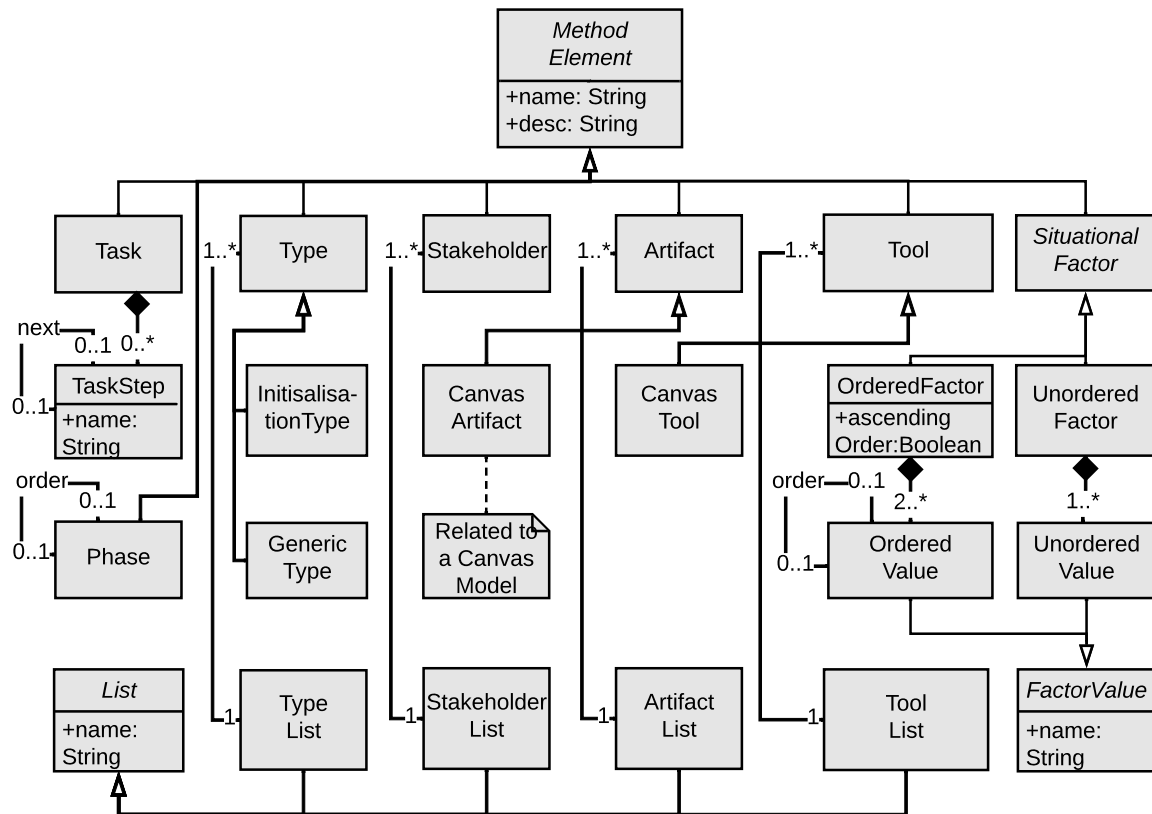


Fig. 5.3 Metamodel of the Method Elements

regardless of the required type. Moreover, types are grouped within a *TypeList* (e.g., *Design* with *featureDesign* and *businessDesign*). Section 6.2.2 explains the type-based composition of the development methods.

The **Stakeholders** are producer (cf. Section 2.2.1) who are involved in the different activities of the BMD. For that, it is possible to define external and internal stakeholders (e.g., *Business Developer*, *Development Agency*). Moreover, those stakeholders are grouped within a *StakeholderList* (e.g., *Company* with *Business Developer* and *Software Developer*).

The **Artifacts** are work products that are created and modified during the BMD. For that, it is possible to define internal or link external artifacts (e.g., *Customer Information*, *Product Prototype*). Moreover, it is possible to create special *CanvasArtifacts* (e.g., *Value Proposition Canvas*, *Business Model Canvas*) that are connected to the *Canvas Models* of the *Canvas Model Repository*. The representation of the canvas models is shown in Section 5.3.3. Last, all artifacts are grouped into an *ArtifactList* (e.g., *Medium* with *Website* and *Landing Page*).

The **Tools** are work units to support the performing of activities during the BMD. For that, different external tools (e.g., *AppAnnie*, *Figma*) are used during different phases to develop and modify artifacts. Moreover, a *CanvasTool* is used to directly create and modify

the canvas artifacts. The creation and modification of canvas artifacts is explained in Section 6.3.2. Last, all tools are grouped into *ToolLists* (e.g., *Prototyping Tools* with *Adobe XD* and *Figma*).

The **SituationalFactors** are used to classify in which situation a building block or a pattern should be applied. Here, those factors are divided into ones with ordinal scales and ones with nominal values. *OrderedFactors* use ordinal scales with ascending or descending to *OrderedValues* (e.g., *low < medium < high* for *developmentSkills*). *UnorderedFactors* use nominal values as *UnorderedValues* for division (e.g., *mass* or *niche* for *marketSize*).

Out of those atomic *Method Elements*, we combine *Method Building Blocks* as compact representations of development steps for the development method.

5.2.2 Modeling of Method Building Blocks

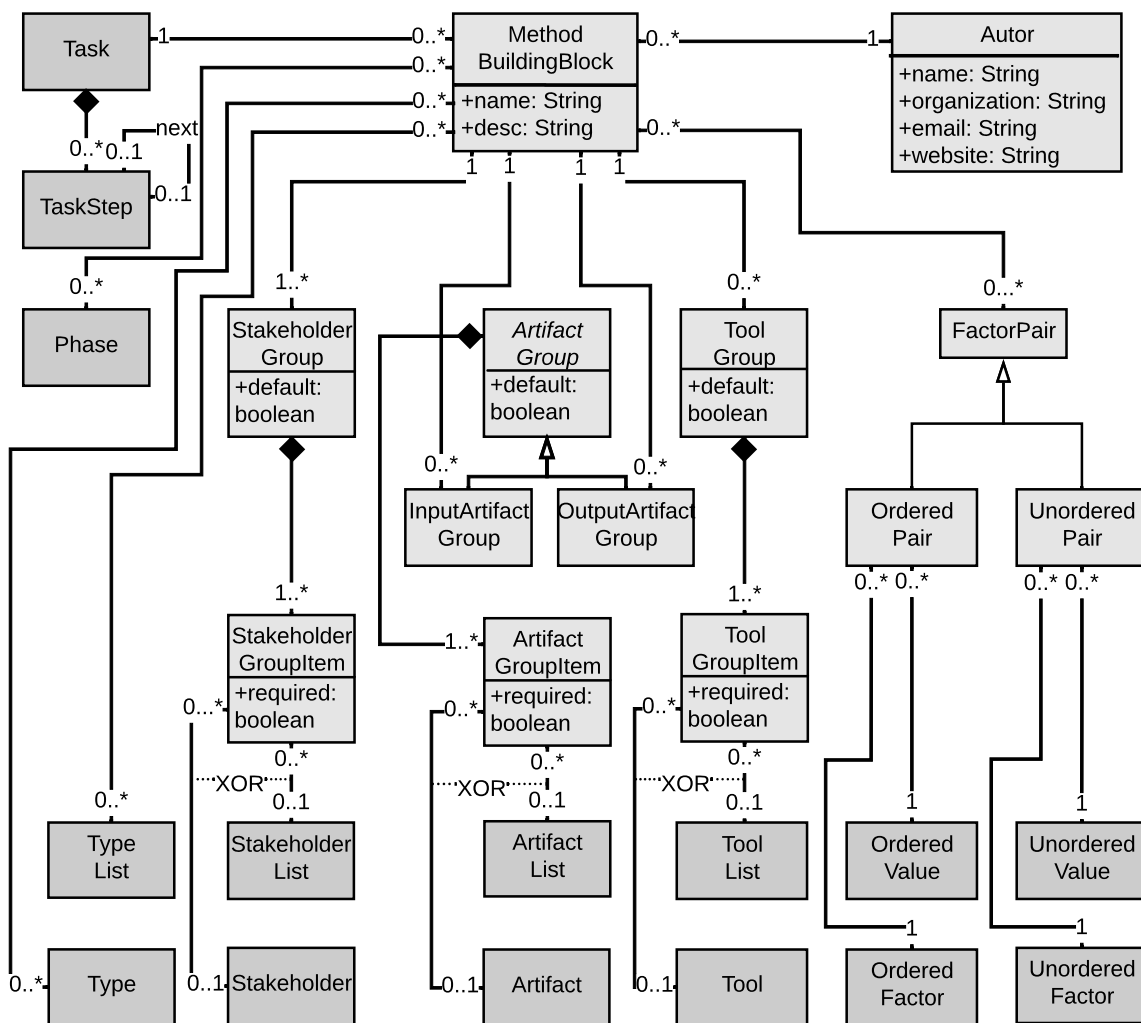
The *Method Building Blocks*, which metamodel is shown in Figure 5.4, are the development steps for the development methods that have a name and a description. Each building block corresponding to an author contains a single task, multiple lists of types or types, multiple phases, multiple groups of stakeholders, artifacts and tools, and multiple factor pairs.

For the single **Task**, each building block is connected to the activity that should be accomplished. Here, those tasks (e.g., *Interview Customer*) are refined into the manual task steps (e.g., *Create Questionnaire*) or the canvas steps (e.g., *Create Canvas*). Moreover, each building block is connected to an **Author** with information about a name, an organization, an email, and a website.

For the multiple **Phases**, each building block is connected to the phases where the development steps can be used (e.g., *Design*). With this, we support the selection of building blocks during the phase-based composition of the development method.

For the multiple **Types** and **TypeLists**, each building block is connected to the types used for the selection of building blocks during the development method's type-based composition. For that, the connection can be used with a single type (e.g., *businessDesign*) or a list of types (e.g., *design*).

For the multiple **StakeholderGroups**, **ArtifactGroups**, and **ToolGroups**, each building block is connected to the involved stakeholders, the developed artifacts, and used tools. Here, the *ArtifactGroup* is refined into the *InputArtifacts* and the *OutputArtifacts*. Moreover, a single group from each is selected as the default group (i.e., *default*) that can be changed during the composition. Each group consists of multiple items (i.e., *StakeholderGroupItem*, *ArtifactGroupItem*, *ToolGroupItem*), which can be mandatory or optional (i.e., *required*) for the group. Moreover, those items are connected to a concrete element (i.e., *Stakeholder*, *Artifact*, *Tool*) or an abstract element list (i.e., *StakeholderList*, *ArtifactList*, *ToolList*). While



the concrete elements are fixed during the knowledge provision (e.g., *Early Adopter* as *Stakeholder*), the abstract lists (e.g., list of *Early Adopter*, *Potential Customer*, *Angel Investor*) allow the selection of a concrete element (e.g., *Angel Investor* as *Stakeholder*) during the composition of the development method. This, in turn, allows us to reduce the manual modeling effort in the method repository while keeping the method space flexible in a targeted manner.

For the multiple **FactorPairs**, each building block is connected to multiple situational factors and selected values. Those pairs can be an *OrderedPair* with an *OrderedFactor* and an *OrderedValue* (e.g., *developmentSkills:medium*) or an *UnorderedPair* with an *UnorderedValue* and an *UnorderedFactor* (e.g., *marketSize:mass*). Those factors, in turn, support guidance in selecting the proper development steps during the composition.

To structure those *Method Building Blocks*, we support arranging them using the optional combination of different *Method Patterns* for the pattern-based composition.

5.2.3 Modeling of Method Patterns

The optional *Method Patterns*, which metamodel is shown in Figure 5.5, are used for the pattern-based arrangements of the development steps of the development methods that have a name and a description. We based those patterns on the Business Process Model and Notation (BPMN), which is used to model the behavior of business processes by providing a graphical representation [Obj10]. To use the concept of BPMN, each pattern that corresponds to an author contains multiple lists of types or types, a part of a BPMN process, and multiple factor pairs.

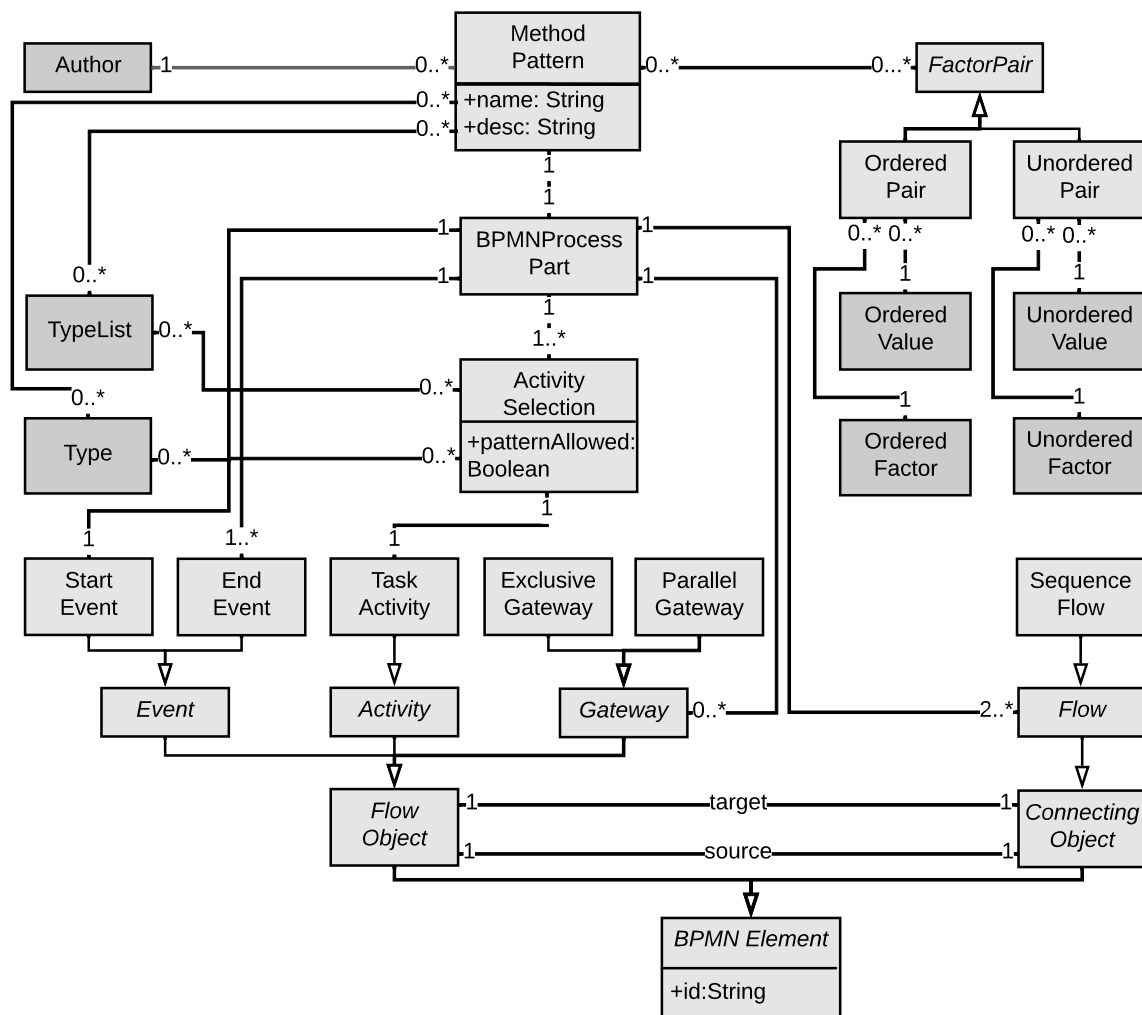


Fig. 5.5 Metamodel of the Method Patterns

For the multiple **Types** and **TypeLists**, each pattern, similar to the building blocks, is connected to the types used for the method repository's choices. For that, the connection can be used with a single type (e.g., *customerDiscovery*) or a list of types (e.g., *discovery*). Moreover, patterns with the *InitialisationType* (i.e., *initialisation*) are used as the start pattern of the pattern-based composition, and patterns with the *GenericType* can be inserted into activities of other patterns regardless of the type.

The **BPMNProcessParts**, see Figure 5.6 for the visual notation of BPMN, consist of the flow objects, connecting objects, swimlanes, and artifacts. *Flow Objects* can have the type of events, activities, and gateways. *Events* denote the happening by a certain trigger. Here, the *Start Event* is triggered at the beginning of the process, and the *End Event* is triggered at the ending of a process. *Activities* denote some work that needs to be accomplished. Here, *Task Activity* is an atomic activity that needs to be done. *Gateways* represent different options for the path depending on certain conditions. Here, the *Exclusive Gateways* follow a single path in the process, while the *Parallel Gateway* follows all paths in the process. The *Connecting Objects* are used to connect the different flow objects with each other. Here, the *Sequence Flow* shows an ordering of the connecting objects. While we summarize here the essential elements used within the thesis, the complete list of elements can be discovered in [Obj10]. Each pattern contains a subprocess using a subset of the *BPMN Elements*. Those elements can be divided into *FlowObjects* that are connected using *ConnectingObjects*. As connections, we use *SequenceFlows* to guide the process. As flows, we use a single *StartEvent*, multiple *EndEvents*, multiple *TaskActivities*, multiple *ExclusiveGateways*, and multiple *ParallelGateways*. While most of the flows are directly used in the process part, the activities are wrapped through an *ActivitySelection*. Here, the selection is connected to multiple *TypeLists* and *Types* in which corresponding building blocks could be inserted into the activity. Moreover, the activities could also allow the usage of other patterns (i.e., *patternAllowed*).

An example of pattern visualization is shown in Figure 5.6. Here, we have the *Init Development Pattern* with a sequence of *Method Building Blocks* or *Method Patterns* with the *Types* of *discover*, *develop*, and *validate*. Moreover, we have the *Validation Cycle Pattern* with the loop on the *Types* of *develop* and *validate*.

For the multiple **FactorPairs**, each pattern, similar to the building blocks, is connected to multiple situational factors and selected values. Those pairs can be an *OrderedPair* with an *OrderedFactor* and an *OrderedValue* (e.g., *businessModelingSkills:low*) or an *UnorderedPair* with an *UnorderedValue* and an *UnorderedFactor* (e.g., *marketType:b2c*). Those pairs are used to guide the pattern selection during the type-based composition.

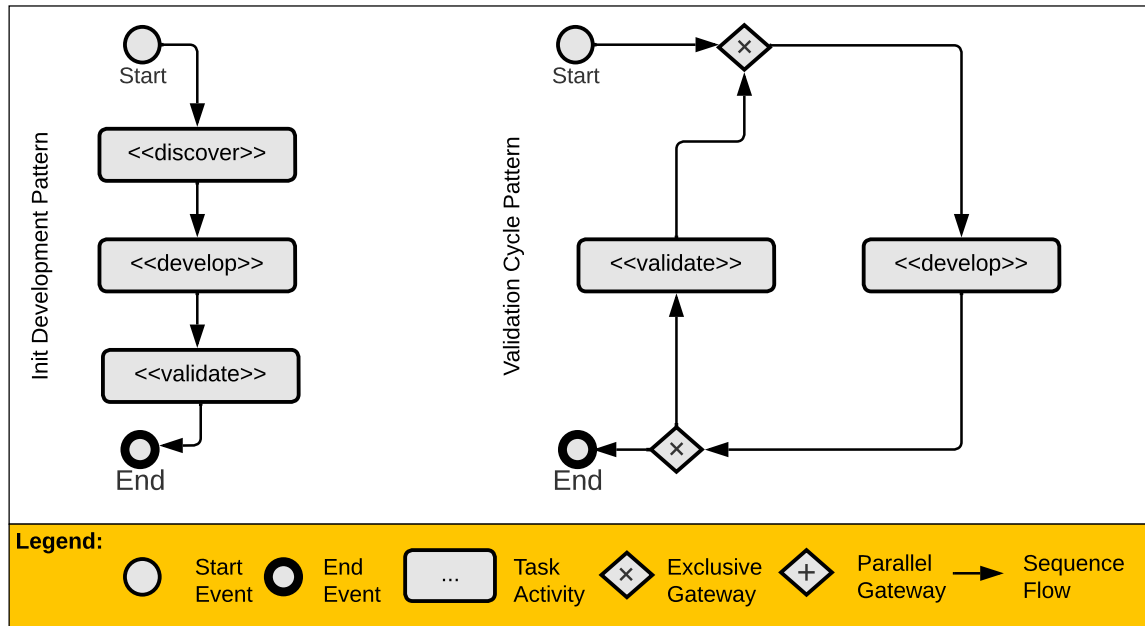


Fig. 5.6 Exemplary Visualisation of the Method Patterns

5.3 Provision of Canvas Model Repository

The *Canvas Model Repository* contains the knowledge needed to compose the canvas artifacts to support the BMD. For that, the repository allows the utilization and storage of knowledge that could be gathered from taxonomies of possible business models (e.g., [HZFN16, AIB07]) or patterns/examples of successful business models (e.g., [RHTK17, GFC14]). The abstracted phase for providing the *Model Repository* can be seen in Figure 5.7. Here, in the beginning, the *Meta-Method Engineer* needs to specify the *Canvas Model Metamodel* once for the repository. Based on that, different *Domain Experts* explain their *Model Knowledge* to the *Method Engineer*. The *Method Engineer*, in turn, models the *Canvas Model Repository* with the different method fragments of the *Canvas Elements*, the *Canvas Building Blocks*, and the *Canvas Models*. The output of this step is the *Filled Canvas Model Repository* with expert knowledge.

In this section, we show the utilization of the atomic parts of the model fragments with our created metamodels. For that, we first explain the atomic parts of the models in the form of *Canvas Elements* (5.3.1). Based on that, we show the hierarchical ordering of different elements to *Canvas Building Blocks* (5.3.2). Last, we represent those building blocks by using *Canvas Models* (5.3.3).

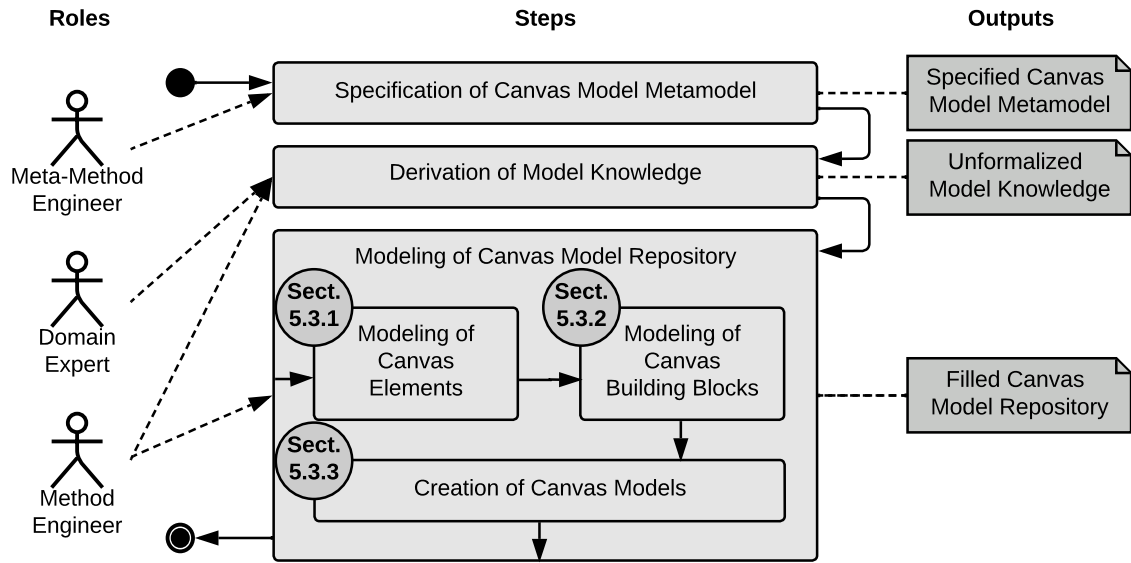


Fig. 5.7 Abstracted Phase of Providing the Canvas Model Repository

5.3.1 Modeling of Canvas Elements

The *Canvas Elements*, which metamodel is shown in Figure 5.8, are atomic parts of the canvas artifacts to support the development method with a name and a description. For that, they are divided into items, relationships, application domains, and instances.

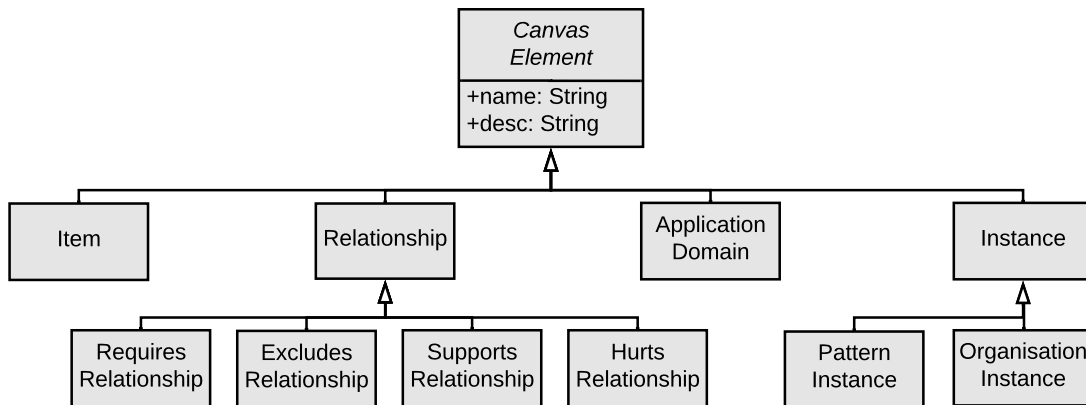


Fig. 5.8 Metamodel of the Canvas Elements

The **Items** are the different business model decisions that are placed as sticky notes on the canvas models. Here, different decisions (e.g., *Advertisements*, *Subscription*) on different refinement levels (e.g., *Monthly Subscription*, *Yearly Subscription*) exist.

The **ApplicationDomains** are part of the context in which the knowledge of the modeling artifact should be applied. Here, different, partially overlapping, domains (e.g., *Mobile App*, *ToDo App*) can exist.

The **Relationships** are dependencies that can occur between two different items. Here, those dependencies show influences of one item to another (e.g., *Native App needs Store License*). Moreover, they are predefined *Relationships* that support the development. The usage of relationships in the *Canvas Module* is shown in Section 7.3.1. Here, the *RequiresRelationship* shows a strong dependency from one item to another (e.g., *a Business User requires High Privacy*), while the *ExcludesRelationship* shows an opposing dependency between two items (e.g., *High Privacy excludes Advertisements*). Moreover, the *SupportsRelationship* shows a positive influence on one item to another (e.g., *Content Curation supports High Quality*), while the *HurtsRelationship* shows a negative influence (e.g., *Mass-Market hurts High Quality*).

The **Instances** provide a selection of a subset of the items for a specific annotation. Here, those selections show a grouping for a particular use case (e.g., *Recommendations*). Moreover, they are predefined *Instances* that support the development. The usage of instances in the *Canvas Module* is shown in Section 7.3.1. Here, the *PatternInstances* refer to subsets of items that are often used together (e.g., *Items of Razer-Blade Pattern*), while the *OrganisationInstances* map subsets of items to an existing organization (e.g., *Items of Todoist Organization*).

Out of those atomic *Canvas Elements*, we combine the *Canvas Building Blocks* as a compact representation of the underlying information for the *Canvas Artifacts*. Those special artifacts are described in Section 5.2.1.

5.3.2 Modeling of Canvas Building Blocks

The *Canvas Building Blocks*, which metamodel is shown in Figure 5.9, are the hierarchical ordering of the canvas elements that have a name and a description. We based this building block on the concept of feature models (FMs), which are used for modeling the structure of different software variants within Software Product Lines (SPLs) with a common graphical representation [ABKS13]. To transfer the concept of FMs, each building block that corresponds to an author contains multiple application domains, multiple items as trees, and multiple selections of relationships and instances.

For the multiple **ApplicationDomains**, each building block is connected to the domains in which the knowledge can be used. Here, those application domains (e.g., *Mobile App*) might be partially overlapping to each other (e.g., *Streaming Apps* and *Messenger Apps* are part of *Mobile App*).

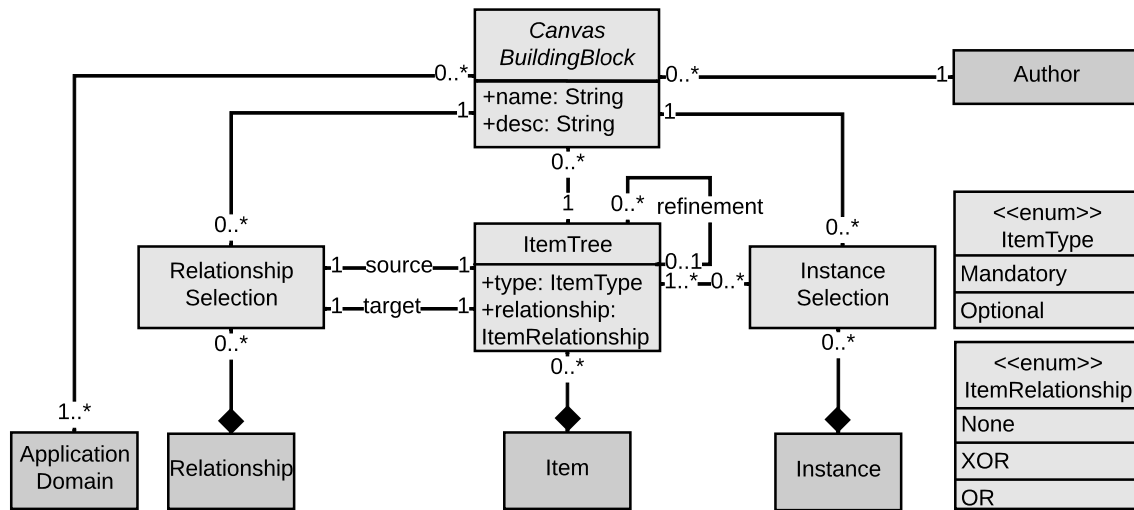


Fig. 5.9 Metamodel of the Canvas Building Blocks

The **ItemTrees**, see Figure 5.10 for the visual notation, consists of all possible combinations stored in a single hierarchical FS of features from which valid combinations are selected. Features can be *Mandatory* or *Optional* for a valid selection. Moreover, there can be *Or*-relationships, where at least one sub-feature needs to be selected, or *Xor*-relationships, where exactly one sub-feature needs to be selected. To refine the valid selections, the cross-tree relationships of *requires* or *can be used*. Each item tree presents the business decisions on the *Canvas Models*. Here, every tree item is used as a wrapper for an item of the *Canvas Elements* (e.g., *Subscription*). Those items can be refined (i.e., *refinement*) using the hierarchical structuring of the tree (e.g., *Subscription* into *Monthly Subscription* and *Yearly Subscription*). Moreover, like in FMs, items have a *Mandatory* (i.e., need to be selected) or *Optional* (i.e., can be selected) type. Moreover, the items have *None* (i.e., just a simple parent-child relationship), an *XOR* (i.e., just a single child is selected), or an *OR* (i.e., at least one child is selected) relationship to their children.

For the multiple **RelationshipSelections** and **InstanceSelections**, each building block is connected to wrappers that link the item trees to the canvas elements. Here, each relationship selection has a starting (i.e., *source*) as well as an ending (i.e., *target*) connection and is linked to a relationship item (e.g., *requires*). In contrast to FMs, we also add additional supporting and hurting dependencies to support the guidance with weaker recommendations. Moreover, each instance selection has connections to a subset of items and is linked to an instance item (e.g., *pattern*). Here, both selections are used to support the BMD with the *Canvas Module*, as explained in Section 7.3.1.

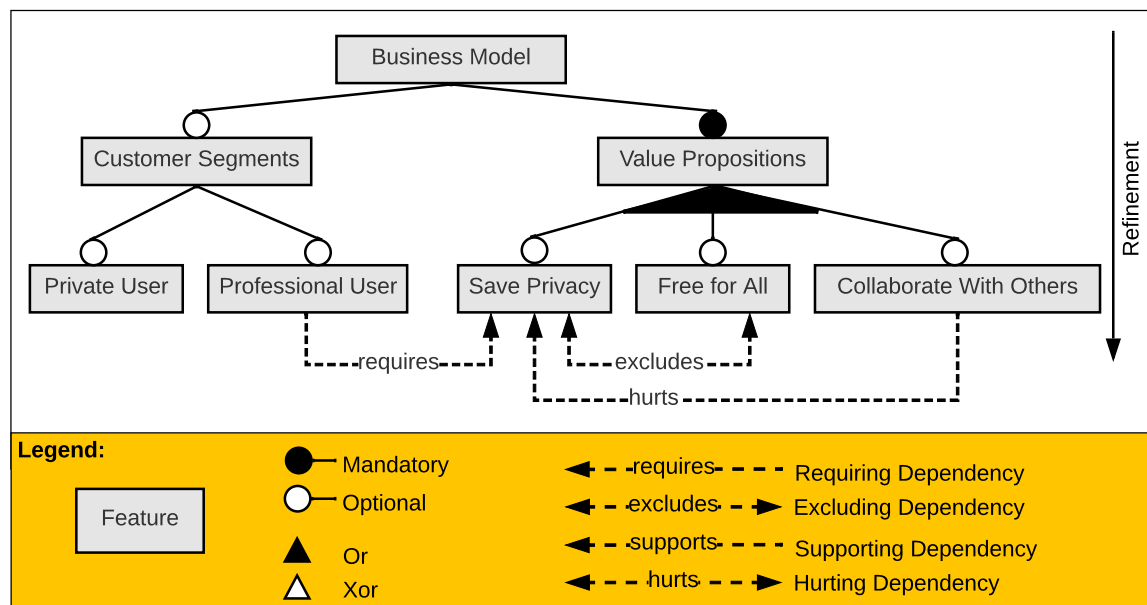


Fig. 5.10 Exemplary Visualisation of the Canvas Building Block

An example of the item tree and relationship visualization is shown in Figure 5.10. Here, we have the *Business Model*, which is divided to the *Customer Segments* and the *Value Propositions*. Here, both items are refined in the hierarchy (e.g., *Customer Segments* to *Professional User*). Moreover, different relationships (e.g., *Professional User* requires *Save Privacy*) exist.

In order to visualize the *Canvas Building Blocks*, we use the representations of *Canvas Models* as geometric modeling language.

5.3.3 Modeling of Canvas Models

The *Canvas Models*, which metamodel is shown in Figure 5.11, are based on a geometric-based BMML. Each canvas is a geometric visual representation of the *Canvas Artifacts*, as explained in Section 5.2.1, that has a name and a description. For that, each model that corresponds to an author contains multiple relationships, a table of multiple columns and rows, and multiple cells.

For the multiple **Relationships**, each model is connected to the relationships that can be used between the inserted items. Here, relationships can be predefined (e.g., *requires*) so that they can be used by the *Canvas Module* as well as custom support modules of the development support engineers.

For the multiple **CanvasRows** and **CanvasColumns**, each model is connected to a grid consisting of rows and columns. For that, *CanvasRows* define an ordering of the rows (i.e.,

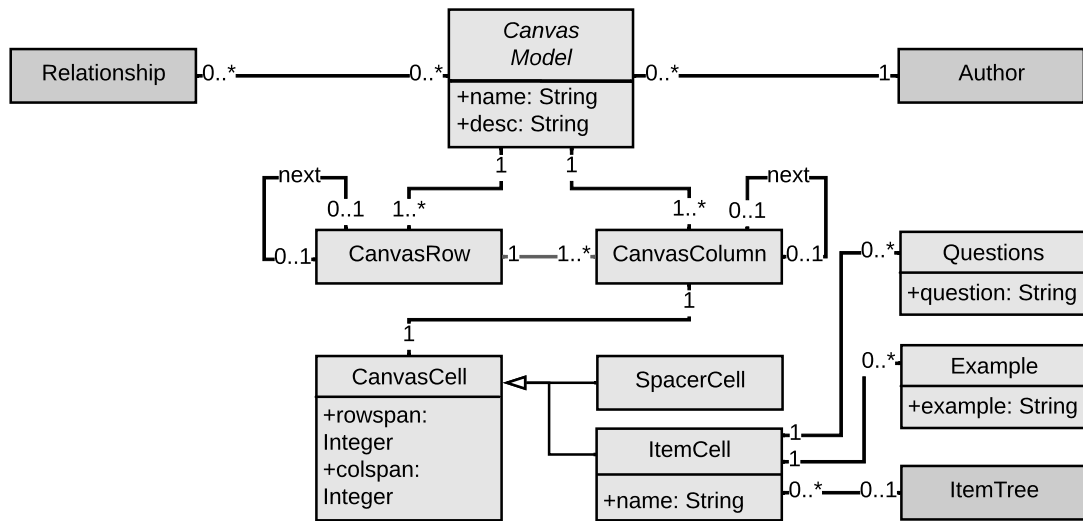


Fig. 5.11 Metamodel of the Canvas Models

next) that are used by the canvas. Each of those rows is connected to multiple *CanvasColumns* as an ordering of the columns (i.e., *next*) where cells can be defined.

For the multiple **CanvasCells**, each column is connected to a cell for representation. The visual notation for the BMC is shown in Section 2.3.1. Here, each cell can span over multiple rows (i.e., *rowspan*) or columns (i.e., *colspan*). Moreover, every cell is divided into an *ItemCell* for inserting the items within the cell and a *SpacerCell* for supporting the structuring with an empty cell. Each cell for items has a name (e.g., *Customer Segments*) and is connected to multiple *Questions* (e.g., *For whom are we creating value? Who are our most important customers?*) and *Examples* (e.g., *Mass Market, Niche Market, Segmented, Diversified, Multi-sided Platform*). Those examples and questions provide guidance for the stakeholders to fill out the canvas. Last, every cell might be connected to an *ItemTree* from a canvas building block with items that can be inserted into the specific cell.

An example of the model for the *Business Model Canvas*, as visualized in Figure 5.1, is shown in Figure 5.12. Here, we have an instance of the *CanvasModel* that is connected to the *RequiresRelationship*, the *ExcludesRelationship*, the *SupportsRelationship*, and the *HurtsRelationship*. Moreover, we have the three *CanvasRows* to specify the grid rows of the canvas. Together with the *CanvasColumns*, we specify placeholders that are filled with the nine *ItemCells* for the nine building blocks. Here, for example, the cell for the *Key Partners* has a *rowspan* of 2 to model the double height in contrast to the *Key Activities* with *rowspan* of 1. For simplification, we have not added the different *Questions* and *Examples* for each *ItemCell* within the model.

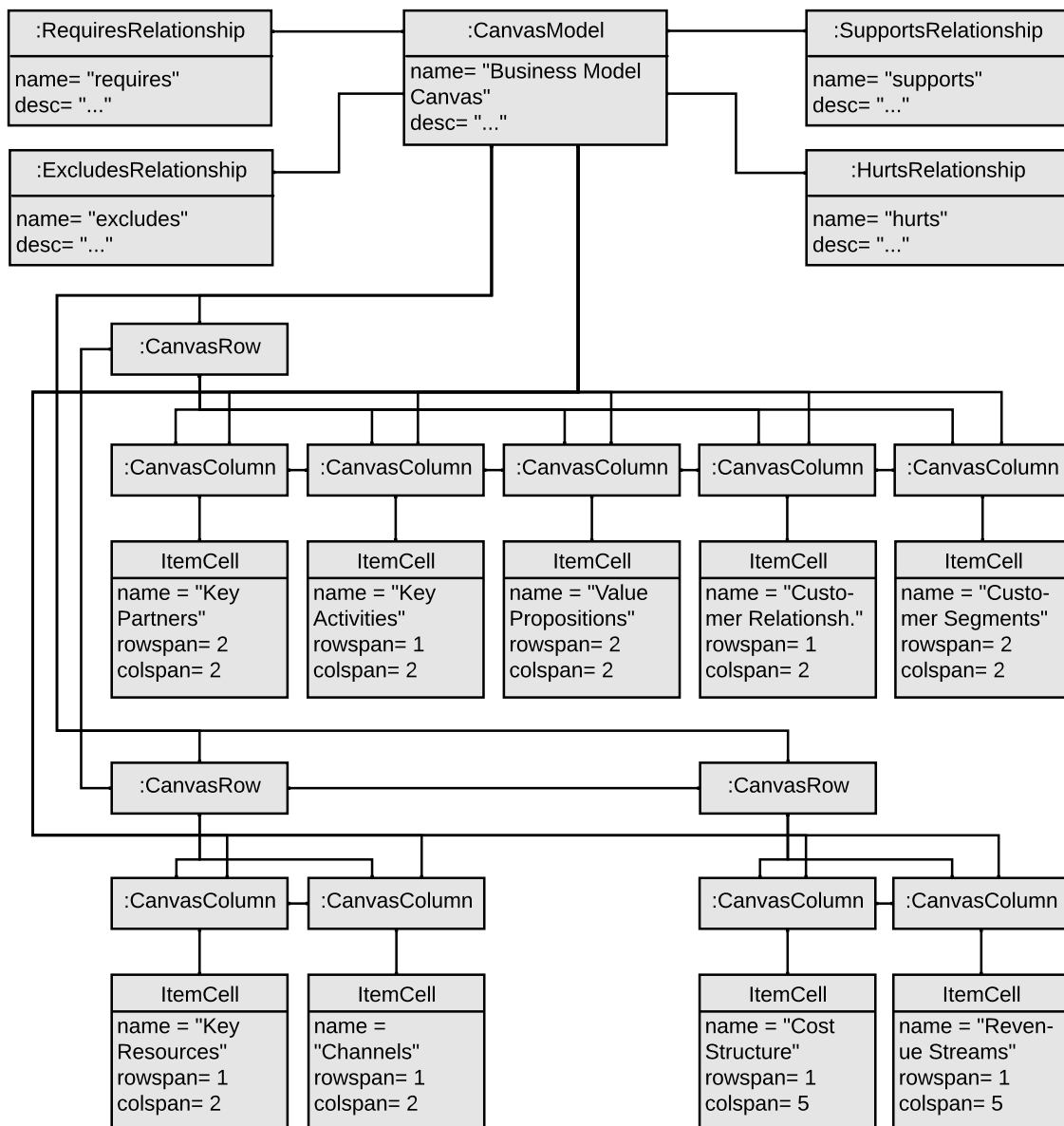


Fig. 5.12 Exemplary Model of the Business Model Canvas

5.4 Summary

Within this chapter, we have provided the conceptual solution for our first stage on knowledge provision of methods and models. For that, we have developed a method repository to structure the development method and a canvas repository to support the development with canvas modeling artifacts. For both repositories, we have shown the underlying metamodels together with exemplary instantiations.

For the **Method Repository**, we have defined the method elements, the method building blocks, and the optional method patterns. The method elements are the atomic parts of the method consisting of the possible situational factors, the defined types, the provided phases, the involved stakeholders, the accomplished tasks, the developed artifacts, and the used tools. Those elements are connected into method building blocks that combine a single task with multiple types, phases, stakeholders, tools, artifacts, and situational factors. Those building blocks are optionally arranged through method patterns that use multiple situational factors and types together with a BPMN process part.

For the **Canvas Model Repository**, we have defined the canvas elements, the canvas building blocks, and the canvas models. The canvas elements are the atomic parts of the modeling artifacts consisting of the applied application domains, the used items, their possible relationships, and exemplary instances. The elements are connected into canvas building blocks that have multiple application domains and a hierarchical ordering of the items together with their relationships and subsets of instances. The building blocks are represented through canvas models that combine multiple relationships together with a grid of the canvas model.

Based on both repositories, we will show the second stage of composition and enactment of development methods for our approach in the next chapter. Here, we will show the composition of the development method out of the method repository and the canvas artifacts out of the canvas model repository. Moreover, we will present the enactment of the development method to modify the canvas artifacts during the conduction of the development steps.

Chapter 6

Composition and Enactment of Development Methods

In the previous chapter, we showed the first stage of our approach by providing development knowledge in the form of a method repository and a canvas model repository. Based on that, this chapter shows the second stage of our solution concept by composing development methods out of those repositories and enacting them. For that, we first refine our SRs and give an overview of the stage (6.1). Based on that, we describe the composition (6.2) and the enactment of the development method (6.3). Finally, we summarize our procedure within the stage (6.4).

6.1 Requirements and Overview

The composition and enactment of development methods is the second stage of our approach, which aims to construct and execute a BMDM to develop a business model for a given context. For that, we refine the SRs, which were derived in Section 4.1.1, of *SR 4: Context Consideration*, *SR 5: Development Method Construction* and *SR 6: Development Method Execution* into detailed Development Method Requirements (DMR) together with providing an overview of both steps to do.

The *Composition of Development Methods* is used to compose the methods out of the method repository based on the situational factors of the organization and connect the canvas artifacts within the method to canvas models of the canvas model repository based on the application domains of the service. Here, the approach needs the possibility to characterize that context together with guidance to construct the method out of the method building blocks. Moreover, the connection of canvas artifacts to multiple canvas building blocks is needed to

support the usage of knowledge from various domain experts. During the whole construction and connection, the quality of the method needs to be checked. Moreover, to allow the unified usage of different model knowledge, those knowledge needs to be consolidated, and different positions between experts need to be detected. Therefore, our DMRs are:

- **DMR 1: Characterization of Context:** The solution should provide the characterization of the context in the form of the situational factors of the organization and the application domains of the service.
- **DMR 2: Guidance in Method Construction:** The solution should provide guidance in the construction of the method based on defined situational factors.
- **DMR 3: Quality Checking during Method Construction:** The solution should provide a quality checking of the method during the whole method construction.
- **DMR 4: Connection of Model Knowledge:** The solution should provide connections between the canvas artifacts of the constructed method and the canvas models based on the defined application domains.
- **DMR 5: Consolidation during Model Construction:** The solution should provide the consolidation of canvas model knowledge with the ability to detect conflicts during the consolidation process.

The *Enactment of Development Methods* is used to create executable development processes out of the composed development methods and conduct the corresponding development steps to develop (canvas) artifacts. Here, the approach needs a lightweight execution engine that executes the development process to provide management over the whole process. Moreover, guidance in the development steps based on the knowledge of the method and the model repository, together with the collaborative development of (canvas-) artifacts, should be provided. Last, the approach needs to allow changes in the context due to uncertainties in the BMD. Therefore, our DMRs are:

- **DMR 6: Execution of Development Method:** The solution should provide a lightweight execution engine to execute the composed development method as a development process.
- **DMR 7: Management of Development Process:** The solution should provide management of the whole development process, including the conducted development steps and the developed artifacts.

- **DMR 8: Guidance in Development Steps:** The solution should provide guidance for the business developer in the development steps based on the knowledge of the method and the model repository.
- **DMR 9: Collaborative Development of Artifacts:** The solution should provide the collaborative development of the artifacts between the business developer and different stakeholders within the development steps.
- **DMR 10: Change of Context:** The solution should provide the change of the context in the form of situational factors of the organization and the application domains of the service.

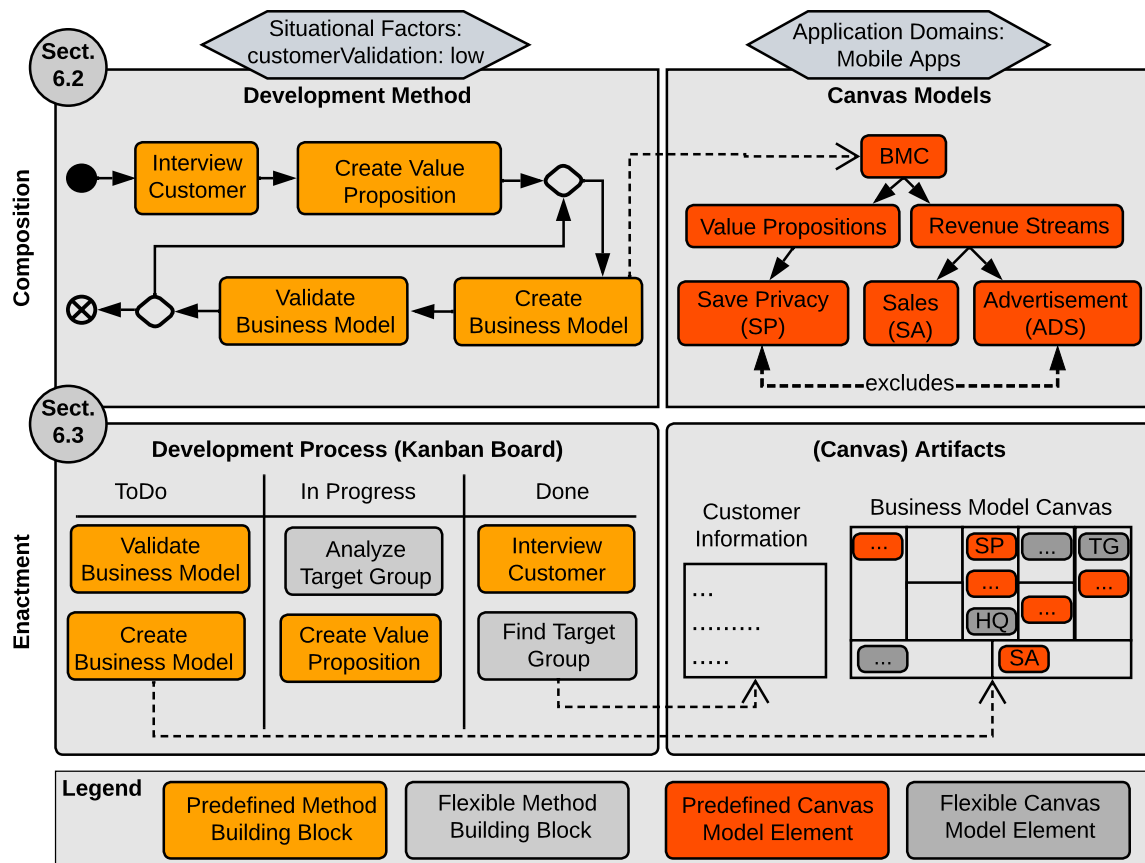


Fig. 6.1 Overview of the Composition and Enactment of Development Methods

Out of the DMRs of the *Composition of Development Methods* and the *Enactment of Development Methods*, we develop an overview of the second stage, as shown in Figure 6.1. The *Composition of Development Methods* is explained in Section 6.2. Here, the context in which the business model should be developed is defined (e.g., *customerValidation: low*), the

development method is composed (e.g., *Interview Customer* before *Create Value Proposition*), and the canvas models are connected (e.g., *Create Business Model* to *BMC*). The *Enactment of Development Methods* is provided in Section 6.3. Here, the steps of the development method are executed (e.g., *Interview Customer*), the (canvas-) artifacts are created (e.g., *Business Model Canvas* for *Create Business Model*), and the context might be changed (e.g., *customerValidation* to *medium*). In the following, we present the details of both phases.

6.2 Composition of Development Methods

The development method composition is needed to construct a development method out of the method repository and the canvas model repository based on a predefined context of the situation of the organization and the application domains of the service. With this, we ensure to provide a development method that best fits the needs of the business developer. The abstracted phase for the *Composition of the Development Methods* is presented in Figure 6.2. Here, in the beginning, the *Method Engineer* defines the *Context Factors* and *Additional Construction Constraints* by interviewing the *Business Developer*. Out of that, he composed the development method by constructing the method based on situational factors and checking its quality. Last, he connects the composed models by consolidating the models based on the application domains and checking their quality. The output of this phase is the *Composed Development Method with Connected Models*.

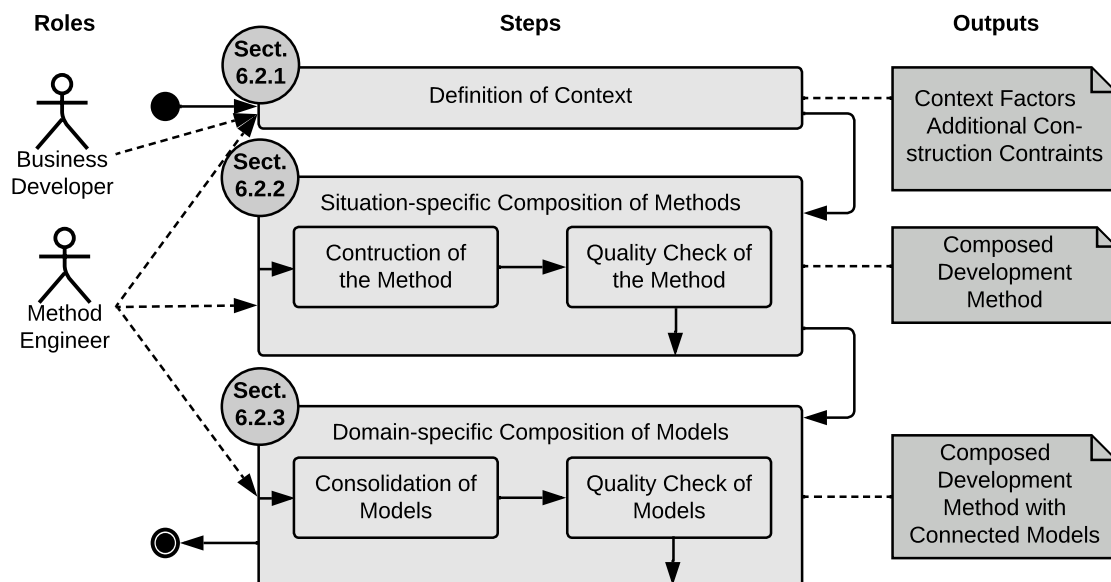


Fig. 6.2 Abstracted Phase of Composition of Development Methods

In this section, we first show the definition of the context out of both repositories (6.2.1). Based on that, we show the situation-specific composition of different method building blocks and, optionally, method patterns to a method (6.2.2). Last, we show the domain-specific composition of different canvas building blocks to a model (6.2.3).

6.2.1 Definition of Context

Before the *Method Engineer* starts with the construction of the method and the connection of the models, he needs to receive the context for which the business model should be developed from the *Business Developer*. For that, the *Method Engineer* conducts an in-depth interview with the *Business Developer* to clarify the purpose and set the potential scope of the development. Here, the *Method Engineer* might question already conducted tasks towards the BMD (e.g., already completed competitor analysis or financial calculations) and corresponding created artifacts (e.g., existing filled-out canvases or created mockups). Moreover, he might ask for current and planned tasks (e.g., current tests with customers or future development of prototypes) and artifacts (e.g., future landing page or prototype). By considering both the past and the future, he increases the chance of composing a development method that best fits the *Business Developer's* purpose. Out of the interview results, the *Method Engineer* derives the explicit context factors and additional construction constraints (cf. composition of situational development methods in Section 2.2.2).

The **Context Factors** are split up into the *Situational Factors* of the organization and the *Application Domains* of the service. The *Situational Factors* are used to identify the most suitable development method during the construction of the methods. The list of possible factors is derived from the specific *Method Elements* in the *Method Repository*, as explained in Section 5.2.1. Here, the *Method Engineer* selects a fitting subset for the organization by comparing their descriptions from the domain experts with the interview results. After that, he chooses an appropriate value for each identified factor. The *Application Domains* are used to identify the most related modeling artifacts during the construction of the models. The list of possible domains is derived from the specific *Canvas Elements* in the *Canvas Model Repository*, as explained in Section 5.3.1. Here, the *Method Engineer* selects a suitable subset that fits best to the developed service by comparing the existing application domains with the interview results. For example, as shown in Figure 6.3, the *Method Engineer* selects the *Situational Factors* of *customerValidation: low* and *marketSize: mass* together with the *Application Domains* of *Mobile App* and *ToDo App* based on the described situation of the *Business Developer* of the mobile to-do app.

The **Additional Construction Constraints** contain information about the organization and the service that can not be directly modeled as factors and domains but influence

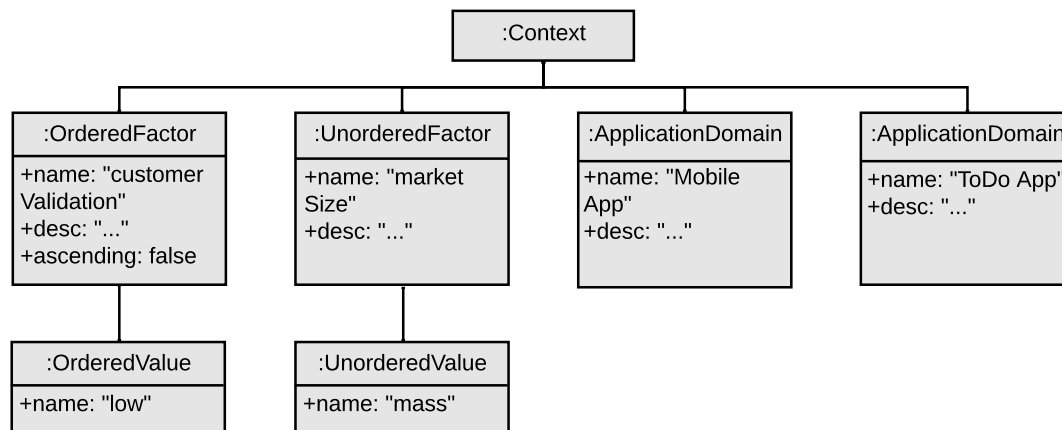


Fig. 6.3 Exemplary Definition of the Context

the development method. That information is manually collected and used within the construction of the methods and the models. For the methods, that information can be divided into already conducted steps in the past, current conductions in the present, and planned steps in the future, together with their existing or planned information about the created artifacts, involved stakeholders, or used tools. This information, in turn, might have an impact on the selected *Method Building Blocks* or their arrangement. For example, the *Business Developer* might have already conducted a competitor analysis in the past or planned to use a certain prototyping tool in the future. For the models, that information can be divided into business decisions in the past, the current decision in the present, or planned future decisions. For example, the *Business Developer* might have already tried out an advertisement model in the past or planned to integrate collaboration features into his service. By considering both, the tailoring of the development method is improved.

Based on the context factors and the additional construction constraints, the *Method Engineer* can start with the situation-specific composition of the method.

6.2.2 Situation-specific Composition of Methods

For the situation-specific composition of the methods, the provided *Situational Factors* with chosen values and additional construction constraints are used to select the *Method Building Blocks* and optional *Method Patterns* of the *Method Repository*. The *Situational Factors* are used as a recommendation mechanism by automatically ordering the useable building blocks (and patterns) according to the weighted distance between the required factors of the organization and provided factors of the building blocks and patterns within the *Method Repository*. Here, the weighted distance is calculated by summing up the distance of each

factor value and dividing them by the number of factors. While nominal values for factors need a concrete matching (e.g., *mass* vs. *niche* for *marketType* has distance 1), ordinal values are weighted according to the scale (e.g., provided *developmentSkills* is *medium* and required *developmentSkills* was *high* has distance 0.5 if *developmentSkills* can have values of *low*, *medium*, *high*). Moreover, matching values in the including direction (e.g., provided *developmentSkills* is *medium* and needed *developmentSkills* was *low* has distance 0) are automatically included to cover *Method Building Blocks* (and *Method Patterns*) that outperform the defined situation. This recommendation mechanism ensures the consideration of the best fitting building blocks (and patterns) according to the provided knowledge of the domain experts in the *Method Repository*. Nevertheless, they have to be cross-checked with the additional construction constraints to fill also those needs. For that, the *Method Engineer* manually needs to control the fulfillment of all additional constraints while selecting patterns and building blocks. Based on that weighted orderings and additional influencing factors, the *Method Engineer* has to construct the method and check the quality.

For the **Construction of the Method**, we support a pattern-based or phase-based construction. For the *Pattern-based Construction of the Method*, the *Method Engineer* starts by choosing a *Method Pattern* with an *InitialisationType* (e.g., *Init Development*) ordered by the matched *Situational Factors* from the *Method Repository*. Next, he fills each activity with a *Type* inside that pattern with a *Method Building Block* (e.g., *Interview Customer* for *discover*) or a *Method Pattern* (e.g., *Validation Cycle* for *validate*) of the needed *Type* or the *GenericType*. Those patterns and building blocks with the needed *Type* are again ordered as a list through their weighted matching between their *Situational Factors* and the factors of the organization. With this pattern-based construction, we provide the *Method Engineer* flexibility in his choices by ensuring control in their integration. This process is repeated until all activities in the method are filled out with building blocks or patterns. To allow a continuous expansion of the method, extending the first pattern by another pattern of *InitialisationType* is possible (e.g., *Init Validation* after *Init Development*). An exemplary pattern-based construction of a method can be seen in Figure 6.4. Here, the *Init Development* is chosen as the first pattern. Based on that, the activity of *discover* is filled with the building block of *Interview Customer* and the activity of *validate* is filled with the pattern of *Validation Cycle*. For the *Phase-based Construction of the Method*, the *Method Engineer* starts by selecting the preordered *Phases* he wants to support from the *Method Elements* of the *Method Repository*. After that, he selects the used *Method Building Blocks* as activities for each phase. Those building blocks with the needed *Phase* are ordered as a list through their weighted matching between their *Situational Factors* and the factors of the organization. After all building blocks have been chosen, the *Method Engineer* provides an execution order of those steps.

For that, each building block is assigned a unique ascending number to define the sequences through the steps of the phases. With this phase-based construction, we provide the *Method Engineer* simplicity in the construction of the method. After the whole method has been constructed using patterns or phases, the abstract information from the building blocks needs to be initialized (cf. the method building block in Section 5.2.2). For that, single items of the groups for the stakeholders, tools, and artifacts inside the *Method Building Blocks* need to be selected (e.g., choose between the Business Model Canvas or Business Information as *Artifact*). Moreover, during the composition, concrete elements from the abstract lists need to be selected (e.g., selecting specific prototyping software as a *Tool*). With this, we ensure the maximal customization of the method for the *Business Developer*.

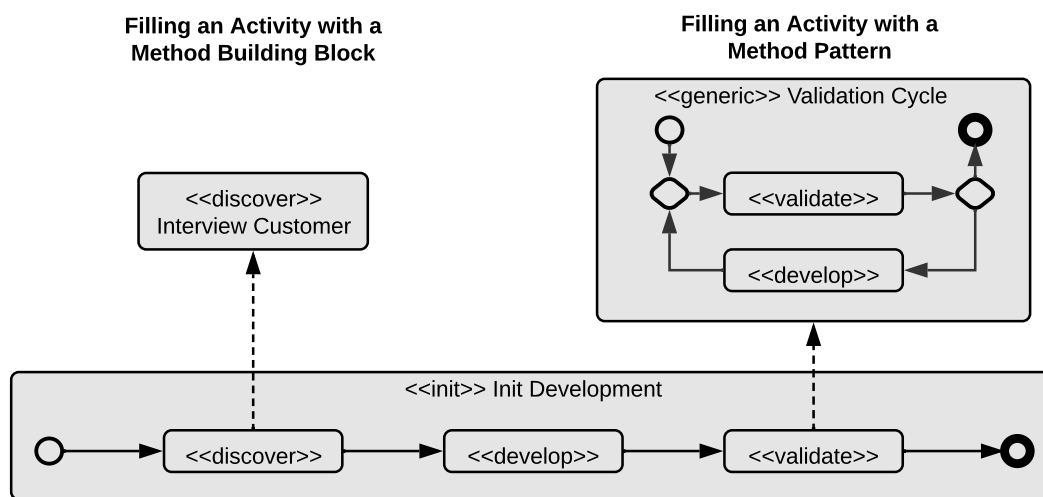


Fig. 6.4 Exemplary Pattern-based Construction of Methods

For the **Quality Check of the Method**, the *Method Engineer* receives support to detect warnings and errors during the method construction. With this thesis, we detect the following warning and errors with the partial support of existing algorithms:

- **Low Value Warning:** The warning states that an ordinal factor of a used building block or pattern has a lower value than the requested one during the definition of the context. This should be controlled as the lower building block or pattern could harm the BMD. We identify those warnings by automatically checking each building block and pattern against the chosen ordinal factors.
- **Invalid Value Warning:** The warning states that a nominal factor of a used building block or pattern has a different value than the requested one during the definition of the context. This should be controlled as the wrong building block or pattern could

harm the BMD. We identify those warnings by automatically checking each building block and pattern against the chosen nominal factors.

- **Unreachable Path Warning:** This warning states that some parts of the constructed method are not reachable during the enactment. This should be controlled as it is very likely that this behavior was created accidentally by nesting different patterns. We identify those warnings by automatically checking a complete traversal of the underlying graph with an existing algorithm.
- **Missing Activity Error:** This error states that an activity within a pattern has not been filled with a building block. This needs to be resolved as empty activities can not be processed during the enactment. We identify those errors by automatically checking each activity of the patterns for missing building blocks.
- **Incomplete Activity Error:** This error states that within an activity, not all items have been selected. This needs to be resolved as incomplete activities can not be processed during the enactment. We identify those errors by automatically checking each building block for the initialization of the elements.
- **Missing Artifact Error:** This error states that within an activity, an input artifact is used that is missing as an output artifact in a previous activity. This needs to be resolved as an activity without all input artifacts can not be processed during the enactment. We identify those errors by automatically back-propagating input artifacts towards the phases or nested patterns to match with output artifacts with an existing algorithm.

In order to ensure the quality of the method, the *Method Engineer* has to resolve the errors and control the warnings manually. After the quality has been checked, the *Method Engineer* can start the domain-specific composition of the used canvas models for the canvas artifacts within the constructed method.

6.2.3 Domain-specific Composition of Models

For the domain-specific composition of the models, the selected *Application Domains* and additional construction constraints are used to select the specific *Canvas Building Blocks* of the *Canvas Model Repository*. The *Application Domains* are used as a recommendation mechanism by automatically selecting corresponding building blocks. Iteratively, the *Method Engineer* selects every *Method Building Block* that is used within an activity of the composed method with at least one *CanvasArtifact* as an *OutputArtifact* (e.g., *Create Business Model* with the *Business Model Canvas*). For each connected *Canvas Model* (e.g., *Business Model*

Canvas), the *Method Engineer* selects one or multiple *Canvas Building Blocks* (e.g., *Mobile App*) that can be used to support the filling out of the canvas models during the enactment. After that, the *Method Engineer* manually can remove elements of the building blocks by considering the additional construction constraints. This, in turn, reduces the model knowledge that *Business Developer* needs to consider during the enactment. If multiple *Canvas Building Blocks* are selected, those knowledge needs to be consolidated and the quality needs to be checked.

For the **Consolidation of Models**, we provide a feature-based or taxonomy-based consolidation. For the *Feature-based Consolidation*, our approach creates an empty reference *Canvas Building Block* as the starting point and includes all knowledge from the first domain-specific *Canvas Building Block* (e.g., *Mobile App*) that should be consolidated. Instead of directly adding those *Canvas Elements* to the reference building block, the approach uses virtual links between those models. This, in turn, simplifies the reaction to changing models over time. At this point, the *Method Engineer* is able to remove not needed elements. Next, the approach consolidates the other selected building blocks with the reference one. For that, the approach automatically consolidates all *Items (Customer)* with the same name as links. Moreover, also all *Relationships* (e.g., *excludes*) that exist in the building block to consolidate are linked automatically if the items are added (cf. the canvas building block in Section 5.3.2). Moreover, due to the different expertise of the experts also, different names and hierarchies of items can exist. Those are handled manually by removing items and adding links between the items. An example of the consolidation of models can be seen in Figure 6.5. Here, the *Reference Canvas Building Block* already contains knowledge from the first building block to which the *Mobile App Canvas Building Block* is consolidated. Here, both items of *Customer Segments* are automatically consolidated using links. To consolidate also items lower in the hierarchy, the items of *One-Sided Market* and *Two-Sided Market* are removed. Moreover, the *Private User* and *Professional User* are manually linked to the *User* and *Supplier*. For *Taxonomy-based Consolidation*, our approach works similarly to feature-based consolidation. However, each item is modeled as optional, and no relationships between the different items exist. This, in turn, reduces the guidance level during the enactment but also the modeling complexity and quality checks during the composition.

For the **Quality Check of Models**, the *Method Engineer* needs to receive support to detect conflicts during the model consolidation. Within this thesis, we detect the following conflicts by continuously analyzing the consolidation steps of the reference building block and the building block to consolidate:

- **Item Conflict:** The conflict occurs if two consolidated items have a different type or relationship. We automatically detect those conflicts by comparing the type and

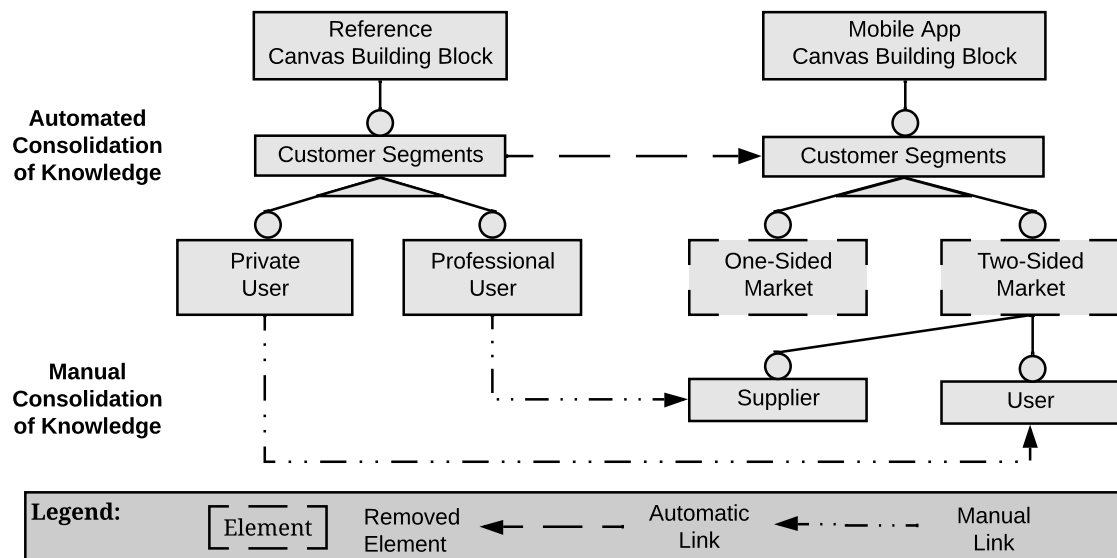


Fig. 6.5 Exemplary Consolidation of Models

relationship of both consolidated items. Here, the engineer needs to choose the existing type/relationship from the reference building block or selects the new type/relationship from the consolidated one.

- **Cross-Relationship Conflict:** The conflict occurs if two consolidated items have wise-versa relationships with different recommendations. This can be required vs. excludes or supports vs. hurts. We automatically detect those conflicts by comparing the cross-tree relationships of both consolidated items. Here, the engineer needs to decide on one of the recommendations.
- **Cross-Hierarchy Conflicts:** The conflict occurs if combinations of cross-tree relationships lead to unreachable items within the hierarchy. We automatically detect those conflicts by creating a minimal spanning tree for each item using an existing algorithm. Here, the engineer needs to remove conflicting cross-tree relationships or items.

During the consolidation process, the *Method Engineer* has to decide how he wants to resolve the conflicts manually. His choices are added as virtual links through the reference building block. After all canvas building blocks have been consolidated, the *Business Developer* can start the enactment of the composed development method.

6.3 Enactment of Development Methods

The development method enactment is needed to execute the composed development method and allow collaboration between different stakeholders during the conduction of development steps. With this, we ensure to guide the business developer and the other stakeholders. The abstracted phase for the *Enactment of the Development Methods* is presented in Figure 6.6. In the beginning, the *Business Developer* selects a development method and the corresponding development steps. Based on that, he executes those steps and creates (canvas-) artifacts in collaboration with *Other Stakeholders*. Moreover, during the execution of the development process, changes in the context might lead to a change in the methods or the models performed by the *Method Engineer*. The outputs of this phase are the *Created (Canvas-) Artifacts*, including the business model.

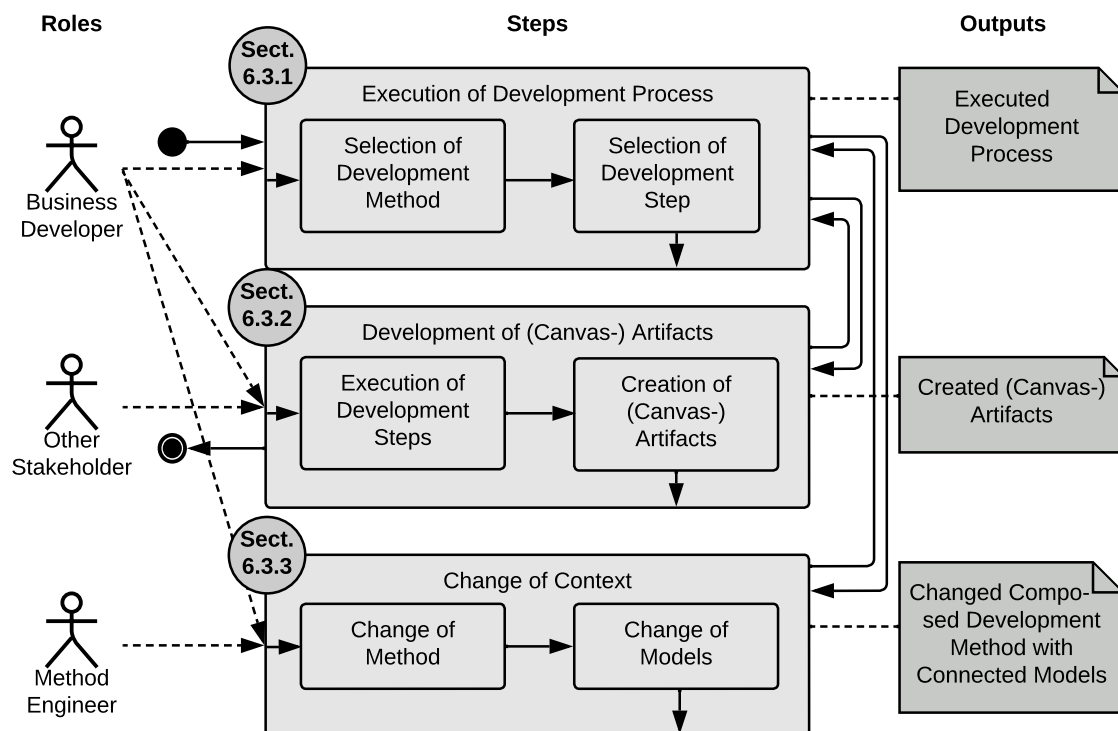


Fig. 6.6 Abstracted Phase of Enactment of Development Methods

In this section, we first show the execution of the development process using our lightweight execution engine (6.3.1). Based on that, we present the involvement of different stakeholders during the artifact development (6.3.2). Last, we deal with changes in the context during the execution of the development process (6.3.3).

6.3.1 Execution of Development Process

To develop a business model, the *Business Developer* has to execute a development process and conduct various development steps. For that, a lightweight process execution engine can be used. Here, a business process engine is a software tool that supports executing and monitoring business processes (modeled, for example, using BPMN) and their activities [Sti14]. On the market, and especially for the de-facto standard of BPMN, different closed-source process engines like the IBM Process Manager¹ or open-source process engines like the Camunda Platform² exist. Instead of using an existing one with a huge feature set that is not needed in our case, we develop our own lightweight one whose features directly fit the domain of BMD. Our lightweight process engine is based on a Kanban board where the development steps are grouped into *ToDo*, *In Progress*, and *Done* steps. During the development, the *Business Developer* selects a subset of development steps in *ToDo* and puts them into *In Progress* for conduction and in *Done* after finishing the steps. Here, the execution engine supports using the already composed development methods and newly ad-hoc created development methods.

The **Composed Development Methods** are used to provide the *Business Developer* maximum control in developing a business model. For that, he selects a composed development method created by the *Method Engineer*. After that, the process engine interprets that development method depending on pattern-based or phase-based construction. For the *Pattern-based Construction*, the approach instantiates and visualizes the corresponding BPMN process model. Next, all directly executable development steps that are modeled as activities in BPMN with connected *Method Building Blocks* are inserted as *ToDo*s of the Kanban board. New executable development steps are automatically inserted into the board based on the execution of those steps by the *Business Developer*. Moreover, within the process models, the *Business Developer* can manually decide on different gateways to pass. For the *Phase-based Construction*, the approach instantiates and visualizes the development method as a sequential process model. Here, each *Method Building Block* is modeled as a single activity and the first building block is inserted as *ToDo* of the Kanban board. After executing a single building block, the following building blocks are automatically inserted into the board. In both cases, the *Business Developer* can manually execute additional development steps to be flexible in the overall development. For that, he selects a corresponding *Method Building Block* from the *Method Repository* where the defined *Situational Factors* order the building blocks. If those selected building blocks are connected to *Canvas Artifacts*,

¹Website of IBM Process Manager: <https://www.ibm.com/docs/en/bpm/>

²Website of the Camunda Platform: <https://camunda.com/>

he must also compose those models or select already composed ones. Moreover, a check on the needed input artifacts has to be done during the selection.

The **Ad-hoc Created Development Methods** are used to provide the *Business Developer* maximum flexibility in developing a business model. After selecting that option, he has no predefined BPMN or sequences process model to use. Instead of that, he needs to set a context in terms of the *Situational Factors* and the *Application Domains* from the repositories (cf. the context definition of the *Method Engineer* in Section 6.2.1). After that, the process engine creates an empty Kanban board where he manually needs to select all development steps to conduct as a *Method Building Block* from the *Method Repository*. Here, he is also responsible for composing the models if he needs them for the *Canvas Artifacts* (cf. the model composition of the *Method Engineer* in Section 6.2.3). Moreover, he needs to care about potential warnings like that all input artifacts of one of his selected development steps have already been created as output artifacts before.

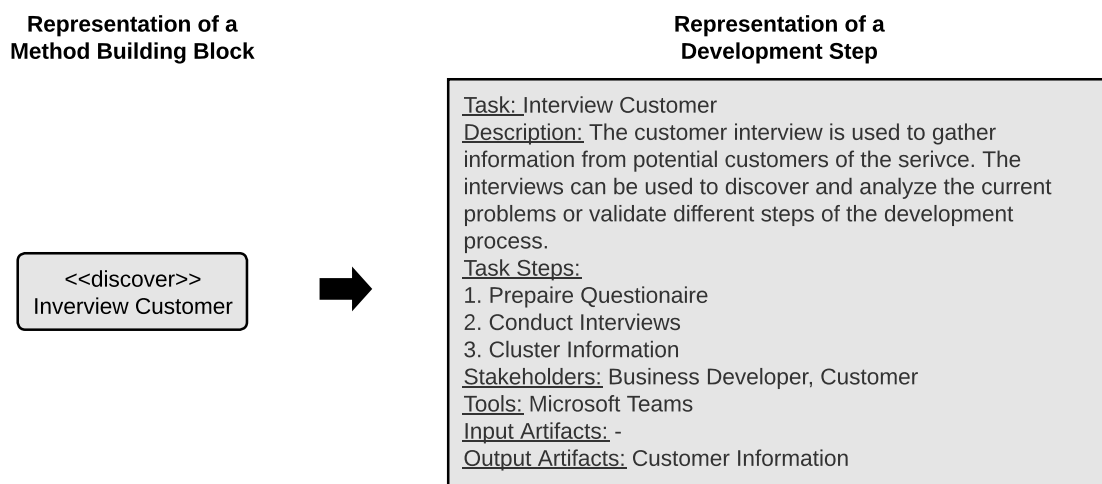


Fig. 6.7 Exemplary Transition from a Method Building Block to a Development Step

During the conduction of the development steps, the process engine provides guidance to the *Business Developer*. For that, each development steps is connected to a *Method Building Block*, as exemplary shown in Figure 6.7. Based on this connection, he directly receives written information about the conduction from the *Domain Experts*. For example, in Figure 6.7, each step has a *Task* (e.g., *Interview Customer*) together with a *Description* (e.g., *The customer interview...*) of what should be done during the task. Moreover, concrete *Task Steps* (e.g., *1. Prepare Questionnaire, 2. ...*) are provided as a guideline for the *Business Developer* and other involved *Stakeholders* (e.g., *Customer*). Moreover, those steps can be supported by the usage of *Tools* (e.g., *Microsoft Teams*). Those tools, in turn, can be used to transfer the required *Input Artifacts* into *Output Artifacts* (e.g., *Customer Information*). With

this support, the *Business Developer* might involve different stakeholders in developing the needed artifacts collaboratively.

6.3.2 Stakeholder Involvement in Artifact Development

During the conduction of the development steps, the *Business Developer* has to create various artifacts that store parts of the development results. This creation, in turn, is, depending on the team size, done on his own or with the involvement of different internal (e.g., marketing manager) and external (e.g., investors) stakeholders. Moreover, those stakeholders might be needed just for specific development steps as modeled in the *Method Building Blocks*. To guide the development of artifacts, the process engine has an *Artifact Manager*. With this manager, the *Business Developer* is able to create artifacts independently or as part of a development step. Moreover, the manager allows the transfer of artifacts between different development processes by providing exporting and importing functionalities. During the creation of artifacts, a division between *Information Artifacts* and *Canvas Artifacts* is done.

For the **Information Artifacts** (e.g., *Customer Information*), the engine provides an artifact description together with an editor. The editor provides a visualization using a graphical user interface with "What-You-See-Is-What-You-Get" (WYSIWYG) features. Within this editor, the *Business Developer* and different *Stakeholders* might collaborate to develop a textual document. For that, the editor is able to import an *Information Input Artifact* that should be modified or directly create a new *Information Output Artifacts*.

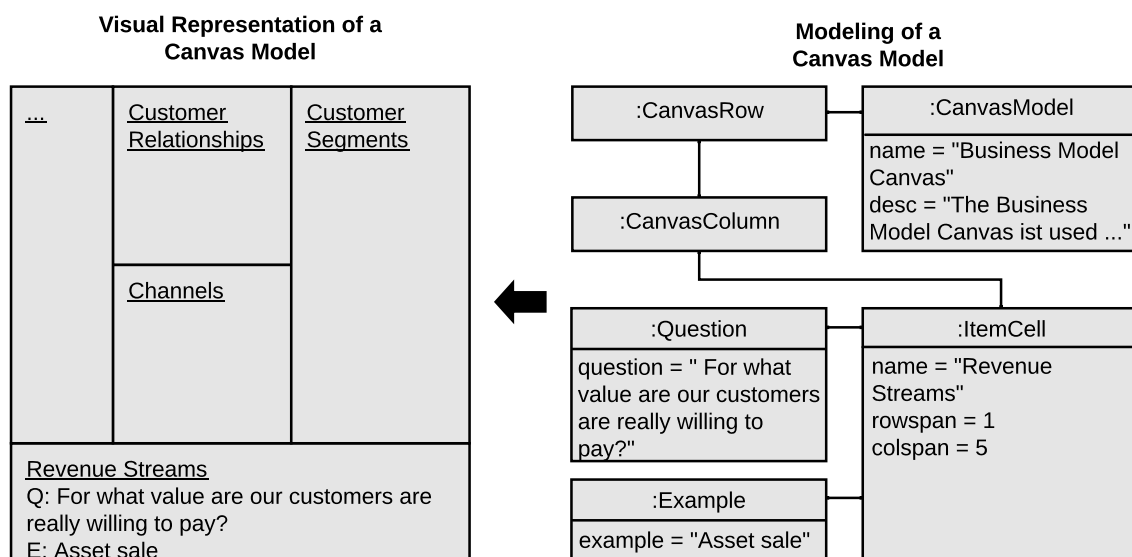


Fig. 6.8 Exemplary Visualisation of Canvas Artifact

For the **Canvas Artifacts** (e.g., *Business Model Canvas*), the engine provides a canvas artifact description together with a canvas board. Here, the board is generated out of the corresponding *Canvas Model* of the *Canvas Model Repository*, as shown exemplary in Figure 6.8 for a part of the Business Model Canvas. Here, we use the *CanvasRows* and *CanvasColumns* as a grid for the positioning of our *ItemCells* (e.g., *Revenue Streams*) on the board. Each cell is positioned based on a defined span of rows and columns, as used within the HTML tables. For each cell, the engine provides guidance by *Questions* (e.g., *For what...*) and *Examples* (e.g., *Asset sale*). Here, the *Business Developer* might collaborate with the different stakeholders on the *Canvas Models*. For that, the board is able to import *Canvas Input Artifacts* or directly create new *Canvas Output Artifacts*.

During the creation of both types of artifacts, the process engine provides a *Collaboration Manager* with a discussion board to allow communication between the *Business Developer* and the different *Stakeholders*. This ensures quick agreements apart from the developed artifacts without the need for additional communication tools. After the creation of the artifact is finished, it is stored as a new artifact or merged with an existing one. Here, the traceability of all artifact changes, together with the discussion, is ensured. For that, the artifact manager provides versioning of all created and modified artifacts together with the possibility of analyzing the changes. Moreover, a change in the context and a subsequent change of the composed development method might be needed during the development.

6.3.3 Change of Context

During the BMD, internal changes in the organization (e.g., changed target market) or external changes in the environment (e.g., changed competitor) might also lead to a change in the context in which the business model is developed. This context, in turn, can affect the *Situational Factors* (e.g., reduced financial resources) or the *Application Domains* (e.g., changed domain). Here, depending on the type of development method, there are two different options. If the *Business Developer* uses an ad-hoc created development method, he needs to change the *Situational Factors* and the *Application Domains* manually. Due to the missing of a process model, he also needs to manually look if already conducted development steps need to be reconducted with changed *Method Building Blocks* or *Canvas Artifacts* need to be recomposed using different *Canvas Building Blocks*. If the *Business Developer* uses a composed development method, he informs the *Method Engineer* about those changes in the context together with additional information (e.g., a new competitor in the market). The *Method Engineer* uses both to modify the composed method and/or the composed models.

For the **Composed Method**, the *Method Engineer* gets the requested changes of the *Business Developer* for the *Situational Factors* and the *Application Domains*. After reviewing

them, he accepts those changes or manually modifies them. Based on the initial choice of a pattern-based or phase-based construction, the composed method needs to be changed. During the whole change, he needs to consider the new additional information well as the initial additional construction constraints of the *Business Developer*. For the *Pattern-based Construction*, he receives an overview of the BPMN process model, including the *Method Patterns*, *Method Building Blocks*, and an *Execution Step Marker*. Similar to the method composition in Section 6.2.2, he might now change both. By adding *Method Building Blocks* where an instance already exists in the existing development method, he can choose to create a duplicate of those building blocks instead of using a new one. With this, he is flexible in changing the complete development method without losing the information of already executed development steps. An exemplary view of a pattern-based composed method based on *Init Development* together with the current *Execution Step Marker* on *Develop Value Proposition* can be seen in Figure 6.9. In this example, the building block of *Interview Customer* is changed to *Survey Experts*, and the pattern of *Validation Cycle* is modified to the building block of *Interview Experts*. For the *Phase-based Construction*, he receives an overview of the sequential process model, including the *Method Building Blocks*, and an *Execution Step Marker*. Similar to the method composition in Section 6.2.2, he changes the considered phases or building blocks. Here, he is able to remove existing building blocks or add new building blocks. Like in type-based construction, he is able to duplicate existing building blocks instead of creating a new one.

After constructing the development method, the *Method Engineer* needs to replicate the current enactment state for both constructions. For that, he conducts a mock execution of the development steps by adding existing artifacts as output artifacts of those development steps. Moreover, he sets the *Execution Step Marker* to a new position where the *Business Developer* should continue executing the development process. In our example on Figure 6.9, we set the *Execution Step Marker* to the building block of *Survey Experts* for a type-based construction of the method. Alternatively, instead of that modification, he might also export the created artifact from the method, compose a completely new method (see method composition in Section 6.2.2), and import the artifacts into that method.

For the **Composed Models**, the *Method Engineer* already reviews the changed *Application Domains*. Out of that, he receives an overview of all *Method Building Blocks* that use affected *Canvas Artifacts* in the BPMN or sequence process model. Model by model and for both the feature-based and taxonomy-based consolidation, the *Method Engineer* revisits the composed models by consolidating the knowledge of new models creating virtual links, or separating the knowledge of existing models by removing virtual links. During these changes in the consolidation, he needs to consider the new additional information well as the

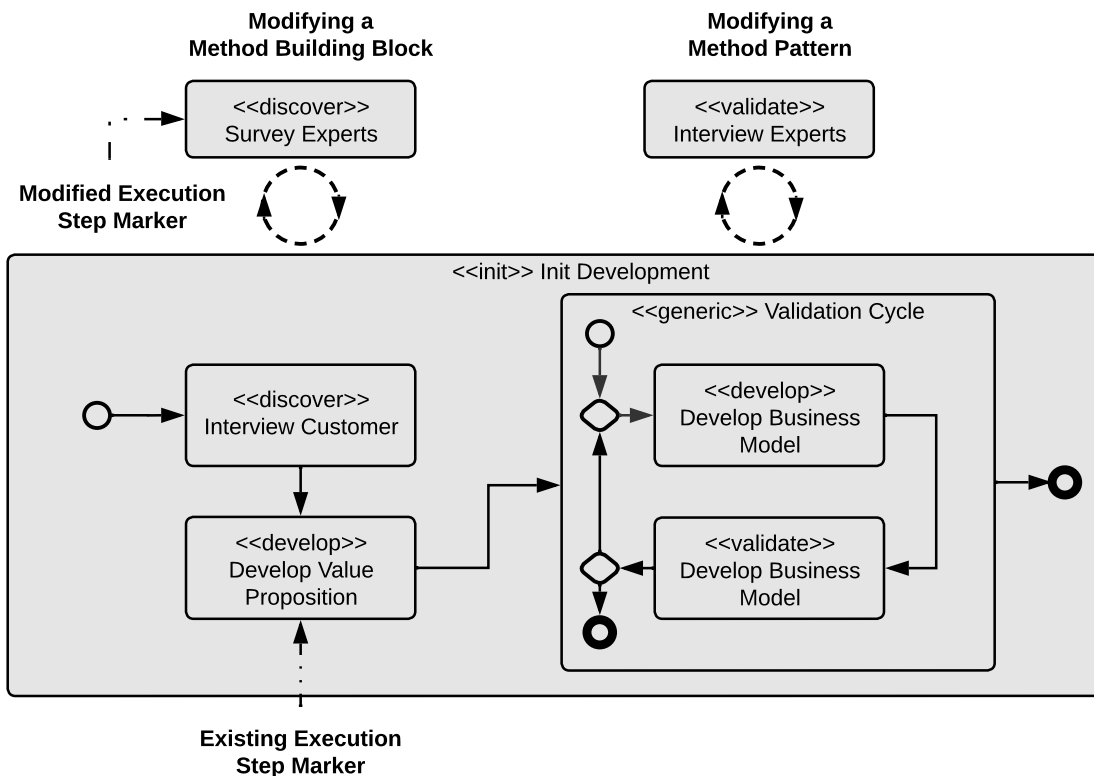


Fig. 6.9 Exemplary Change of Context for a Pattern-based Constructed Method

initial additional construction constraints of the *Business Developer*. Alternatively, instead of modifying the existing composed models, he might also remove all composed models and composes completely new models (see Section 6.2.3) for the canvas artifacts. This, in turn, reduces the overhead if the new models highly differ from the existing models so that the majority of the virtual links need to be changed.

After the context and the connected method and models have been successfully modified by the *Method Engineer*, the *Business Developer* can continue the development of the business model.

6.4 Summary

Within this chapter, we have provided the conceptual solution for our second stage on composition and enactment of development methods. For that, we have shown the construction of methods and consolidation of models based on a defined context in terms of situational factors and application domains, together with an execution of the development process and

conduction of development steps that can be modified through a change in the context. We have explained the necessary parts for both parts based on an exemplary instantiation.

For the **Development Method Composition**, we have defined a modeling for the context of the situational factors and the application domains. Based on that, we support the pattern-based and phase-based construction of the method based on the combination of method building blocks and optionally method patterns, together with checking the quality of the method against different warnings and errors. Moreover, we support the feature-based and taxonomy-based consolidation of the canvas building blocks for the canvas artifacts in the constructed method based on virtual links, together with checking the quality by detecting possible conflicts.

For the **Development Method Enactment**, we have explained an execution of the development process based on the composed or ad-hoc development methods. Here, we support pattern-based and phase-based constructed methods. Based on that, we support the collaboration of different stakeholders on the development of artifacts where we focus especially on canvas artifacts with underlying canvas models. Last, we have provided a modification of the development method, considering the methods and the models, based on changes in the context.

Based on the composition and enactment of development methods, we will show the third stage of support of development steps for our approach in the next chapter. Here, we will show how to support the different development steps of the method building blocks with execution steps based on the support modules. Moreover, we will show the execution of those steps to provide flexible IT support for developing business models. For that, we will present different exemplary modules that support the BMD in the design and the validation phase.

Chapter 7

Support of Development Steps

In the previous chapter, we showed the second stage of our approach by composing and enacting development methods. Based on that, this chapter shows the third stage of our solution concept by supporting development steps using the assistance of support modules. For that, we first refine our SRs and give an overview of the stage (7.1). Based on that, we describe the modularization of development support (7.2) and the application to different types of modules (7.3). Finally, we summarize our procedure within the stage (7.4).

7.1 Requirements and Overview

The support of development steps is the third stage of our approach, which aims to provide flexible assistance using different software tools during the BMD. For that, we refine the SRs, which were derived in Section 4.1.1, of *SR 7: Development Support Formalization*, *SR 8: Development Support Construction* and *SR 9: Development Support Execution* into detailed Development Assistance Requirements (DAR) together with providing an overview of the modularization of the software tools and further application to development steps.

The *Modularization of Development Support* combines different types of IT assistance into dedicated support modules, consisting of software tools with steps to conduct and meta artifacts with artifacts to create that can be used during BMD. Here, the approach needs the possibility to use internal software tools specially programmed for our solution and external software tools that already exist and should be used. For the internal software tools, the approach should atomize those tools into atomic steps to provide a flexible combination of them. Moreover, meta artifacts should be defined to define a common structure that allows reusing the created artifacts across the tools of different support modules. To increase the acceptance by the stakeholders, the modules should be explained understandably together with the possibility of extending them in the future. Therefore, our DARs are:

- **DAR 1: Interoperability of Modules:** The solution should provide interoperability of support modules to allow using internal and external software tools.
- **DAR 2: Atomization of Tools:** The solution should provide atomization of internal software tool functionalities in single steps for maximum flexibility in reusing them.
- **DAR 3: Usage of Meta Artifacts:** The solution should provide structured meta artifacts so that different software tools can create and modify the same artifacts with their assistance.
- **DAR 4: Understandability of Modules:** The solution should provide proper explanations of the software tools and meta artifacts to allow a unified understanding of all support techniques among the business developer and the other stakeholders.
- **DAR 5: Extensibility of Modules:** The solution should provide the creation of new and extension of existing support modules with software tools and meta artifacts for novel assistance techniques.

The *Application to Development Steps* is used to apply the support modules to the different development steps to assist the BMD. Here, during the composition of the development method, the support steps of the software tools should be added to the development steps of the development method. Moreover, during the enactment of the development method, the artifacts based on the meta artifacts are created and modified. Within this, the approach needs to allow the collaborative development of artifacts. Therefore, our DARs are:

- **DAR 6: Guidance in Development Support Construction:** The solution should provide guidance in the construction of development support for specific development steps.
- **DAR 7: Atomized Interface Steps:** The solution should provide an interface between the single steps during the development support and their implementation in the tools.
- **DAR 8: Execution of Development Support:** The solution should provide the execution of the development support as single steps based on the internal software tools.
- **DAR 9: Guidance in Development Support:** The solution should provide guidance in the steps of the software tools to conduct the development steps.
- **DAR 10: Collaborative Development Support of Artifacts:** The solution should provide the collaborative development of the artifacts concerning the meta artifacts.

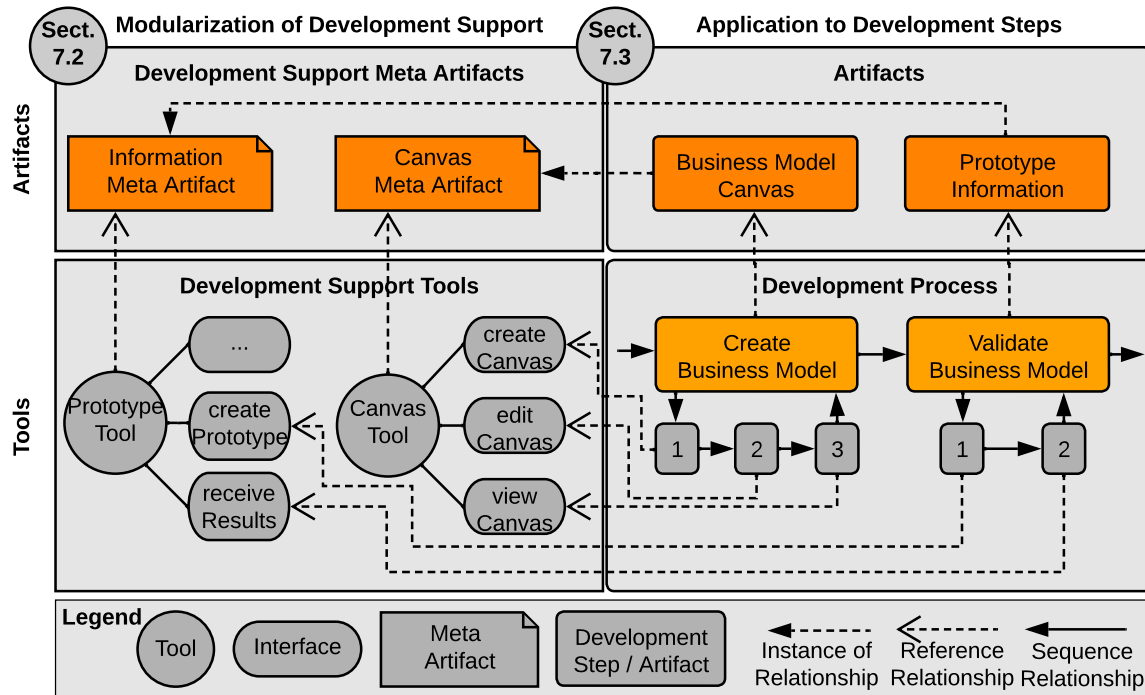


Fig. 7.1 Overview of the Support of Development Steps

Out of the DARs of the *Modularization of Development Support* and the *Application to Development Steps*, we develop an overview of the third stage, as shown in Figure 7.1. The *Modularization of Development Support* is explained in Section 7.2. For that, we present the provision of development support (e.g., *Canvas Tool*), their composition within the development steps (e.g., 1, 2, 3 in *Create Business Model*), and their enactment (e.g., *Business Model Canvas*) during the conduction of those steps. The *Application to Development Steps* for exemplary support modules is shown in Section 7.3. Here, we present three exemplary support modules (e.g., *Canvas Module*) together with the software tools (e.g., *Canvas Tool*) and meta artifacts (e.g., *Canvas Meta Artifact*). In the following, we present details on the modularization concept and the exemplary support modules.

7.2 Modularization of Development Support

The modularization concept is needed to structure the internal development support into software tools and connect them to meta artifacts. Based on those support modules, the development support for specific development steps using the software tools is composed and enacted to develop artifacts based on the meta artifact. With this, we provide additional development support for certain development steps of the second stage. The abstracted phase

for providing the *Modularization of Development Support* is shown in Figure 7.2. Here, in the beginning, the *Meta-Development Support Engineer* needs to provide a *Modularized Architecture* and optionally an SDK for the solution, which the *Development Support Engineer* uses to create and integrate the support modules. Based on that, *Method Engineer* composed the different development steps and, optionally, the development support artifacts to return the *Composed Development Method with Development Support*. That composed development method, in turn, is executed by the *Business Developer* with the support of the *Other Stakeholders* to create different *Development Support Artifacts*. The outputs of this phase are the *Integrated Development Support Modules*, the *Composed Development Method with Development Support*, and the *Created Development Support Artifacts*.

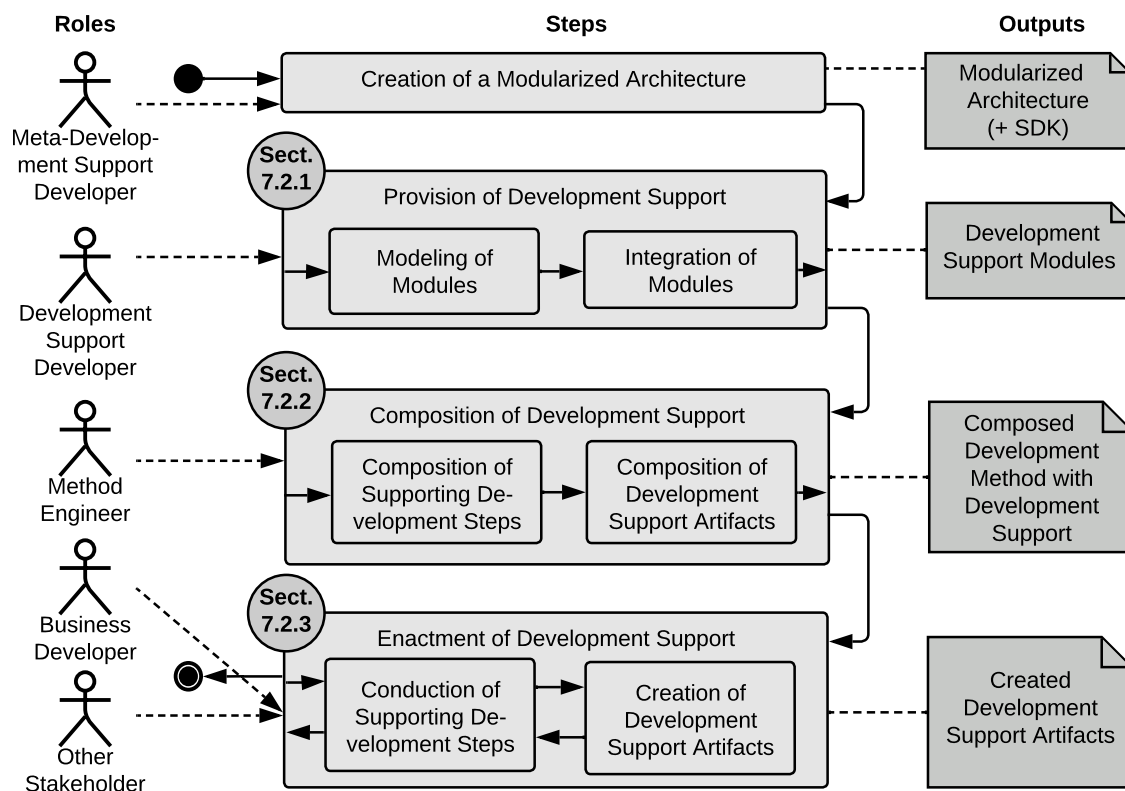


Fig. 7.2 Abstracted Phase of Modularization of Development Support

Based on the modularized architecture, which will be explained in Section 8.1, this section first shows the provision of the development support by integrating the *Development Support Modules* (7.2.1). Based on that, we describe the composition of the *Development Support Steps* and the *Development Support Artifacts* (7.2.2). Last, we present the enactment of the method and conduction of the *Development Steps* to create the *Development Support Artifacts* (7.2.3).

7.2.1 Provision of Development Support

Different software tools have supported the BMD in research and practice in recent years. As divided in Section 2.3.3, that support can cover the visualization of business models (e.g., predefined canvas models to fill out), their design (e.g., presentation of business model patterns), or even decision support (e.g., stress testing of different business assumptions). However, currently, different *Development Support Developers* start from scratch when creating their development support. For that, we provide a modularization concept so that they can focus on the actual supporting features (e.g., analysis of enterprise data) and dismiss commodity features (e.g., canvas representations). Here, in general, we need to divide between internal and external development support.

- **Internal Development Support:** Internal support describes the development support that is specially developed for our solution. Here, the development support is developed on provided constraints and interfaces of our solution so that it can be integrated into the overall concept and the other support techniques. Examples of that could be the design or calculation of business models.
- **External Development Support:** External support describes development support that is developed independently of our solution in terms of software tools. Here, the software tools can be used alone and the created artifacts are added to our solution, or interfaces between our solution and the software tools are developed. Examples of that could be the design or validation of prototyping using existing prototyping tools.

Modularization, in turn, splits up different software functionalities of internal development support into separate parts so that they can be used and modified with fewer dependencies on each other [Par72]. With this, the *Development Support Developer* is able to split his development support into atomic parts so that they can be recombined flexibly to support different development steps. To provide maximum reusability, the *Development Support Developer* has to split the overall *Development Support Module* into different *Development Support Tools* and *Development Support Meta Artifacts*, as shown in Figure 7.3. The support tools ensure the actual functionalities for the development support by providing different *Development Support Steps* that are composed together for specific development steps. The meta artifacts specify the different *Support Artifacts* that are created and modified within the steps. By splitting the modules into tools and meta artifacts, we ensure the easy usage of artifacts by different tools together with their exchange. Moreover, the *Development Support Engineer* can freely decide whether he wants to add more internal business logic into the tools or the meta artifacts as long as the interfaces defined by the *Meta-Development Support*

Engineer are correctly used. For using those interfaces, those modules need to be modeled and integrated in a standardized way.

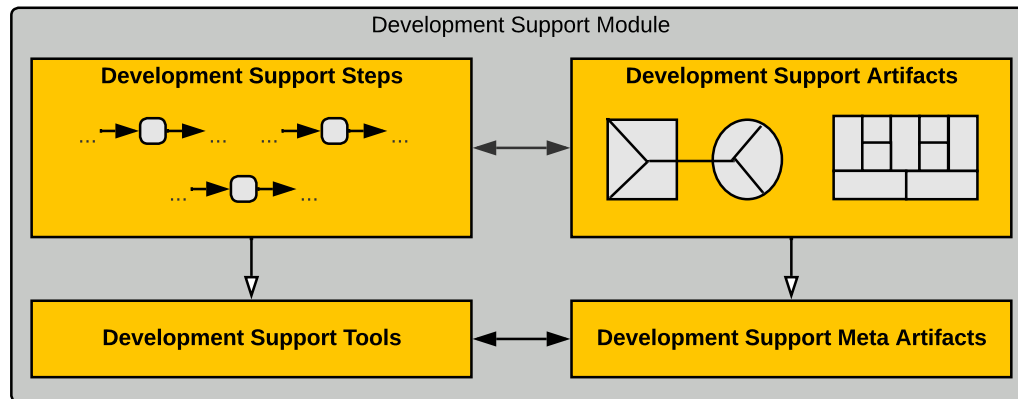


Fig. 7.3 Overview of the Support Modules

The **Modeling of Modules** is done by extending the *Method Elements* of the *Method Repository*. Here, like for the elements, the *Method Engineer* can manually add those elements to the repository or import a configuration that is provided by the *Development Support Engineer* within the *Development Support Module*. Here, the extended part of the metamodel for the method elements is shown in Figure 7.4 on the top (cf. to the metamodel for method elements in Figure 5.3). For adding the development support, we have to change the two existing elements of the *Tool* and the *Artifact*. For the *Tool*, we add the *Modularized Tool* (e.g., *Canvas Tool*) to define the support tools. For the *Artifact*, we add a *Meta Artifact* to support the underlying structure of different artifacts. Moreover, we add the *Information Meta Artifact* as a meta artifact to cover the existing textual information. Based on that, we create the *Information Artifacts* (e.g., *Customer Information*) that are already used as artifacts in the second stage and *Modularized Artifacts* (e.g., *Business Model Canvas*) as two types. Here, the *Modularized Artifacts* also include the canvas artifacts of the last stage. Based on this preparation, we provide the modeling of the *Development Support Module*.

The *Development Support Module*, as shown in Figure 7.4 on the bottom, contains a name, a description, a version, and its relationship to the *Development Support Developer* as *Autor*. Moreover, the module can contain different *Development Support Meta Artifacts* and *Development Support Tools*. The *Development Support Meta Artifacts* are concrete *Meta Artifacts* with a name, a description, a version, and a link to the source code for integrating the specific meta artifact into the modularized architecture. Moreover, each meta artifact can have various *Development Support Artifacts*. The support artifacts can be statically provided by the meta artifact in advance or dynamically created during the knowledge provision. Those artifacts have a name, a description, and a unique identifier for using the artifacts within the

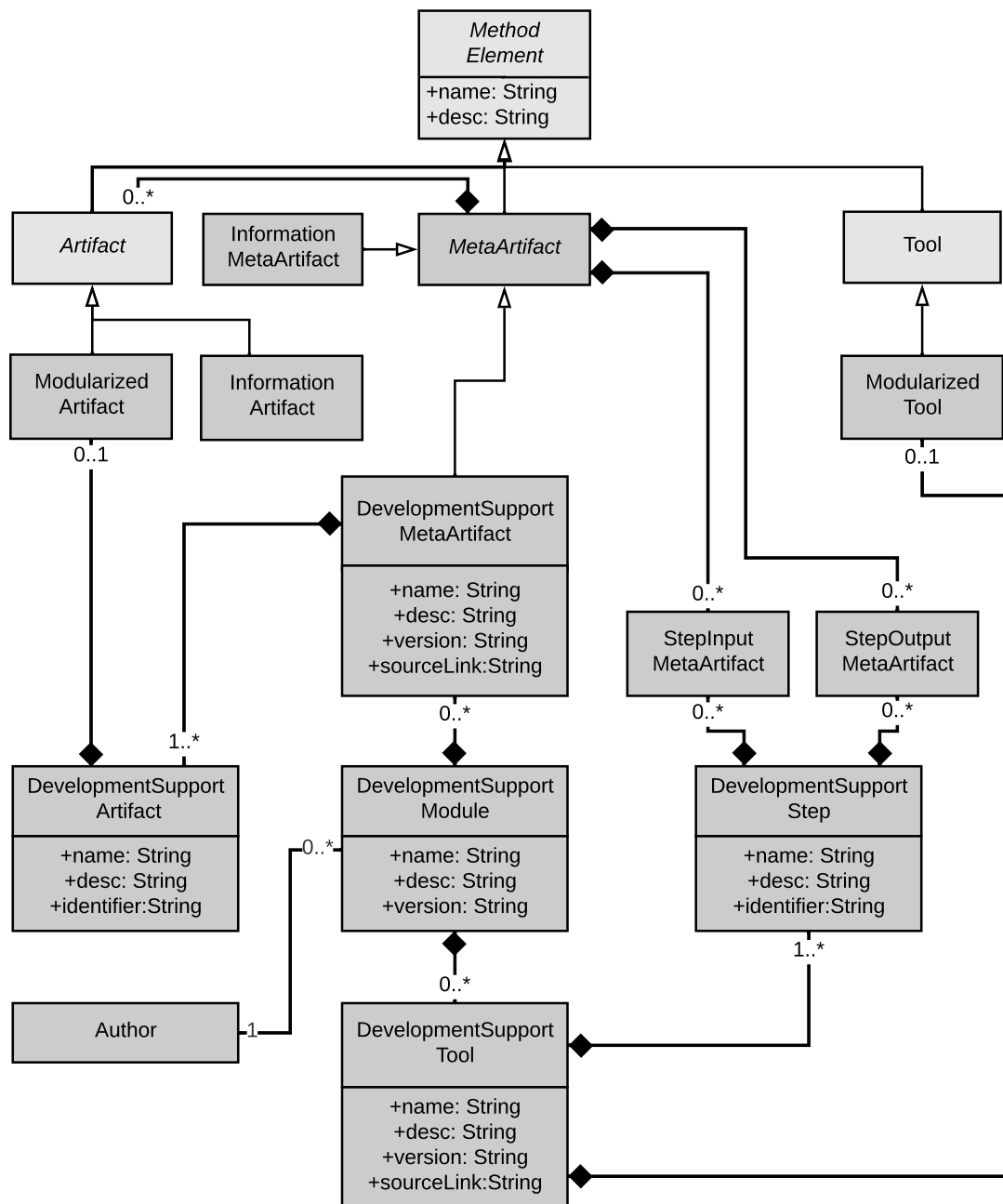


Fig. 7.4 Extension of the Method Elements for the Modularization

support tools. This is done by connecting those artifacts with a *Modularized Artifact*. The *Development Support Tool* has a name, a description, a version, and a link to the source code for integrating the specific tool into the architecture. This is done by connecting the tools to a *Modularized Tool*. Moreover, each tool has certain *Development Support Steps* for proving the atomic functionalities of the tool. The support steps are provided statically by the tool

in advance. Those steps are defined through a name, a description, and a unique identifier for providing an identification during the composition. Moreover, each step has connected *Step Input Meta Artifacts* and *Step Output Meta Artifacts* to set the scope of artifacts they can handle.

An example of the *Canvas Module* can be seen in Figure 7.5. Here, the module is connected to the *Canvas Meta Artifact* and the *Canvas Tool*. The meta artifact holds information about the customizable canvas models as introduced in Section 5.3.3. One example is the *Business Model Canvas* which is connected to the specific *Modularized Artifact* (i.e., *BMC*) of the *Method Repository*. The tool provides the support steps to collaborate with the meta artifact. Examples are the *CreateCanvas* and *RefineCanvas*, which are connected to the needed *Step Input Artifacts* and *Step Output Artifact*. Also, the tool is connected to a *Modularized Tool* (i.e., *CT*) of the *Method Repository*.

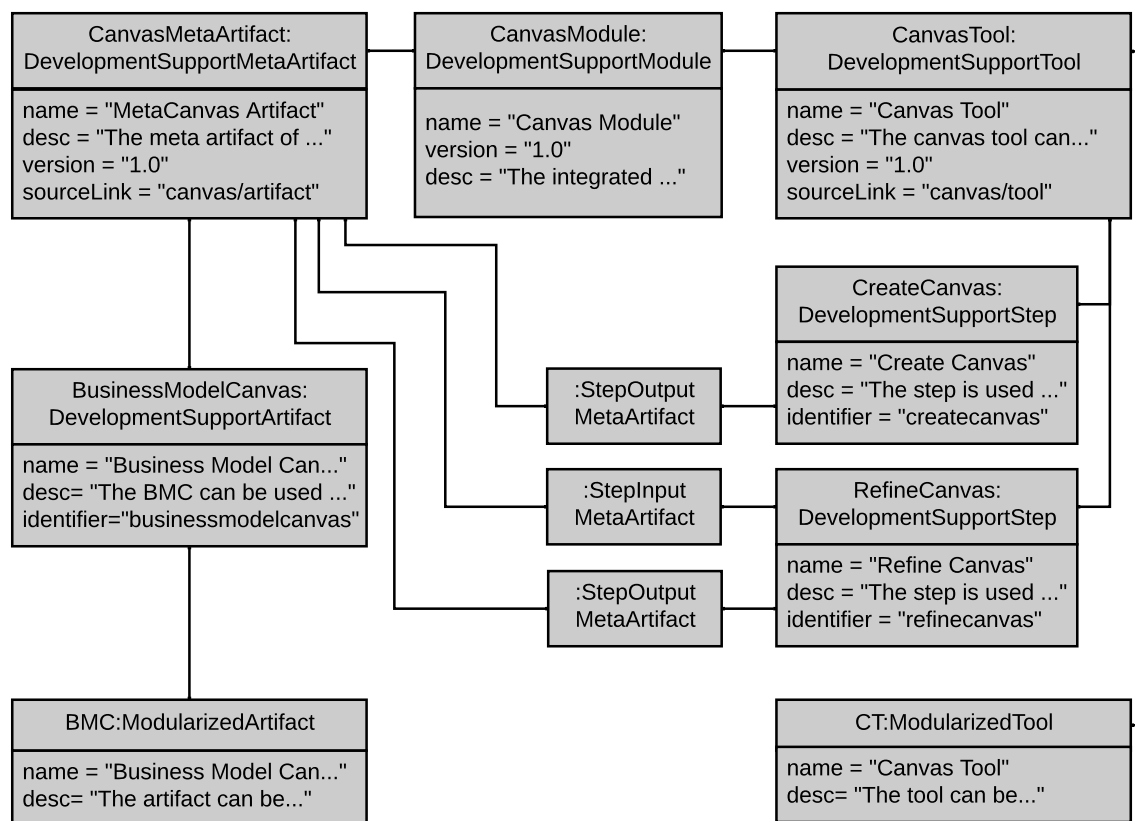


Fig. 7.5 Exemplary Method Elements for the Development Support

The **Integration of Modules** is done by extending the solution's functionalities by directly exchanging information between both. Those standardized information exchange needs to be implemented for the meta artifacts as well as the support tools. We develop that

exchange based on the two concepts of application programming interfaces and software hooks. Application programming interfaces provide direct communication of at least two programs based on a standardized specification. In our solution, we use interfaces to gather information from the meta artifacts and support tools for our solution. Hooks, in turn, directly integrate external program functionalities into an internal program. In our solution, we use hooks to directly use functional parts of the meta artifacts and the support tools in the solution. For providing both, our solution uses an SDK that is able to automatically generate code stubs for a support module with the required interfaces and hooks. Those code stubs, in turn, can be used as a starting point by the *Development Support Engineer* to create its customized development support. Those interfaces and hooks are integrated into the different parts of our solution:

- **Knowledge Provision Hooks/Interfaces:** The hooks and interfaces for the knowledge provision are used to provide information for the usable support tool and conductible support steps, together with creating the dynamic support artifacts from the defined support meta artifacts.
- **Composition Hooks Hooks/Interfaces:** The hooks and interfaces for the development method composition are used to configure and combine the support steps to the development steps together with composing different support artifacts.
- **Enactment Hooks/Interfaces:** The hooks and interfaces for the development method enactment are used to provide information and functionalities for conducting the support steps, together with functionalities for creating and modifying support artifacts within and apart from those support steps.

In the following sections, we explain the integration of the specific interfaces and hooks during the different stages. The overall module design that the *Development Support Developer* needs to implement is explained in Section 8.1. Inside the knowledge provision of development support, which is explained in this section, the meta artifacts need an interface to provide all modeled information, as shown in Figure 7.4, to allow an automated extension of the *Method Repository*. This includes especially the support artifacts of the meta artifacts, which can be dynamically created within the module and need to be accessed for a connection to the *Modularized Artifacts*. Moreover, also the support tools need an interface to provide all information to extend the *Method Repository*. This includes the connection of the support tools to the *Modularized Tools*. Here, the support tools might also use an additional hook to show specific functionalities or configuration possibilities during the knowledge provision of the solution. With this, the supporting tool is able to provide functionalities for the *Method*

Engineer to utilize knowledge, for example, about different meta artifacts, apart from the composition and enactment. In our solution, we allow the integration of hooks for the provided knowledge in the navigation bar of our software tool.

One example is the already mentioned *Canvas Module*, which is explained in Figure 7.5. Here, the *Canvas Meta Artifact* and the *Canvas Support Tool* provide interfaces to access their information (e.g., name, description) for the *Method Repository*. Moreover, the *Canvas Meta Artifact* is able to provide dynamically created canvas models (e.g., *Value Proposition Canvas*, *Business Model Canvas*) for their connection with the *Modularized Artifacts*. Those canvases, in turn, are created by the *Method Engineer* with a software hook in the navigation bar for the *Canvas Support Tool* to allow the creation of canvas elements, canvas building blocks, and canvas models from different *Domain Experts*.

Based on modeling the modules and their integration into the solution, the *Method Engineer* can start composing development support for specific development steps for the *Business Developer*.

7.2.2 Composition of Development Support

During the creation of *Method Building Blocks*, which is explained in Section 5.2.2, the *Method Engineer* is also able to freely construct the development support out of the *Development Support Modules* for the *Business Developer*. For that, the combined *Development Support Steps* of the *Development Support Tools* are used to create and modify the *Development Support Artifacts* based on the *Development Support Meta Artifacts*. Here, we need to consider the composition of the development support steps and the artifacts.

The **Composition of the Development Support Steps** is done during the knowledge provision of the *Method Repository*, as explained in Section 5.2. Here, we extend the *Method Building Blocks* as shown in Figure 7.6 so that they provide those support. For that, the *Method Engineer* selects the used *Modularized Tools* and *Artifacts* for inputs and outputs (consisting of *Information Artifacts* and *Modularized Artifacts*). Based on that, he combines different *Execution Steps* to transform those *Input Artifacts* into *Output Artifacts*. Every *Execution Step* is connected to a single *Development Support Step* of the used *Development Support Tool*. With this, the approach knows what action should be performed by the tool and which *Artifacts* of which *Meta Artifact* are expected as *Step Input Meta Artifact* and *Steps Output Meta Artifact*. With this information, the *Method Engineer* selects the *Execution Step Input Artifacts* and *Execution Step Output Artifact* for each *Execution Step*. Here every *Execution Step Input Artifact* is connected to an initial *Input Artifact* or an *Execution Step Output Artifact*, while every *Execution Step Output Artifact* is connected to another *Execution Step Input Artifact* or a final *Output Artifact*. With this, the approach provides a pipeline

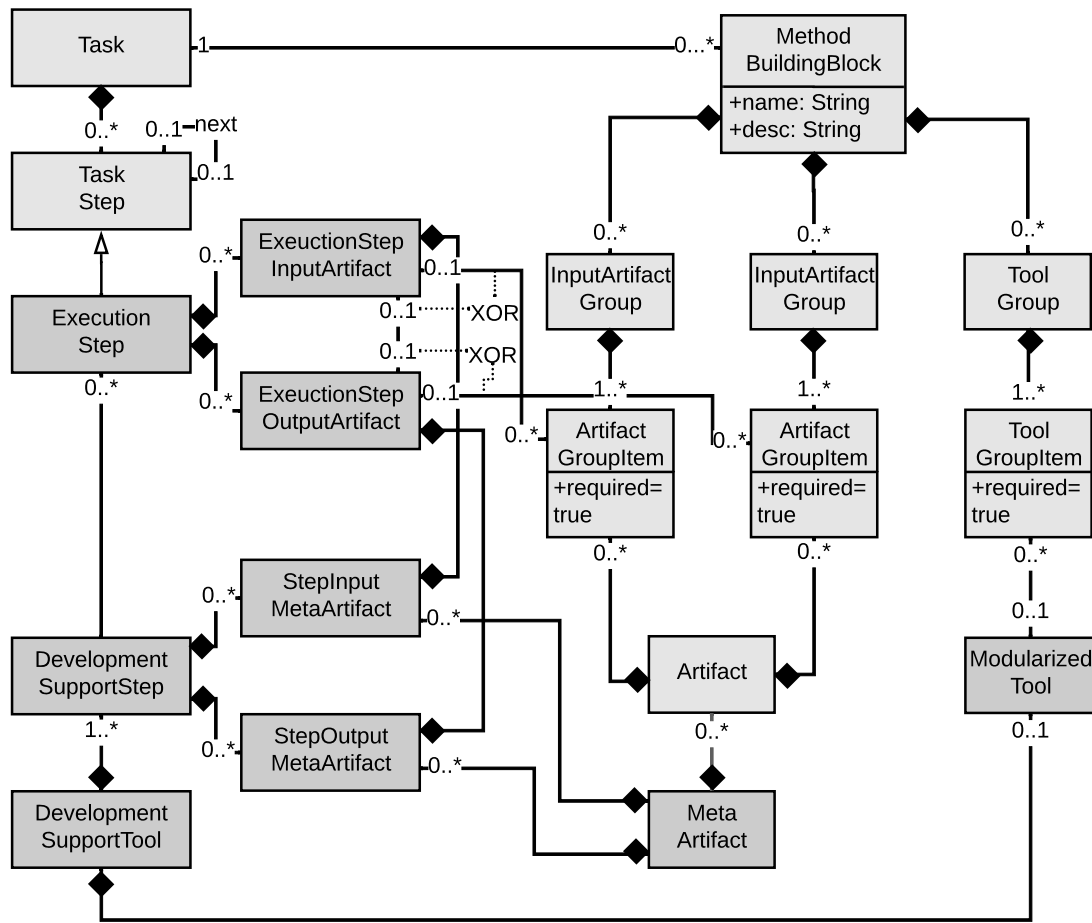


Fig. 7.6 Extension of the Method Building Blocks for the Modularization

process for transforming the artifacts. During the whole construction of the development support, the quality needs to be checked against warnings and errors. In our approach, we provide automated detection of the following warnings and errors:

- **Missing Default Tool Warning:** This warning states that one of the development support tools is not set as default for the building block. This should be resolved as it reduces the chances of dismissing the selection during the composition of the development method. We identify those warnings by checking the selected development support steps against the selected tools.
- **Missing Default Artifact Warning:** This warning states that one of the development support artifacts is not set as default for the building block. This should be resolved as it reduces the chances of dismissing the selection during the composition of the devel-

opment method. We identify those warnings by checking the selected development support artifacts for their default setting.

- **Missing Input Step Artifact Error:** This error states an execution step input artifact is not connected to an input artifact or an execution step output artifact. This needs to be resolved as otherwise, the execution step can not be executed. We identify those errors by checking the inputs of all execution steps for their connections.
- **Missing Output Artifact Error:** This error states that a modularized output artifact is not connected to an existing output step artifact. This needs to be resolved as otherwise, the output artifact can not be created during the conduction of the development step. We identify those errors by checking all modularized output artifacts for their connections.

Moreover, interfaces and hooks are used during the composition of the development support steps. Here, the support tools provide interfaces to gather information about the development steps to conduct, including the steps' descriptions, input artifacts, and output artifacts. Moreover, each step might provide a hook with a form to configure the step during the composition of the development support. Here, those forms can be freely combined out of text and selection lists together with buttons. Moreover, the forms might be dynamically filled during the composition of the development support steps.

One example of such development support for the *Business Model Development* can be seen in Figure 7.5. Here, we use the *CanvasTool* and the *CanvasMetaArtifact*. The support is provided by the three *Execution Steps* (i.e., *Step1*, *Step2*, *Step3*), which are related to the *Development Support Steps* of *CreateCanvas*, *EditCanvas*, and *RefineCanvas*. Between those steps, the *Artifact* is transferred until it is used as an *OutputArtifactGroupItem*. Moreover, we also provide a hook for the *CreateCanvas* step to allow the selection of a specific canvas model where we specify the *Business Model Canvas* during the construction of the development support steps. With this, we avoid setting the specific canvas by the *Business Developer* during the conduction of the development step.

The **Composition of the Development Support Artifacts** is done during the domain-specific composition of the modeling artifacts, as explained for the canvas artifacts in Section 6.4. Here, the *Method Engineer* constructs the development out of *Method Building Blocks* and, optionally, *Method Patterns*, including building blocks with development support. Here, he chooses different selections for the stakeholders, artifacts, and tools. Moreover, he discovers the different execution steps. To provide flexibility during the composition, we created a hook for a customized form similar to the provision of development support for artifacts. That hook might be, for example, used to configure the composition of different related artifacts or select created artifacts from the provided knowledge. However, for this,

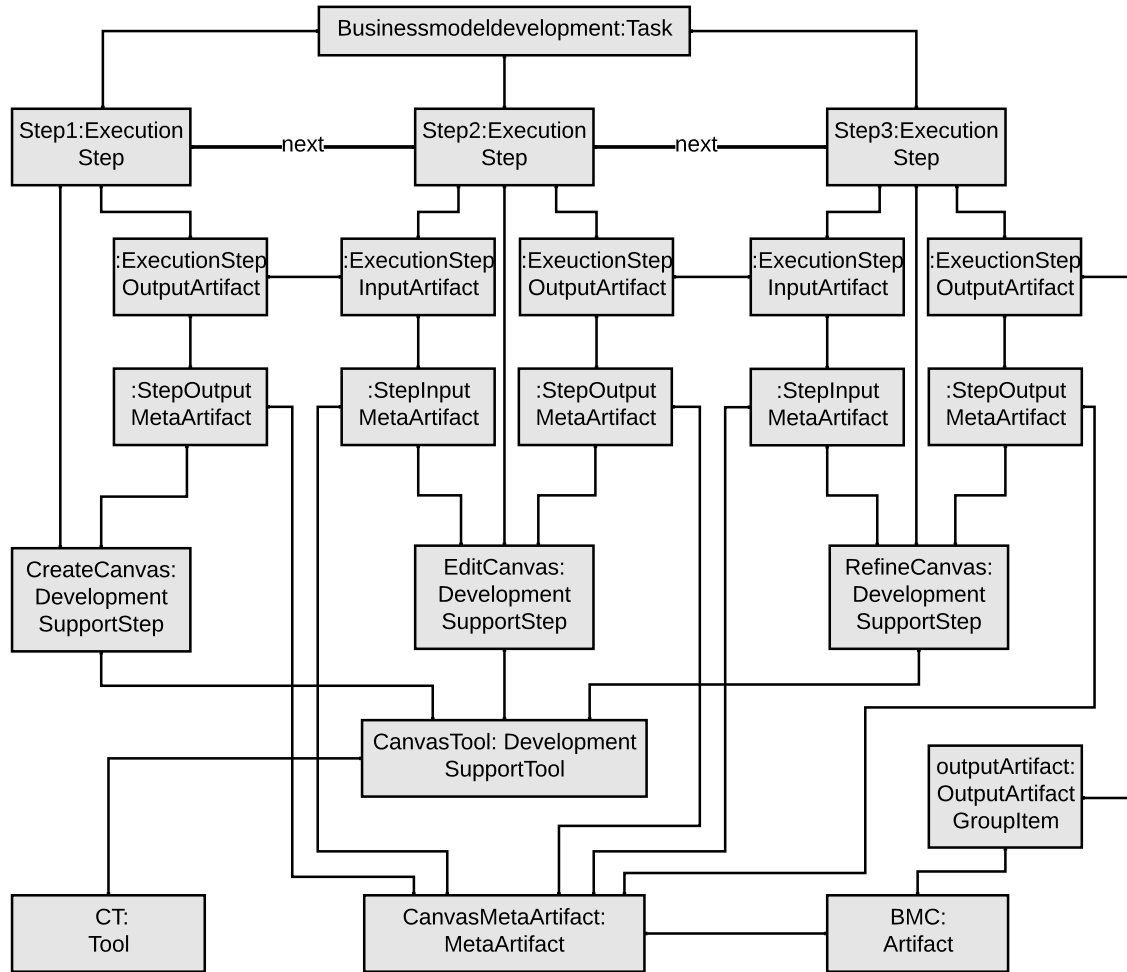


Fig. 7.7 Exemplary Method Building Block for the Modularization

we leave the whole flexibility to the *Development Support Developer* of the *Development Support Module* and the corresponding *Development Support Meta Artifact*. During the selection of the building block's tools and artifacts, the modeling artifact's quality needs to be controlled. Here, within our approach, we ensure that the default values for the input and output artifacts and that the connected tools are not deactivated, which would result in errors by enacting the development method.

One example here is the *Canvas Module*, which is explained in Figure 7.5. Here, the composition of the model is shown in Section 6.5. To connect the composition of the method to the composition of the models, we created a form as a hook for the *CreateCanvas* step where the *Method Engineer* can choose an existing composed model from a selection list or click on a button to compose a new one. By clicking this button, he is forwarded to a specific

part of the tool for the composition within the *Canvas Support Tool* and backward after he composed the models. After that, he can choose the composed model in the selection list.

After the *Method Engineer* has added the development support to the building blocks and composed the development method with the corresponding artifacts, the *Business Developer* can use the development support during the enactment.

7.2.3 Enactment of Development Support

After the *Development Support Steps* have been composed during the creation of the *Method Building Blocks* in the knowledge provision of methods and models, and additionally, for the *Development Support Artifacts*, in the composition of the development method, the *Business Developer* enacts those support in the enactment of the development method. To support the enactment, we have extended our *Process Engine*, as shown in Figure 7.8. Here, the *Execution Manager* is responsible for executing the different *Development Support Steps* of the *Development Support Tools*. Here, when the *Business Developer* starts to conduct a development step, the engine checks if the connected *Method Building Block* contains composed *Execution Steps*. If this is the case, the engine pipelines the *Business Developer* through the different *Execution Steps*. During this pipelining of the steps, it exchanges the current state of the *Development Support Artifacts* based on their *Development Support Meta Artifacts* with the *Artifact Manager*. Here, the created and/or modified *Development Support Artifacts* are stored temporarily during the execution of the steps and permanently after the whole development step. Moreover, the *Collaboration Manager* allows the communication of different *Stakeholders* during each step. Here, each step is connected to a discussion board to allow collaboration of the stakeholders and reason the information within the artifacts. Based on that extended engine, the *Business Developer* can conduct development steps and/or create (modularized) artifacts independently.

The **Conduction of Development Steps** can be done by selecting a predefined development step from the Kanban board or a flexible development step from the *Method Repository*. While the predefined development step is already configured, the flexible one needs to be configured by the *Business Developer* to use the corresponding *Development Support Tool*. If the development step contains execution steps, those are enacted by the *Execution Manager*. The manager then executed the steps after each other by receiving the hooks for execution from the *Development Support Tool*. Each hook contains a dedicated software component that should be integrated into the step management. Here, the software component can be freely designed by the *Development Support Developer* with the hook and interface constraints made by the *Meta-Development Support Developer*. This is done with the support of the source link to the tool and the identifier of the *Development Support Step*. Based on

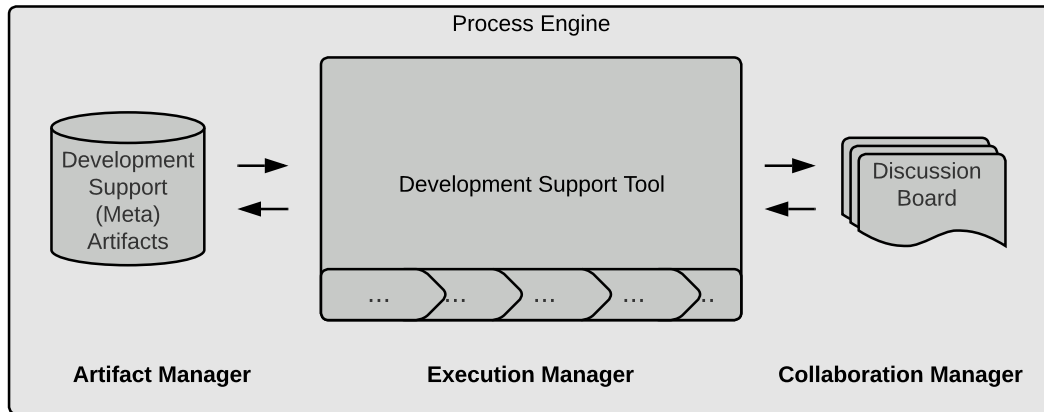


Fig. 7.8 Extension of the Process Engine for the Modularization

these constraints, the manager established the pipelining of artifacts between the different components. Here, every component might exchange data with the corresponding *Development Support Meta Artifact*, which is also designed as a separate component. Here, the identification is made with the source link to the meta artifact and the identifier of the artifact. Moreover, each meta artifact is self-responsible for its own storage management within the database, which increases the overall flexibility of the solution. Here, the components for the *Development Support Steps* and the components for the *Development Support Meta Artifacts* are completely free in their implementation as long as they provide the functionalities (i.e., interfaces, hooks) that are needed by the extended process execution engine (see also the modularization in Section 8.1 for all needed functionalities). Because the whole storage management of the *Development Support Meta Artifacts* is implemented internally, we need to provide some interfaces to get the name of an artifact, copy it, and remove it. During each execution step, a collaboration of different stakeholders is possible with the *Collaboration Manager*, and at the end, the created or modified artifacts are stored as references within the *Artifact Manager*.

One example here is the *Canvas Module* which communication of the *Development Support Tool* and the *Development Support Meta Artifact* can be seen in Figure 7.9. Here, we assume that we already created a *Business Model Canvas* as *Development Support Artifact* during the provision of development support. Based on that, the *Business Model Development* consists of the three execution steps of *CreateCanvas*, *EditEdit*, and *RefineCanvas*. Here, the *Canvas Tool* has a source link (i.e., *canvas/tool*) where a specific interface can be triggered with an identifier (e.g., *createcanvas*) to receive the specific component (e.g., *CreateCanvasStepComponent*) that can be used as a hook for the execution steps. During each step, the component communicates with the *Canvas Artifact Meta Component*, which

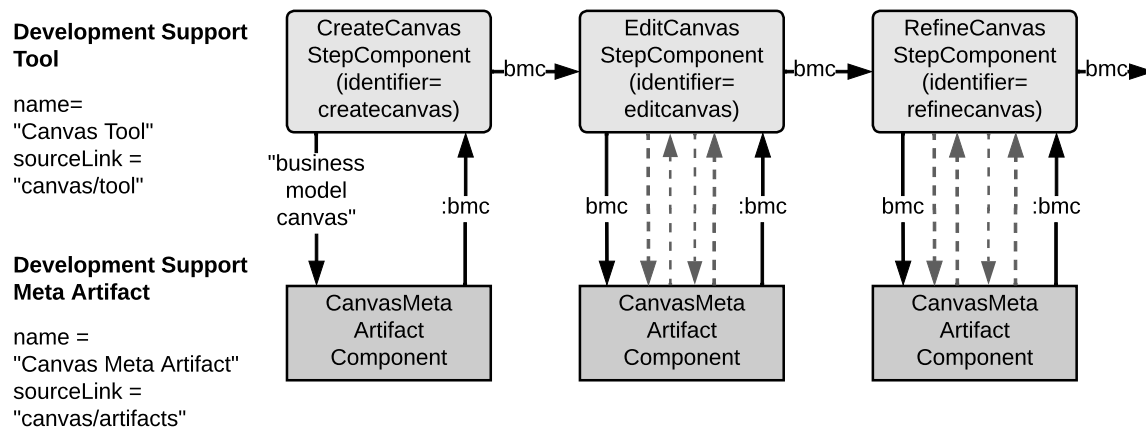


Fig. 7.9 Exemplary Usage of the Execution Manager for the Modularization

is identified by the link to the source (i.e., *canvas/artifacts*). While during the first step, the *Business Model Canvas* is selected as the default artifact (i.e., *business model canvas*) and the canvas is directly created (i.e., *bmc*), the other steps provide continuous communication between both components until each step is finished. As a result, the *Business Model Canvas* (i.e., *bmc*) is created as a *Modularized Output tArtifact*.

The **Creation of (Modularized) Artifacts** should also be done by the *Business Developer*, independently from the development steps, to allow flexibility in the BMD. For that, the solution provides the *Artifact Manager*, where existing artifacts can be viewed, updated, and deleted, and new artifacts can be created. Here, the *Business Developer* chooses the (*Modularized*) *Artifact* to create from the *Method Repository* in the *Artifacts Manager*. Depending on the type of artifact, the manager internally starts the *Information Artifact* creation process or the *Modularized Artifact* creation process from the *Development Support Meta Artifact*. Here, the source link and the identifier establish the connection between the artifact and the meta artifact. For that, each meta artifact needs to provide hooks for viewing, editing, and deleting created *Modularized Artifacts* together with an interface to delete them. Here, again, the *Development Support Developer* can decide if he wants to implement those functionalities within the *Development Support Meta Artifacts* or connects the functionalities from the *Development Support Tool*. Inside those hooks, communication between different stakeholders using the collaboration manager is also possible to support the development process.

One example here is our *Canvas Module*, where different canvas models should be creatable without the usage of a *Method Building Block*. Here, we provide the *Canvas Meta Artifact* with the hooks for creating, viewing, and updating specific *Canvas Artifacts* together with an interface to delete them. With this, the *Artifact Manager* can handle the independent

management. As we want to keep the meta artifact as small as possible, those hooks were implemented in the *Canvas Tool* and connected in the *Canvas Meta Artifact*. Moreover, while developing the *Canvas Module*, we take care to design a *Canvas Meta Artifact* that can be easily used by other modules, as canvas models are the de-facto standard in BMD.

Within the conduction of development steps and the creation of artifacts, the *Business Developer* can use different types of support modules that can be developed by the *Development Support Developers*.

7.3 Types of Support Modules

The development can be supported by different support modules for the different support forms (e.g., visualization, design, development) and various phases (e.g., discover, design, validate) of BMD. Here, we already classified different BMDSSs into modeling & configuration, analysis & simulation, and evolution & validation within our tool analysis conducted in Appendix A. In the thesis, as shown in Figure 7.10, we present different exemplary modules to show the overall applicability of our concept. Those are an integrated visualization- and design-support module (i.e., *Canvas Module*) that meta artifact is made for reusability in other modules, an internal decision-support module (i.e., *HypoMoMap Module*) that presents a specific solution for the model-based validation, and an external decision-support platform (i.e., *CPBV Platform*) that work independently of the solution. With those predefined modules, we ensure the out-of-the-box usage of development support within our solution for the *Method Engineer* and the *Business Developer*.

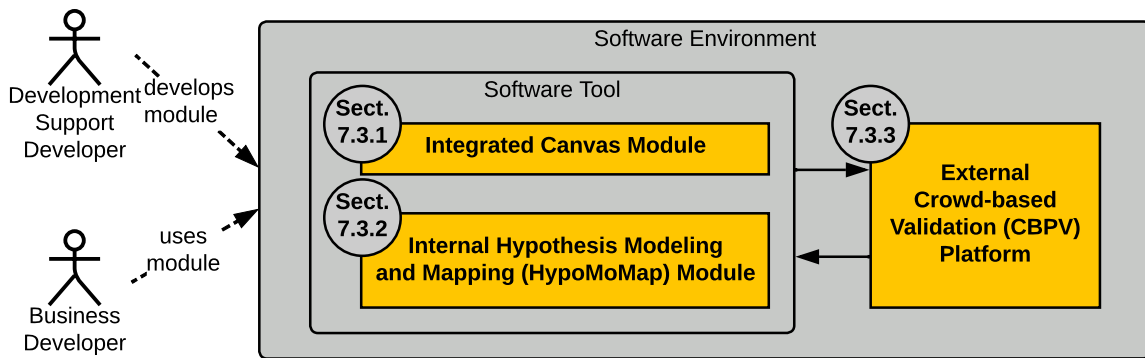


Fig. 7.10 Exemplary Modules for Development Support

In this section, we first show our integrated *Canvas Module* to design the business model based on the canvas model repository (7.3.1). Next, we present the internal *Hypothesis Modeling and Mapping Module* that supports validating business models and product features

(7.3.2). Last, also for the validation of business models and connected prototypes, we show the external *Crowd-based Prototype Validation Platform* (7.3.3).

7.3.1 Integrated Canvas Module

The integrated *Canvas Module* [GKE21, GYNE22a] is a development support for the design phase. Here, the idea is to provide guidance on different elements for different canvas models based on predefined knowledge in the existing *Canvas Model Repository*. For that, we based our approach on the hierarchical modeling of requirements engineering [DFH16], where different relationships between the requirements (e.g., one requirement supports or hurts another one) are modeled. We combine that with feature models [ABKS13], where also relationships between the features exist, and selections of those features are made. Based on that, we model the canvas elements of the canvas models by providing a structured derivation of the elements to fill out the canvas models from those hierarchies. Moreover, we provide additional functionalities, like the detection of patterns that are modeled as a subset of elements.

The support module can be directly used within our solution. For that, the *Method Engineer* creates different building blocks like *Business Model Development* or *Value Proposition Development* with the *Canvas Tool* as *Modularized Tool* and the corresponding *Modularized Output Artifacts* like *Business Model Canvas* or *Value Proposition Canvas*. Moreover, he defines the different *Support Steps*, like *Create Canvas*, *Edit Canvas*, *Refine Canvas* in those building blocks. In the following, we give an overview of the module, the provided meta artifact, and the provided tool. The implementation (see Section 8.2.2) and evaluation (see Sections 4.2, 9.1, 9.2) are shown in separate chapters.

Module Overview The *Canvas Module* consists of a development process, which is based on the creation of canvas artifacts in Section 6.3.2, and a visual notation, which is based on the creation of the canvas models in Section 5.3.3. Here, the development process is split up into the two partly interchanging phases of creation and refinement.

In the **Creation Phase**, the *Business Developer* fills out a *Canvas Artifacts* (e.g., *Value Proposition Canvas*, *Business Model Canvas*) for the first time. He can select the corresponding *Canvas Elements* from the *Canvas Building Blocks* or create new elements for the model. Here, he might use the existing guiding questions and examples as support for his ideation. Moreover, he can also use existing literature about possible business models like the Business Model Generation book [OP10] or about potential business model patterns like the Business Model Navigator pattern cards [GFC14] as input. For the best results during the refinement phase, he needs to look up an existing similar element in the repository with a

search functionality before creating a new one from scratch. The modeling of the repository is described in Section 5.3.3.

In the **Refinement Phase**, the *Business Developer* improves the filled-out *Canvas Artifacts* with the knowledge of the *Canvas Model Repository*. Based on that knowledge, he receives hints for possible strengths and weaknesses of his models, the search and identification of patterns, and the comparison against existing organizations. Moreover, he has the possibility to add competitors that models should be compared with his own designed one. For that, we provide the following guidance mechanisms:

- **Discovering Business Elements:** First, during the design of new business models, the knowledge is used as a library to discover possible business model elements that the *Business Developer* might use. By providing descriptions for all elements, the library ensures a common understanding between the *Business Developer* and other *Stakeholders*. Moreover, the module provides the functionality to check the designed business model against the defined constraints (modeled as requires relationship and excludes relationships), which supports the *Business Developer* in building effective business models.
- **Suggesting Business Patterns:** Next, the existing knowledge is also used to suggest possible business model improvements to the *Business Developer*. For this, the module provides functionalities to suggest possible business model patterns if parts of the patterns are already used in the business model. Moreover, it supports the analysis of strengths (modeled as support relationships) and weaknesses (modeled as hurt relationships). This supports the *Business Developer* in focusing on the most critical parts of the business model.
- **Comparing Business Models:** Finally, the *Business Developer* compares their designed business models with examples of the knowledge. Here, it is possible to directly choose competitors' business models to analyze competitive advantages by differences in the selected elements. Moreover, it is possible to search for similar existing business models in the whole library by matching the own selected elements and the elements of the organizations. These organizations, in turn, can be analyzed by the *Business Developer* to gather more insights into his own business.

Based on the results of the guidance, the *Business Developer* has the option to choose to improve the design of his canvas artifact or finish the design and start with the next development step.

Provided Meta Artifact The module is based on the *Canvas Meta Artifact*, where multiple *Canvas Artifacts* (e.g., BMC, VPC) are created dynamically by the *Method Engineer* during the knowledge provision. Here, he uses the existing *Canvas Model Repository* to create canvas models with their structure and possible relationships, as explained in Section 5.3.3. One exemplary canvas model where we provide design support is the *Business Model Canvas*, as shown in Figure 7.11.

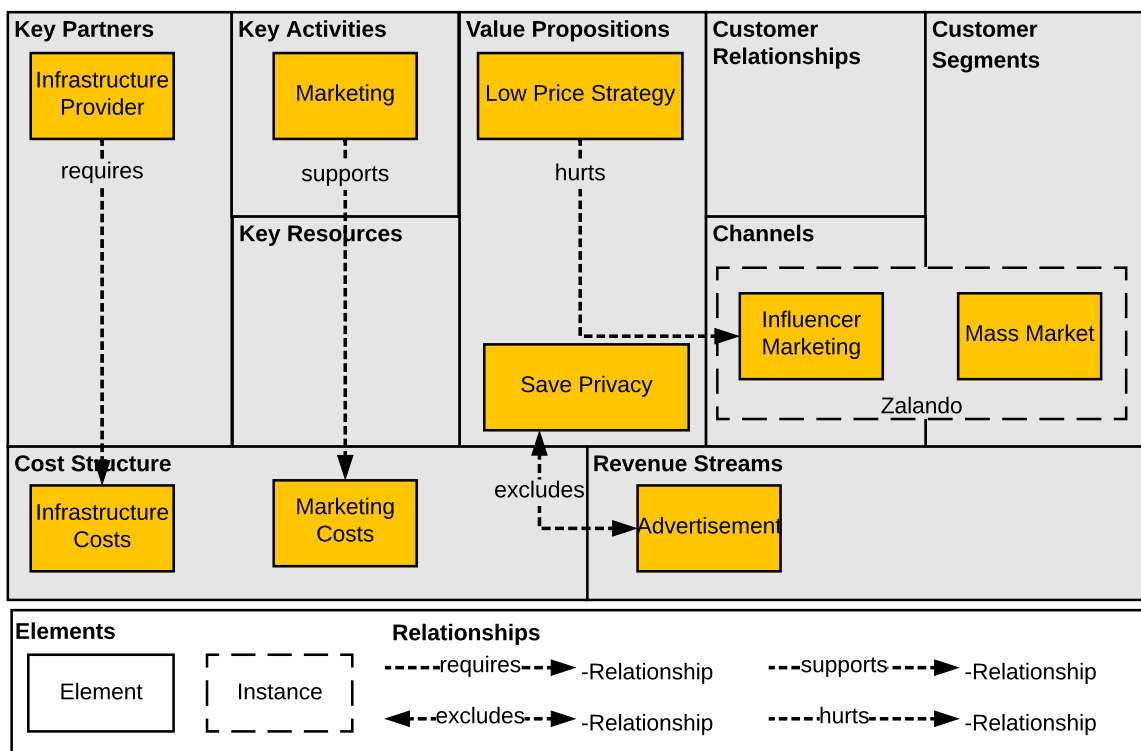


Fig. 7.11 Exemplary Visual Notation for the Canvas Artifact

The **Business Model Canvas** consists of the nine building blocks where *Elements* (e.g., *Infrastructure Provider*) are used to fill them. The *Method Engineer* provides additional information during the knowledge provision, see Section 5.3.3, that the *Business Developer* uses. Here, *Instances* group a set of elements to a pattern or organization (e.g., *Zalando* with *Mass Market* and *Influencer Marketing*). Moreover, hard constraints of *Requiring* (e.g., *Infrastructure Provider requires Infrastructure Costs*) and *Excluding* (e.g., *Save Privacy excludes Advertisement*) relationships might be defined. Last soft constraints of *Supporting* (e.g., *Marketing supports Marketing Costs*) and *Hurting* (e.g., *Low Price Strategy hurts Influencer Marketing*) relationships might be modeled.

Provided Tool To use the corresponding meta artifact, we provide the *Canvas Tool*. Here, the tool is able to create new and modify existing canvas models, providing guidance in the creation process and a comparison against other canvas models. To support the development, we provide a hook in the knowledge provision to add different canvas models as supporting artifacts together with creating canvas elements and canvas building blocks. While a hook creates the selection of the exact canvas model during the creation of method building blocks for the *Create Canvas Model* support step, the composition is supported by a hook for the model composition during the composition of the development method for the same step. Each support step, in turn, is supported by a single component as a hook during the enactment. To compose the *Method Building Blocks*, the *Method Engineer* chooses from the following *Development Support Steps*. Here, all support steps, excluding the initial creation, have a *Canvas Meta Artifact* as *Input Step Artifact* and *Output Step Artifact*:

- **Create Canvas:** The step creates an empty *Canvas Artifact* of the configured canvas model that should be filled with elements in the next support steps.
- **Edit Canvas:** The step is used to edit an existing *Canvas Artifact* by adding, modifying, and removing specific elements.
- **View Canvas:** The step is used to view an existing *Canvas Artifact* with its elements based on the configured canvas model.
- **Refine Canvas:** The step is used to refine an existing *Canvas Artifact* by using the hints and patterns from the composed model.
- **Create Competitors:** The step is used to create competitors on the existing *Canvas Artifact* that are used for a competitor analysis in the next support steps.
- **Edit Competitors:** The step is used to edit the existing *Canvas Artifact* by modifying or removing the created competitors.
- **Compare Competitors:** The step is used to compare the existing *Canvas Artifact* with the created competitors within a competitor analysis.

Apart from the integrated *Canvas Module*, the *Method Engineer* might also use the internal hypothesis modeling and mapping module to support the validation phase of the *Business Developer*.

7.3.2 Internal Hypothesis Modeling and Mapping Module

The internal *Hypothesis Modeling and Mapping (HypoMoMap) Module* [GYE20] is a development support for the validation phase. Here, the idea is to provide a validation of customer needs, including business models and product features, that are interpreted as hypotheses, which need to be validated or disapproved by conducting experiments with the customer [MWA19]. For that, we based our approach on goal-oriented requirements engineering that is used to structure requirements as interrelated goals [van01] that needs to be proofed over time. We transfer that concept to hypothesis engineering by modeling the hypothesis and experiments as hierarchies with a mapping between them to mark which hypothesis can be validated by which experiments. Moreover, we provide an approach to iteratively select the optimal experiments that should be conducted to validate the customer needs.

The module can be directly used within our solution. For that, the *Method Engineer* creates a *Hypothesis-based Validation* building block with the *HypoMoMap Tool* as *Modularized Tool* and the *HypoMoMap* as *Modularized Output Artifact*. Moreover, he defines the different *Support Steps*, like *Create HypoMoMap*, *Edit HypoMoMap*, *Execute Experiments* in those building blocks. In the following, we give an overview of the module, the provided meta artifact, and the provided tool. The implementation (see Section 8.2.2) and evaluation (see Section 4.2) are shown in separate chapters.

Module Overview The *HypoMoMap Module* consists of a development process, which is split up into the development support steps, and a visual notation, which is provided as a meta artifact that contains a single static artifact. Here, the development process is divided into the two partly interchanging phases of creation and validation.

In the **Creation Phase**, the *Business Developer* has to define hypotheses and experiments at the beginning. While the hypotheses of the business model and the product features can be derived from the business strategy and its product goals [OB14], the experiments can be chosen from existing libraries [BO20]. Based on this, he needs to create a visual notation of the *Hypotheses Lake* for modeling the hypothesis to validate and *Experimentation Island* for modeling the experiment to conduct, as exemplarily shown for the meta artifact in Figure 7.12. For this, the hypotheses and experiments are structured in interrelated hierarchies. Here, each hypothesis has to be decomposed into separately validatable units, which are connected with AND/OR relationships. With the term validatable units, we mean to split the hypotheses into small assumptions, which can be validated with strong evidence within an experiment. Moreover, the validation of these small assumptions can support the validation/disapproval of different hypotheses. While in the beginning, all hypotheses are labeled with the type untested with no estimated evidence, the *Business Developer* has to estimate the evidence and

costs of the experiments or use the predefined values of the *Method Engineer*. Those predefined values can be deposited during the knowledge provision. To support this process, he can use the experimentation catalog, which has been proposed in [BO20]. Moreover, he has to prioritize the essential hypothesis to consider at the beginning. After that, he needs to create the mapping of hypotheses to experiments where all hypotheses are mapped to experiments that can be used for validation. Here it is crucial to choose alternative experiments for each hypothesis so that different hypotheses can be validated/disapproved with the same experiment.

In the **Validation Phase**, the *Business Developer* has to choose and conduct experiments iteratively. Because this choice is a critical part, we provide three different techniques, which have different outcomes and are, therefore, used for different settings:

- **Highest Priority:** With this setting, a hypothesis with the highest priority is chosen. This setting is also used by other models [OB14, FSMM17] and ensures that the most important assumptions are validated first.
- **Best Estimated Ratio:** With this setting, an experiment with the maximum of hypotheses evidence gain, which can be validated in a single execution of the experiment in comparison to the costs of the experiment, is chosen. The hypotheses evidence gain is defined by the accumulated evidence scores between the current evidence score and the estimated evidence score with the experiment of all selected hypotheses. This setting is used to maximize the validated learning by also considering the costs.
- **Best Discounted Ratio:** Within this setting, the hypotheses evidence gained from single hypotheses in the best-estimated ratio is discounted with their priority. With this, a focus the validated learning on the most critical assumptions is established.

After conducting the selected experiment, the *Hypotheses Lake* and *Experimentation Island* have to be updated. For this, each tested hypothesis is typed as validated or disapproved with the evidence score of the experiment. Additionally, the evidence scores are propagated to the higher levels of the hierarchy. While with an OR-relationship, the higher hypothesis is set to the evidence score of the lower hypothesis, the AND-relationship assumes the lowest evidence score of all lower hypotheses. Moreover, new hypotheses that are derived from the experiment can be added to the model. In the end, the mapping of hypotheses to experiments needs to be updated. For this, all mappings between hypotheses and experiments which do not provide further evidence gain are removed. Moreover, the mappings between new hypotheses and experiments might be needed. The validation phase is repeated until there is no further experiment on the *Experimentation Island* that provides additional evidence for the hypotheses of the *Hypotheses Lake*.

Provided Meta Artifact The module is based on the *HypoMoMap Meta Artifact*, where, in contrast to the *Canvas Meta Artifact*, just a single *HypoMoMap Artifact* is statically created. Here, that artifact, which exemplary visual notation is shown in Figure 7.12, consists of a *Hypotheses Lake* and an *Experimentation Island*.

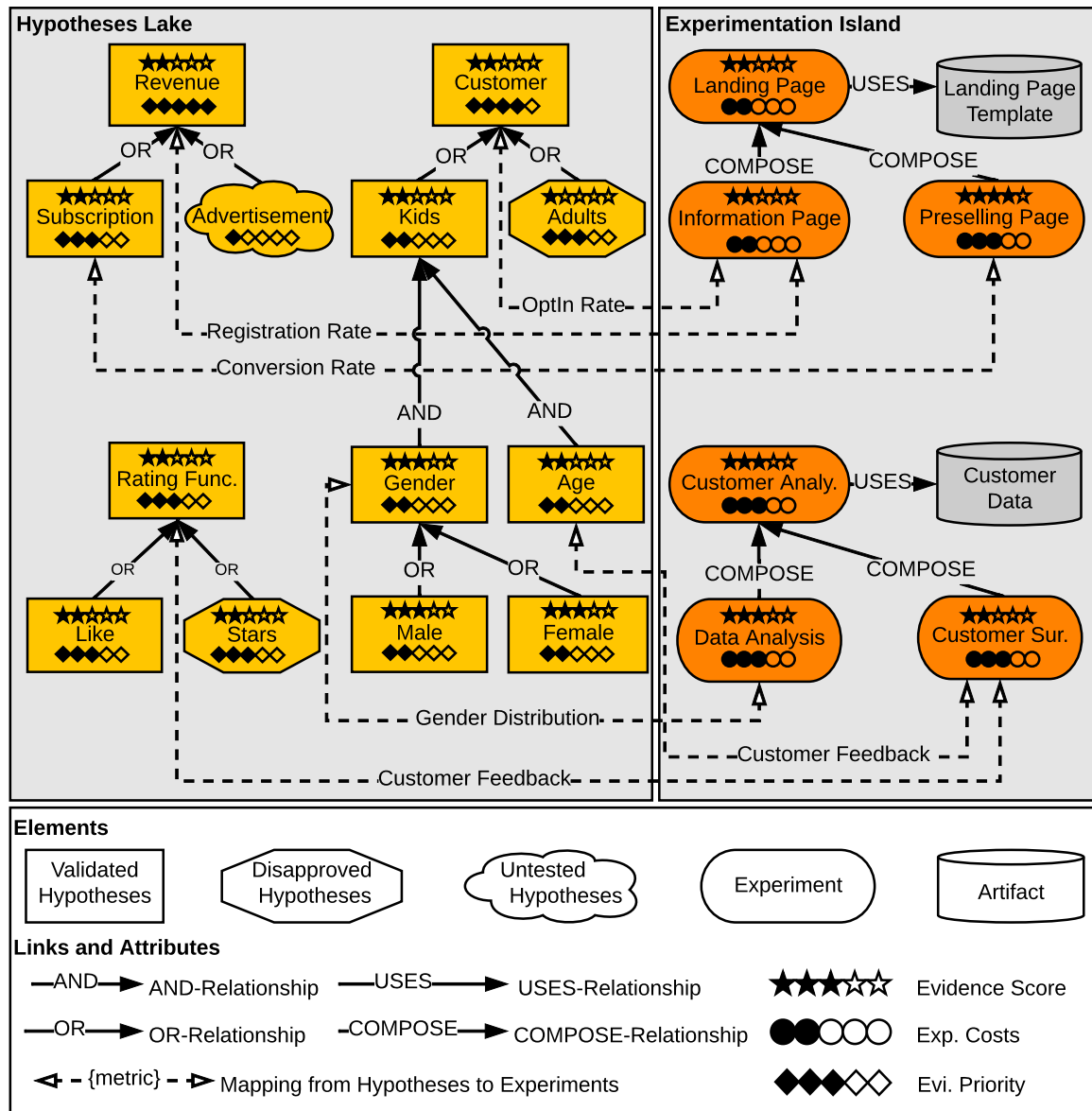


Fig. 7.12 Exemplary Visual Notation for the HypoMoMap Artifact

In the **Hypotheses Lake**, different elements of hypotheses (i.e., *Validated Hypotheses*, *Disapproved Hypotheses*, *Untested Hypotheses*) are modeled with corresponding estimated evidence (e.g., *Kids* with a score of 2) and a priority (e.g., *Kids* with a priority of 2). The hypotheses can interrelate in a hierarchical order similar to the *Canvas Building Blocks*

in Section 5.3.2 (e.g., *Customer* is decomposed into *Kids*). The hierarchy can be made with AND/OR relationships. While with an OR-relationship, each hypothesis in a higher hierarchy level can be validated with a single lower interrelated hypothesis (e.g., *Customer* or *Kids*), the AND-relationship provides only validation to a higher hypothesis by validating all lower interrelated hypotheses (e.g., *Kids* into *Age* and *Gender*). Moreover, each hypothesis is mapped to the experiments, which are used to validate or disprove the hypothesis (e.g., validate *Gender* with *Data Analysis*). For the mapping, we explicitly model the metric used to validate the hypothesis so that during the conduction of the experiments, different metrics might need to be measured (e.g., *Gender Distribution* for *Gender*).

On the **Experimentation Island**, different elements of experiments (e.g., *Data Analysis*) and corresponding artifacts (e.g., *Customer Data*) are chosen. Each experiment has estimated evidence and costs (e.g., *Data Analysis* with a score of 3 and costs of 3) and can use different provided artifacts. Moreover, the experiments can be decomposed into more accurate experiments of the same type (e.g., *Customer Analysis* to *Data Analysis* and *Customer Survey*).

Provided Tool To use the corresponding meta artifact, we provide the *HypoMapMap Tool*. Here, the tool is able to model and map the hypotheses and experiments as well as choose and conduct the experiments. To reduce the configuration time, we provide a hook in the knowledge provision to add experiments and artifacts in a structured form by the *Method Engineer* so that they can be used for different validation steps by the *Business Developer*. To compose the *Method Building Blocks*, the *Method Engineer* can choose from the following *Development Support Steps*. Here, all support steps, excluding the initial creation, have a *HypoMoMap Meta Artifact* as *Input Step Artifact* and *Output Step Artifact*:

- **Create HypoMoMap:** The step creates an empty *HypoMoMap Artifact* that should be filled with hypotheses and experiments during the next support steps.
- **Edit HypoMoMap:** The step is used to edit an existing *HypoMoMap Artifact* by adding, modifying, and removing specific hypotheses and experiments.
- **View HypoMoMap:** The step is used to view an existing *HypoMoMap Artifact* with its executed experiments and the validation state of the hypotheses.
- **Add Hypotheses:** The step is used to add new hypotheses to an existing *HypoMoMap Artifact* that should be validated within the experiments.
- **Add Experiments:** The step is used to add new experiments to an existing *HypoMoMap Artifact* and create a mapping to the existing hypotheses.

- **Execute Experiments:** The step is used to conduct certain experiments from an existing *HypoMoMap Artifact* by providing different selection options for the experiments.

Instead of the internal *HypoMoMap Module*, the *Method Engineer* might also use the external crowd-based prototype validation platform to support the validation phase of the *Business Developer*.

7.3.3 External Crowd-based Prototype Validation Platform

The external *Crowd-based Prototype Validation (CBPV) Platform* [GAYE21, GPYE22] is the largest development support for the validation phase that we have created within our thesis. Here, the idea is to use aggregated user feedback that can be provided iteratively on software prototypes, including product features and the business model, before the development to judge the idea behind the product and save development resources [LM16]. For that, we based our solution on crowdsourcing techniques [Lei12] that can be used to collect feedback from many potential users. Here, the collaborative intelligence of many potential users is able to reduce the bias of the single business developer [BS07] and therefore support the product validation. The platform is an external software artifact that is not directly integrated as an *Development Support Module*. However, the *Method Engineer* can create a *Crowd-based Validation* building block with the *CBPV Platform* as *Tool*, and *Prototype Information* as possible *Output Information Artifact*.

To design the platform, we conducted a separate DSR study with three design cycles to develop a platform for software developers to support the prototype validation process using the crowd. We evaluated the third cycle of the thesis and the second cycle of the platform in the same user study as presented in Section 9.2. Within the platform study, we provided abstracted design knowledge in the form of design principles and an overall solution concept, together with a situated implementation of design features and a software artifact. While in the following, we present our underlying research approach, the derived design principles and the conceptual solution design, the situated implementation (see Section 8.2.2), and evaluation (see Section 9.2) are shown in separate chapters.

Research Approach As a research approach, we use DSR to gain abstracted design knowledge about the crowd-validation of software prototypes that different software designers can implement into their new and existing software tools. With this study, we aim to answer the RQ of how to design platforms that integrate crowdsourcing techniques in the iterative validation of prototypes. For DSR, we use the cycle of Kuechler and Vaishnavi [KV08] that is also used within the overall research approach of our thesis as presented in Section 1.2.

Within this study, we aim to solve the problem of crowd-validation of mobile application prototypes but also ensure that they can be generalized to related application areas. For this, we use design principles (DP) to codify the knowledge in a transferable way [GKS20]. Moreover, we base our DSR on the opportunity creation theory (OCT) [ABA13], as also used in our overall research approach, as kernel theory to stick in line with similar approaches like business model validation [DLE17] or venture ideation [Vog17] from digital entrepreneurship [Nam17]. The cycle, as shown in Figure 7.13, consists of the following five iteratively conducted steps. First, we identify the *(1) Awareness of [the] Problem* based on a real-world problem and provide a *(2) Suggestion* of a possible solution. Next, we work on the *(3) Development* of the software artifact and conduct an *(4) Evaluation* of it. Based on the evaluation results, another iteration is conducted, and/or our research contributions as *(5) Conclusion* are provided. The contribution of our study, compared to the contribution types in Figure 1.6, are the nascent design theory of our design principles (Level 2) together with the situation implementation of our software platform (Level 1).

DSR Cycle	First Cycle	Second Cycle	Third Cycle
Awareness of Problem	Deriving Design Requirements (DRs) from Literature Review and Tool Analysis	Revisiting DRs from Lessons Learned and additional Literature	Revisiting DRs from Lessons Learned
Suggestion	Conceptualizing of Design Principles (DPs) based on Theoretical and Empirical Findings	Revisiting of DPs and Concept	Revisiting of DPs and Concept
Development	Formation of Design Principles to Design Features (DFs) and Instantiation as Prototype	Revisiting of DFs and Reinstantiation as Prototype	Revisiting of DFs and Improvement of Prototype
Evaluation	Evaluation with Expert Workshop (n=6) and Questionnaire	Evaluation with User Study (n=14) and Questionnaire	Evaluation with User Study (n=26) and Questionnaire
Conclusion			Communication of Design Knowledge and Prototype

Fig. 7.13 DSR Process for the CPBV Platform

In the **First Cycle**, we got aware of the problem by conducting a literature review and tool analysis in the application areas of lean development, UI prototyping, and crowdsourcing to derive initial design requirements (DRs) for our approach. Based on mapping the theoretical

and empirical DRs, we suggested our first design principles (DPs) together with a preliminary concept. Out of that DPs we developed the first design features (DFs) and instantiated them in a software prototype. Last, we evaluated them in an online expert workshop (n=6), where we explained the overall concept, showed the software platform, and asked for feedback. Subsequently, we gave the experts access to the platform. We sent out a questionnaire to rate the importance of the DPs and provide feedback on the overall idea, the proposed solution, the current drawbacks of the platform, and additional feedback.

In the **Second Cycle**, we took the lessons learned from the expert workshop together with additional literature to revisit the underlying DRs. Based on that, we also revisited our DPs and the suggested concept. This lead also to a redevelopment of the DFs and a complete new instantiation of the software prototype. We evaluated the DPs and the prototype in a student seminar on the lean development of mobile applications. Here, the students (n=14) were divided into different groups (g=6) to develop an idea for an app within the seminar iteratively. Here, one student per group needed to upload their prototype with questions to the platform. Next, every student gave feedback on two predetermined prototypes by answering the questions that could be used to improve the prototypes. Last, the students evaluated the prototype of the platform on the platform itself by rating the importance of the DPs together with feedback on the overall idea, the proposed solution, the current drawbacks of the platform, and additional feedback.

In the **Third Cycle**, which results are also shown within this thesis, we took the lessons learned from the user study to revisit our DRs. Out of that, we improved the DPs and the overall solution concept. Moreover, we improved the DFs and the existing software platform based on those changes. We evaluated the DPs and the prototype similar to the second cycle in a student lecture for the systematic development of AR/VR applications. Within the lecture, the students (n=26) had a mini project where they needed to develop such an AR/VR application in a group (g=8). Again, one student needed to upload the prototype, each student needed to evaluate two predefined prototypes, and all students needed to evaluate the DPs together with the platform.

Derived Design Principles We codify our abstracted design knowledge during the design science study within the DPs. Here, each DP shows a certain aspect of the platform and is based on the revisited DRs during the three cycles. Here, those DRs were derived from a literature review and tool analysis on the topics of lean development, UI prototyping, and crowdsourcing. In the following, we show the nine DPs together with references to literature and tools that build the foundation for the mapped DRs.

DP 1: User Variety states that *the solution should provide functions for integrating different internal and external users (e.g., platform user, crowd worker) to allow developers to participate with a heterogeneous group of users within the validation process*. In literature, this is reasoned by the fact that developers in early product development have high uncertainties that can be validated by testing the underlying assumptions [Bla13]. By using the knowledge of a crowd, the assumptions can be proven [MNJR16], and the biases of the developers can be reduced [BS07]. Here, the users can come from internal sources like employees or external sources like Amazon Mechanical Turk¹.

DP 2: Task Iteration states that *the solution should provide functions for conducting tasks iteratively to allow developers an incremental improvement of the prototypes over time*. In literature, this is reasoned by the fact that user feedback could support the adjustment of product features [Rie14] and the business model [McG10] to the market. For that, that feedback can be provided by a crowd of users like with ClickWorker² where the rapidness of the given feedback is a critical factor of success [Bla13].

DP 3: Prototype Diversity states that *the solution should provide functions for integrating different types of prototyping (e.g., mockups, click dummies) to allow developers a flexible choice for their current validation developments*. In literature, this is reasoned by the fact that depending on the stage of the product development, also different prototypes like textual descriptions, images, or click dummies can be used [LCK⁺11]. Here, different prototypes can ensure the refinement of the product features or business model over time [OB15]. For the visualizations, also external tools like Figma³ can be used.

DP 4: Feedback Diversity states that *the solution should provide functions for integrating different types of feedback (e.g., free texts, ratings) to allow developers a flexible choice for their current validation challenges*. In literature, this is reasoned by the fact that depending on the type of the development stage also, different types of feedback are necessary [BO20]. Depending on the type of test, that feedback can consist of qualitative or quantitative information [OB15]. Different types of feedback are also integrated within the prototyping tool of UIGiants⁴.

DP 5: Filter Mechanisms states that *the solution should provide functions for the filtering between users and tasks to allow developers and users to shortlist evaluations based on specific criteria (e.g., skill set, interests)*. In literature, this is reasoned by the fact that to ensure the quality of the feedback, the tasks must be just conducted by users of a relevant target group of the developer [MYW⁺15]. Conversely, users should see only tasks in which

¹Website of Amazon Mechanical Turk: <https://www.mturk.com>

²Website of ClickWorker: <https://clickworker.com>

³Website of Figma: <https://www.figma.com>

⁴Website of UIGiants: <https://www.uigiants.com>

they are interested [KCS08]. Amazon Mechanical Turk also uses two-sided filtering between the task provider and the crowd worker.

DP 6: Aggregation Mechanisms states that *the solution should provide functions for aggregating and visualizing the feedback to allow developers to provide understandable and traceable improvements to the prototypes*. In literature, this is reasoned by the fact that depending on the number of individual user feedback, it could be a time-consuming and challenging activity to process them. Here, the feedback should be provided to the developer in an aggregated form for fast processing [GSS⁺11]. Moreover, appropriate visualizations should support the developers in interpreting the feedback [XHB14]. ClickWorker also aggregates the results of conducted tasks from different crowd workers.

DP 7: Incentive Mechanisms states that *the solution should provide functions for supporting extrinsic and intrinsic incentives (e.g., rank lists, money) to allow developers to motivate users in the validation process*. In literature, this is reasoned by the fact that giving valuable feedback is time-consuming and should ideally be done regularly [Rie14]. Therefore, users should be offered extrinsic incentives like money or intrinsic incentives like fame [HH12]. While money is used as an extrinsic incentive by Amazon Mechanical Turk, intrinsic incentives like ratings and views are used by social media platforms like YouTube⁵.

DP 8: Non-Disclosure Mechanisms states that *the solution should provide functions for integrating non-disclosure agreements to allow developers to protect their prototypes from user thefts*. In literature, this is reasoned by the fact that developing new ideas is a creative and challenging activity that often needs the collaboration of various stakeholders [EBL16]. Depending on the trust between the developers and the users, non-disclosure agreements can be necessary for a more intensive idea exchange [FCP12]. Those agreements are also often requested by clients on projects with a larger volume on the micro job platform Fiverr⁶.

DP 9: Governance Mechanisms states that *the solution should provide functions for integrating governance into the process to allow the platform owner to take necessary actions against developers and users that misuse the validation process*. In literature, this is reasoned by the fact that providing valuable interactions between the developers and the users is the key task for the platform to stay successful. Good governance of those interactions will let the users stick much longer on the platform [ES16]. Here, governance in terms of policies, regulations, and accountability should be provided by the platform [PvC16]. This, in turn, must be implemented in nearly every platform like Innocentive⁷, which aims to solve problems.

⁵Website of YouTube: <https://www.youtube.com/>

⁶Website of Fiverr: <https://www.fiverr.com>

⁷Website of Innocentive: <https://www.innocentive.com>

Conceptualized Solution Design Out of the codified DPs for the abstracted design knowledge, we conceptualize an overall solution design of the CBPV platform as visualized in Figure 7.14. It consists of the three roles of the *Developer*, the *User*, and the *Platform Owner* and the five components of the *Task Creation*, the *Task Conduction*, the *Task Evaluation*, the *Task Incentivisation*, and the *Task Approval*.

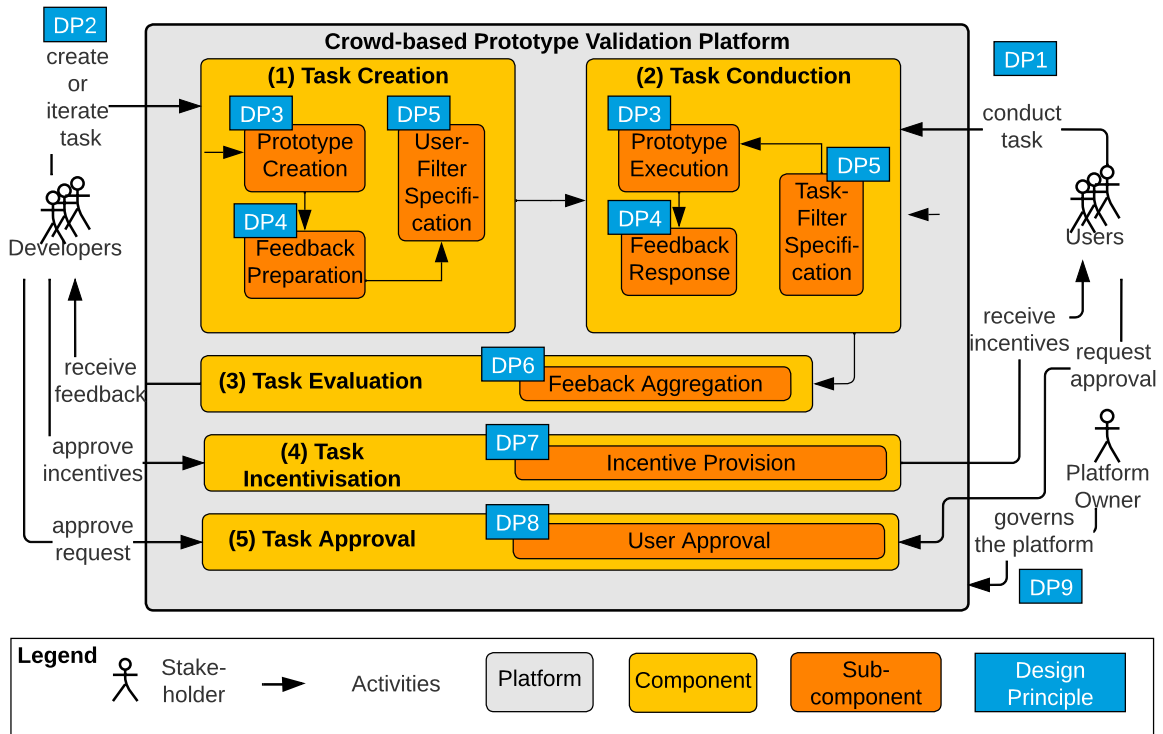


Fig. 7.14 Solution Design for the CPBV Platform

In the beginning, different *Developers* and *Users* (i.e., DP 1) register to the platform, each with a specific profile of their skills. After that, the *Developer* creates a new or iterates an existing task (i.e., DP 2) in the *Task Creation* by creating different types of prototypes (i.e., DP 3), preparing different types of questions (i.e., DP 4), and selecting specific criteria for users (i.e., DP 5). Next, the *User* selects different tasks (i.e., DP 5) in the *Task Evaluation*, depending on the approval process, executes the prototype (i.e., DP 3), and provides feedback (i.e., DP 4). This feedback is aggregated and visualized (i.e., DP 6) in the *Task Evaluation* and displayed to the *Developer*. Based on that, the *Developer* can provide intrinsic and extrinsic incentives (i.e., DP 7) to the *Users* in the *Task Incentivisation*. Moreover, the *Developer* can decide on an automatic selection of access to the prototypes with or without the usage of a non-disclosure agreement (i.e., DP 8) in the *Task Approval*. Here, a manual selection is possible where the users ask for approval and get access to the prototype. Last,

the *Platform Owner* governs the whole platform against misuse (i.e., DP 9) by moderating developers/users and tasks.

7.4 Summary

Within this chapter, we have provided the conceptual solution for our third stage in support of the development steps. For that, we have shown the modeling and integration of support modules together with their construction and execution for concrete development steps. Moreover, we have provided three exemplary modules that can be used to support the development of business models.

For the **Development Support Modularization**, we have defined an extension for the method elements to allow the atomic modeling of support tools and meta artifacts. Based on that, we have extended the method building blocks to allow the composition of development support steps for a single building block. Last, we have shown the enactment of that development support based on the execution of those support steps in those building blocks.

For the **Module Types**, we have provided the integrated canvas module to allow support in the design of canvas models based on the canvas model repository. Moreover, we have presented the internal HypoMoMap module that allows the continuous validation of assumptions about the business model and the product features. Last, the external CBPV platform allows the continuous validation of different prototypes with feedback from the crowd.

Based on the support of development steps and the other two previous stages, we will show the situated implementation of our solution within the next chapter. Here, we will show our underlying modularized architecture with a general overview and a focus on the usage of the support modules. Moreover, we will show our implemented software tool together with the developed support modules.

Part III

Implementation, Evaluation, Conclusion

Chapter 8

Implementation

In the previous chapters, we showed a conceptual overview together with detailed instructions about all three stages of our approach. Based on that, this chapter presents the modularized architecture for our solution together with an implemented software tool. For that, we first introduce our modularized architecture on which the software tool relies, together with the technologies used (8.1). Based on that, we present our software tool together with the implemented support modules (8.2). Last, we summarize the results of both (8.3).

8.1 Modularized Architecture

In this thesis, we have proposed a situation-specific BMD approach in the application domain of software ecosystems. For that, we have presented a solution with the three stages of knowledge provision of methods and models, composition and enactment of development methods, and support of development steps. Based on that, we designed a modularized architecture for a software tool that covers all those stages. Inside that architecture, we focus on the extensibility by using different development support modules.

In this section, we first introduce our designed architecture with the integration of the development support modules (8.1.1). Based on that, we present the applied technologies to the architecture to develop the software tool (8.1.2).

8.1.1 Designed Architecture

The designed architecture is used to structure the development of the software tool. An overview of that *Modularized Architecture* is presented in Figure 8.1. Here, we have the main components of the *Knowledge Provision Manager*, which is responsible for managing the knowledge of the methods and the different meta artifacts, the *Development Method*

Composer, which is responsible for composing the development method based on the method and the connected meta artifacts, and the *Development Method Enactor*, which is responsible for enacting the development methods and conducting of development steps to create artifacts. All main components are connected to the *Development Support Modules*, which are responsible for providing development support for certain development steps. The component diagram of our modularized architecture, which is explained in more detail in the next paragraphs, can be seen in Figure 8.2.

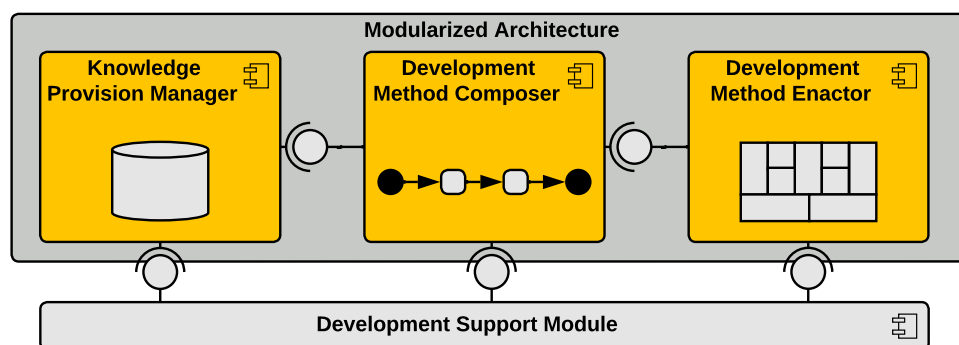


Fig. 8.1 Overview of the Modularized Architecture

In the **Knowledge Provision Manager**, we have the *Method Provider*, which is responsible for creating the method repository, as shown in Section 5.2. Within the method provider, the method elements, method building blocks, and method patterns are created. Moreover, the provider composes the development support steps by using the *Support Step Information Interface* and additional *Support Step Configuration Hooks* from the *Development Support Tools* as presented in Section 7.2.2. With those steps, the provider interacts with instances of the *Development Support Meta Artifact* that are provided by the *Meta Artifact Information Interface*, as explained in Section 7.2.1. To use those support tools and meta artifacts of the *Modules*, the *Module Provider* contains a *Module Registry* to integrate different *Development Support Modules* in the software tool. The meta artifacts are also able to provide an *Artifact Provision Hook*, which is enabled by the *Meta Artifact Provider*. One example of such an artifact provider would be the creation of the canvas model repository, as shown in Section 5.3. Here, within that specific artifact provider, the canvas elements, canvas building blocks, and canvas models are created.

In the **Development Method Composer**, we used the *Provided Knowledge* to compose the development method. For that, the *Context Manager* provides the *Situational Factors* and the *Application Domains*, as explained in Section 6.2.1. The factors are used by the *Method Composer* to construct the methods as explained in Section 6.2.2. During the method construction, the different method patterns are nested for the pattern-based construction, or

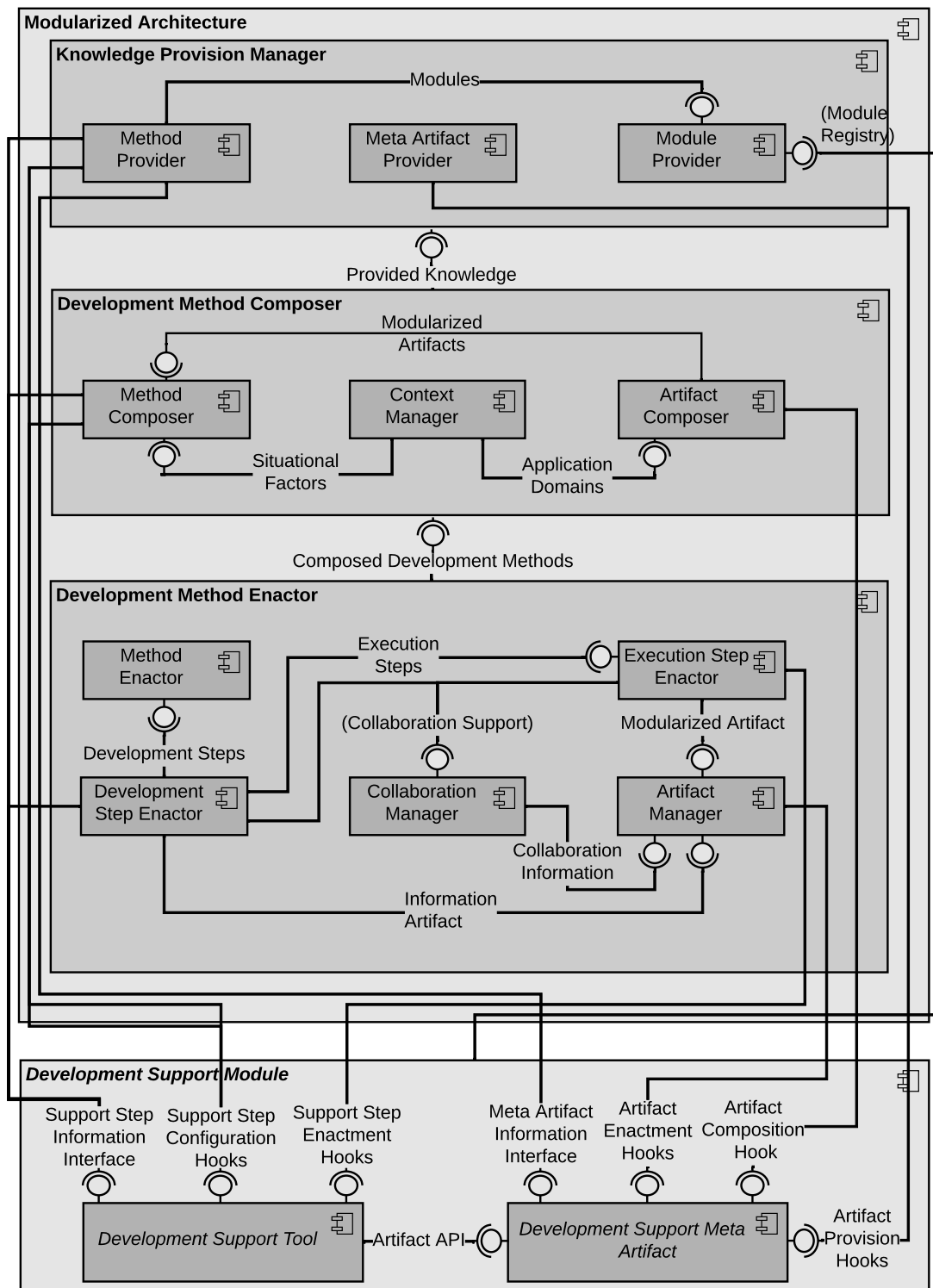


Fig. 8.2 Component Diagram of the Modularized Architecture

the phases are filled for the phase-based construction. To allow development support, as presented in Section 7.2.2, the *Support Step Information Interface* provides corresponding execution information about the steps, and the *Support Step Configuration Hooks* are used for additional configurations of steps with parameters. *Modularized Artifacts* are enhanced with additional knowledge using the domain and the *Artifact Composer* with an *Artifact Composition Hook* from the meta artifact. One example of such an *Artifact Composer* would be the composition of canvas models, as shown in Section 6.2.3. Those canvas models are constructed using feature-based or taxonomy-based consolidation. Last, the *Context Manager* can also be used to change the context and modify the composed methods and models under the usage of the other composer, as presented in Section 6.3.3.

In the **Development Method Enactor**, we used the *Composed Development Methods* to develop the business models. For that, the *Method Enactor* executes the development method as presented in Section 6.3.1. Within the method enactor, the development method is visualized using the nested method patterns or the different phases. The *Development Step Enactor* takes the *Development Steps* and, based on the development support, creates the *Information Artifacts*, as presented in Section 6.3.2, or uses the *Support Step Information Interface* to split the atomic *Execution Steps*, as shown in Section 7.2.3. By creating the *Information Artifacts*, it uses the *Collaboration Support* from the *Collaboration Manager* explained in 6.3.2. The *Execution Step Enactor* takes those *Execution Steps* and executes them with the corresponding *Support Step Execution Hooks*, presented in 7.2.3. One example of such execution hooks are the creation, editing, and refinement of canvas models, as explained in Section 7.2.3. Here, also *Collaboration Support* is used for each step. Last, those *Modularized Artifacts* are stored within the *Artifact Manager*. Here, each meta artifact provides different *Artifact Enactment Hooks* to create, read, update and delete those artifacts. Moreover, the *Collaboration Information* is stored with each artifact.

To support the creation and modification of artifacts by the *Development Support Tool*, each *Development Support Meta Artifact* provides an *Artifact API*. If the *Meta Artifact* should also be used by other *Development Support Developers*, that API should be explained in detail by the developer. Based on that modularized architecture, we applied different technologies to support and efficient development of the software tool.

8.1.2 Applied Technologies

To implement the software tool based on our designed architecture, we use different web technologies to increase the quality of our tool and support an easy extension in the future. Those are the *Angular Framework* and *Bootstrap* for the frontend, *PouchDB* and *CouchDB*

as databases, and the *BPMN.js Library*, the *Quill Editor* and the *JSON Schema Vocabulary* as additional libraries.

For the **Frontend**, we based the whole architectural logic on *Angular*¹. Here, Angular is a framework for building single-page client applications based on HTML for the visual representation and TypeScript for the control logic. TypeScript² is a strongly typed programming language based on JavaScript. This allows developers the fast and secure development of web-based applications. The main parts of an Angular application can be seen in Figure 8.3³. Here, those parts can be divided into modules, components, templates, services, and directives.

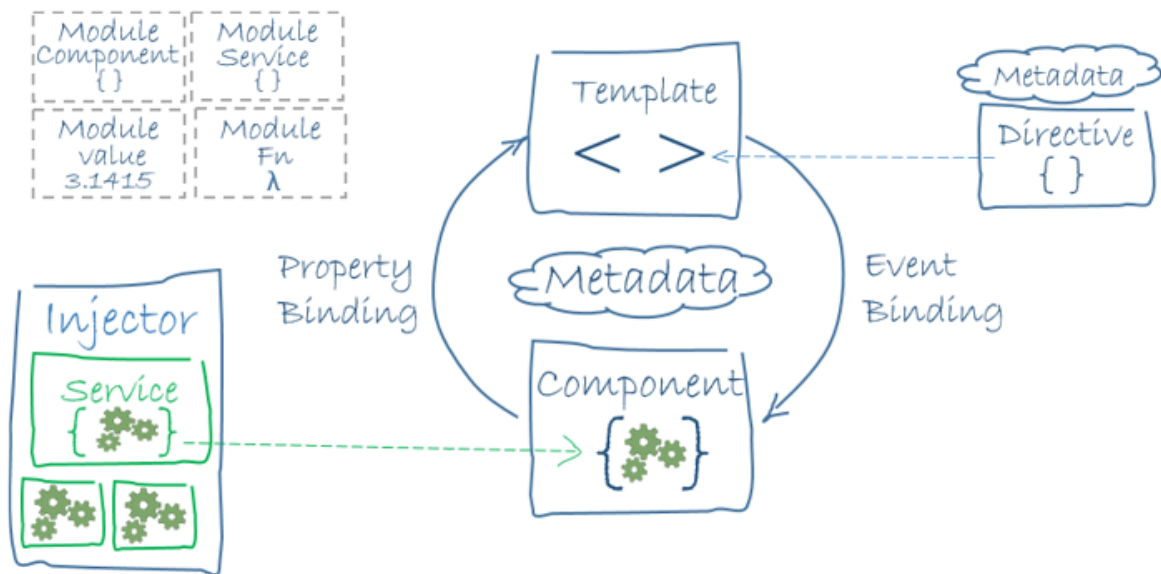


Fig. 8.3 Main Parts of an Angular Application

The architecture of Angular is split up into different *Modules*, where each module contains a subset of functions to achieve a certain goal. Those modules, in turn, can be imported and exported among different Angular applications. Here, each application has a root module that is started during the launch of the application and imports other modules to provide additional functionalities. Inside those modules, *Components* provide an application logic for the application with customized functionalities. Overall, the root module needs to provide an initial root component that can be triggered during the application launch.

The root module of our situated implementation with the imported components is shown in Listing 8.1. Here, we import the standard modules of Angular (e.g., *NgModule*), modules

¹Website of Angular: <https://angular.io/>

²Website of TypeScript: <https://www.typescriptlang.org/>

³Architecture of Angular: <https://angular.io/guide/architecture>

for the additional libraries (e.g., *QuillModule*), and used components (e.g., *ToolExplanationComponent*). For the situated implementation of the *Development Support Modules*, we used separate Angular modules for the *Development Support Tool* (e.g., *CanvasModule*) and the *Development Support Meta Artifact* (e.g., *CanvasMetaArtifactModule*) to allow a more flexible combination of meta artifacts with different support tools. Moreover, within our solution, we provide a generation of code stubs for the support tools and the meta artifacts. After installing our software tool as explained in Appendix B, the command of *ng generate ./module-schematics:tool-module* can be run the command line for creating a support module and the *ng generate ./module-schematics:meta-artifact-module* for a meta artifact. Both commands will start a configuration process based on Angular Schematics⁴. Here, Angular Schematics is a template-based code generator integrated into the Angular CLI to improve the coding efficiency for similar reusable code. While inside this section, we show the basic functionalities of those generated hooks, a technical documentation is provided together with the source code of our software tool. During the launch of the software tool, the *AppComponent* is started as the root component. Here, all modules, including the one for particular development support, are imported to the root module so the modularized architecture can access them. Moreover, all modules have access to a separate database service for the PouchDB to store their data without limiting constraints.

```
1 import { NgModule } from '@angular/core';
2 import { QuillModule } from 'ngx-quill';
3 import { CanvasMetaArtifactModule } from './module/canvas/canvas-meta-
  -artifact/canvas-meta-artifact.module';
4 import { CanvasModule } from './module/canvas/canvas/canvas.module';
5 import { AppComponent } from './app.component';
6 import { ToolExplanationComponent } from './tool-explanation/tool-
  -explanation.component';
7 // ...
8
9 @NgModule({
10   imports: [
11     NgModule,
12     QuillModule.forRoot({
13       format: 'json',
14     }),
15
16     CanvasMetaArtifactModule,
17     CanvasToolModule,
18     // ...
```

⁴Website of Angular Schematics: <https://angular.io/guide/schematics>


```
19     ],
20     declarations: [
21         AppComponent,
22         ToolExplanationComponent,
23         //...
24     ],
25     providers: [],
26     bootstrap: [AppComponent],
27 })
28 export class AppModule {}
```

Listing 8.1 Root Module for the Modularized Architecture

Those functionalities are connections by interactions with a template. Here, the *Template* is an HTML file with additional Angular markup code, which allows a file modification before its rendering. Those templates can also be designed using Bootstrap. Bootstrap⁵ is an open-source CSS framework for responsive user interfaces of websites. Components and templates can be connected in both directions using property binding and event binding. Here, Listing 8.2 shows a part of our *AppComponent*, where we include the *app.component.html* as a template. Moreover, both are connected through a binding so that a logout command in the template will trigger the *authService* to log out the user.

```
1 import { Component } from '@angular/core';
2 // ...
3 @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css'],
7 })
8 export class AppComponent {
9     constructor(
10         private authService: AuthService,
11         //...
12     ) { // ...}
13
14     logout(): void {
15         this.authService.logout().subscribe();
16     }
17
18     //...
19 }
```

Listing 8.2 Root Component for the Modularized Architecture

⁵Website of Bootstrap: <https://getbootstrap.com/>

Those rendering is supported dynamically by using directives. Here, Directives access the document object model as a bridge for the HTML structure by using Create-Read-Update-Delete (CRUD) operations on the elements. Last, Services enable the integration of additional functionalities that could be used by multiple components. Those services are injected into the components that should use the functionalities. Within Angular, we use services to provide correct execution steps for the development support and use the internal navigation component to include components such as hooks. An example of the canvas tool is shown in Listing 8.3. Here, the *CanvasService* creates and registers a new *Module* with a name, used methods, the reference to a *CanvasApiService*, and items added to the navigation bar for specific roles. The used methods are defined as a *ModuleMethod* containing the different development support steps with inputs and outputs. Moreover, with the *CanvasApiService*, which implements a standardized *ModuleApiService*, each of those support steps is connected to a dedicated component using the internal routing system. Moreover, the functionalities for creating, editing, and viewing meta artifacts are registered using the registry of the *CanvasMetaArtifactApiService*.

```

1 @Injectable({
2   providedIn: 'root',
3 })
4 export class CanvasService {
5   constructor(
6     private canvasApiService: CanvasApiService,
7     private canvasMetaArtifactApiService: CanvasMetaArtifactApiService
8     ,
9     // ...
10  ) {}
11
12  init(): void {
13    const module = new Module(
14      'Canvas Tools',
15      'Canvas Module',
16      this.getMethods(),
17      this.canvasApiService,
18      [
19        {
20          name: 'Model Composition',
21          route: ['companyModels'],
22          roles: [InternalRoles.EXPERT],
23        },
24        // ...
25      ]
26    );
27  }
28 }

```

```
25     );
26     this.moduleService.registerModule(module);
27     console.log('Registered Canvas Module');
28
29     // Similar registrations for edit and view the artifacts
30     this.canvasMetaArtifactService.registerCreateMethod(this.
        createModel);
31 }
32
33 // Similar functions for edit and view the models
34 async createModel(
35     router: Router,
36     reference: Reference,
37     artifactId: DbId
38 ): Promise<void> {
39     const queryParams = referenceToApiQueryParams(reference);
40     await router.navigate(['canvas', 'artifact', artifactId, 'create'
        ], {
41         queryParams,
42     });
43 }
44
45 private getMethods(): ModuleMethod[] {
46     const FormBuilder = this.fb;
47     return [
48         // ...
49         {
50             name: 'editCanvas',
51             description: 'Edit a canvas',
52             input: [{ name: 'Canvas', type: CompanyModel.typeName }],
53             output: [{ name: 'Canvas', type: CompanyModel.typeName }],
54             isMethodCorrectlyDefined: this.isMethodCorrectlyDefined.bind(
                    this),
55             getHelpTextForMethodCorrectlyDefined:
56                 this.isMethodCorrectlyDefinedHelpText.bind(this),
57         },
58         // ...
59     ];
60 }
61 //...
62 }
```

Listing 8.3 Service for the Canvas Tool

The *CanvasMetaArtifactApiService*, which is shown in Listing 8.4, implements the standardized *MetaArtifactApi* with functionalities for creating, editing, and viewing the artifacts using the registered canvas tool in the *CanvasService*. Moreover, additional functionalities for getting the name of an artifact, copying, removing, and comparing it need to be implemented.

```
1 @Injectable()
2 export class CanvasMetaArtifactApiService implements MetaArtifactApi
3 {
4   createMethod?: MetaArtifactApi['create'];
5   editMethod?: MetaArtifactApi['edit'];
6   viewMethod?: MetaArtifactApi['view'];
7
8   constructor(
9     // ...
10  ) {}
11
12  // Edit and view hooks are similar to create
13  create(router: Router, reference: Reference, artifactId: DbId):
14    void {
15    if (this.createMethod == null) {
16      console.warn('No module for the creation of canvas models added
17        ');
18    } else {
19      this.createMethod(router, reference, artifactId);
20    }
21  }
22
23  // Copy and remove artifacts are similar to create
24  async getName(model: ArtifactDataReference): Promise<string |
25    undefined> {
26    const companyModel = await this.companyModelService.get(model.id)
27      ;
28    if (companyModel.instances.length > 0) {
29      return companyModel.instances[0].name;
30    }
31    return undefined;
32  }
33  // ...
34 }
```

Listing 8.4 API Service for the Canvas Meta Artifact

For the **Database**, we use NoSQL as a storage and retrieval mechanism to allow flexible usage for the artifact management of the different modules. Here, PouchDB⁶ is a JavaScript database that directly runs within different web browsers by storing the data locally in the web storage. It is inspired by the functionalities of CouchDB. CouchDB⁷ is a document-oriented database where the storage mechanism is based on JSON and the query mechanism on JavaScript. Here, PouchDB is also able to synchronize its own data with a CouchDB and, therefore, allows their usage across different web browsers. In our situated implementation, we use PouchDB to allow the usage of our software tool directly in the web browser. To support the collaboration of different stakeholders and persistent storage, PouchDB can be connected to a CouchDB together with flexible user management.

For the **Additional Libraries**, we use different packages to reduce the development time and increase the quality of our software tool. First, the *BPMN.js*⁸ library is used to create models in BPMN 2.0, embed those diagrams in own applications, and support the extension by different customizations. In our situated implementation, we use BPMN.js in various parts. Those are the knowledge provision of method patterns, the pattern-based construction of methods, the execution of composed development methods, and the change of those methods. Second, *Quill*⁹ is a What-You-See-Is-What-You-Get (WYSIWYG) editor to create and update text documents, including media data, in the browser by a communication based on JSON. We use Quill to collaboratively create information artifacts during the execution of the development process and their discovery within the artifact manager. Third, *JSON Schema*¹⁰ is a vocabulary to annotate and validate JSON documents. For that, JSON Schema allows the definition of separate schemas for different JSON inputs, where the properties can be flexibly checked. Here, Listing 8.5 shows a part of the schema for the knowledge of the canvas knowledge. Here, general information about the location of the schema, the title, and the description is given. Moreover, for each JSON file, we define general knowledge about the model (e.g., *name*), the used canvas (e.g., *definition*), the items (e.g., *feature*) as objects, and instances (e.g., *instance*) as arrays. Here, *features* and *instances* relate to external definitions that are referenced through the *\$ref*. Moreover, the development support developers are able to design their own schemas for their artifacts.

```

1 {
2   "$id": "http://github.com/sebastiangtts/situational-business-
      model-developer/...",
3   "title": "Canvas Knowledge Schema Definition",

```

⁶Website of PouchDB: <https://pouchdb.com/>

⁷Website of CouchDB: <https://couchdb.apache.org/>

⁸Website of BPMN.js: <https://bpmn.io/>

⁹Website of Quill: <https://quilljs.com/>

¹⁰Website of JSON Schema: <https://json-schema.org/>

```

4      "description": "This schema defines the supported JSON files for
      the canvas knowledge",
5      "type": "object",
6      "properties": {
7          "$definition": {
8              "description": "The definition of the canvas structure.",
9              "$ref": "#/definitions/canvasDefinition"
10     },
11     "name": {
12         "description": "Name of the canvas knowledge.",
13         "type": "string"
14     },
15     "features": {
16         "description": "The list of features that is used in the
            canvas knowledge.",
17         "type": "object",
18         "additionalProperties": {
19             "$ref": "#/definitions/feature"
20         }
21     },
22     "instances": {
23         "description": "The list of patterns and examples that
            are used in the canvas knowledge.",
24         "type": "array",
25         "items": {
26             "$ref": "#/definitions/instance"
27         },
28         "uniqueItems": true
29     }
30     //...
31 }
32 }

```

Listing 8.5 JSON Schema for the Canvas Knowledge

We use JSON Schema for importing and exporting the knowledge of the method and the canvas model repositories and, therefore, support the transfer of that knowledge between different method engineers. Moreover, internal modules can also use those libraries to support the import and export of their created knowledge. Based on those technologies for the frontend, the database, and additional libraries, we implemented our software tool.

8.2 Software Tool

Our developed BMDT is called *Situational Business Model Developer (SBMD)*. The SBMD supports all three proposed stages of knowledge provision, composition and enactment of development methods, and support of development steps. Moreover, the source code of the tool is released as open-source¹¹, and the tools can be directly used in the web browser¹². To allow an easy extension and provision of modules, we based our tool on the highly used web framework Angular. The installation guidelines of the tool are presented in Appendix B.

In the following, we give an overview of the implemented stages within our software tool (8.2.1). Based on that, we present the developed versions of our modules (8.2.2).

8.2.1 Implemented Tool

In this thesis, we provide the SBMD as the situated implementation of our solution concept and the modularized architecture. In the following, we present parts of our implementation according to the stages of knowledge provision of methods and models, composition and enactment of development methods, and support of development steps.

Knowledge Provision of Methods and Models In the first stage, we provide CRUD operations for the knowledge inside the method repository and the canvas model repository. Inside the method repository, as shown in Figure 8.4, method elements are combined into method building blocks and, optionally, structured according to method patterns. Here, as shown in Figure 8.4 (a), we have the modeling of a method building block (see Section 5.2.2). For that, we define the name of the building blocks (*A*) together with a description (*D*). Moreover, we are able to add different types or type lists (*B*) together with nominal or ordinal situational factors (*C*). As shown in Figure 8.4 (b), we have the modeling of the method patterns (see Section 5.2.3). For that, we define the name of the pattern (*E*) and the corresponding situational factors (*H*). Next to that, we construct a BPMN pattern (*F*), where each activity is linked to the supported types (*G*).

Inside the canvas model repository, as shown in Figure 8.5, canvas elements are structured through canvas building blocks and visualized through canvas models. Here, as shown in Figure 8.5 (a), we have the modeling of the canvas building blocks (see Section 5.3.2). For that, each building block is directly mapped to a canvas model (*A*) where different items can be placed (*B*). Moreover, the application domains for the building block are defined (*C*).

¹¹Source Code of the Tool: <https://github.com/sebastiangtts/situational-business-model-developer>

¹²Online Version of the Tool: <http://sebastiangtts.github.io/situational-business-model-developer/>

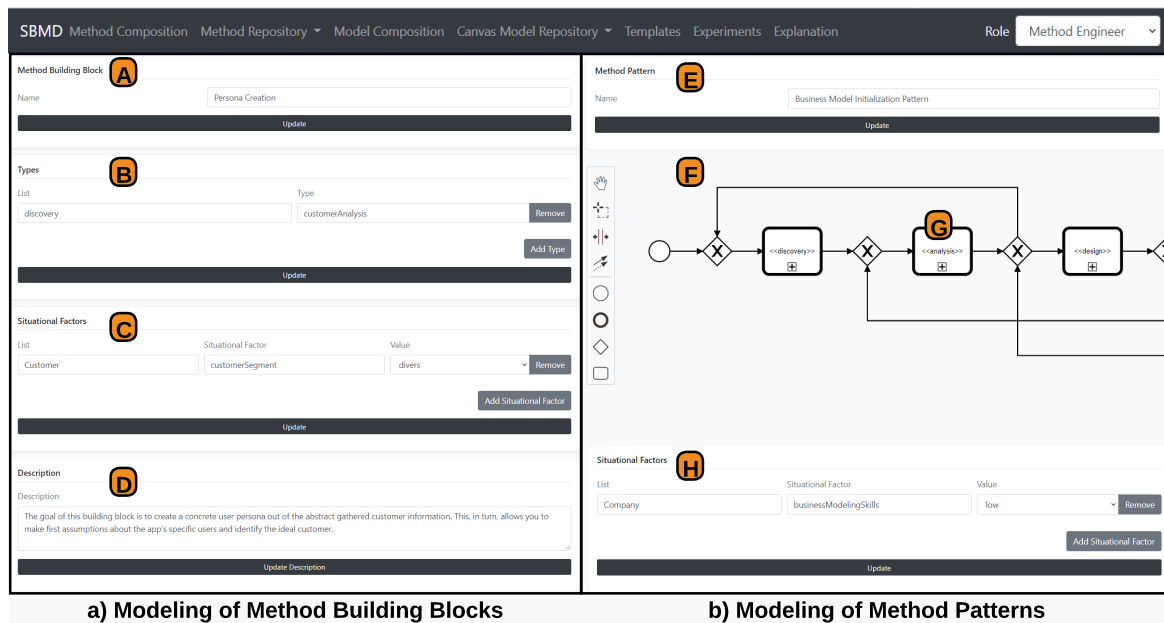


Fig. 8.4 Screenshots for Provision of Method Repository

As shown in Figure 8.5 (b), we have the modeling of the canvas models (see Section 5.3.3). Here, we define the visualization of the model based on the definition of rows and columns (D). Moreover, we provide a preview of the visualization (E) together with definitions of different supported relationships (F).

Composition and Enactment of Development Methods In the second stage, we construct and execute the development method out of the method repository and the canvas model repository. For the development method composition, as shown in Figure 8.6, we compose the methods and models based on the defined context. Here, as shown in Figure 8.6 (a), we define the context (see Section 6.2.1) by selecting application domains (A) and situational factors (B) from the repositories. Moreover, we show the pattern-based composition of methods in Figure 8.6 (b) (see Section 6.2.2). Here, the method is constructed from different nested patterns (C). Moreover, the activities are filled with method building blocks that are complete (E) or incomplete (D).

For the development method enactment, as shown in Figure 8.7, we enact the composed development method. Here, as shown in Figure 8.7 (a), we provide an execution of the development process (see Section 6.3.1). For that, we provide a Kanban board (A), where the different development steps can be placed on (B). Moreover, in Figure 8.7 (b), we have the collaboration of the stakeholders in the artifact development. Here, we provide a canvas board (C), where the different items can be placed on (D).

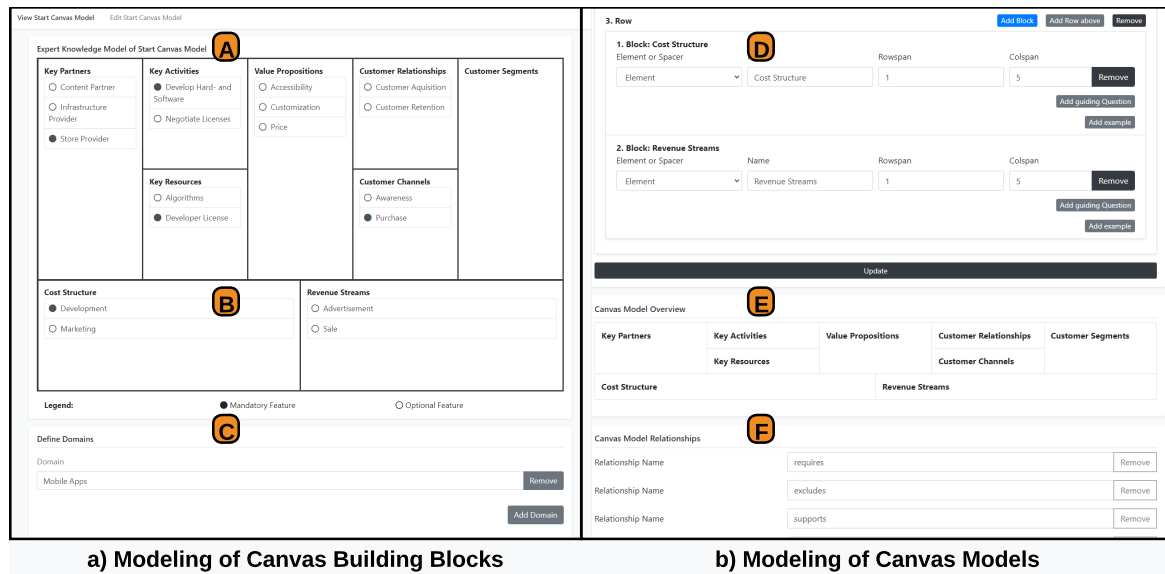


Fig. 8.5 Screenshots for Provision of Canvas Model Repository

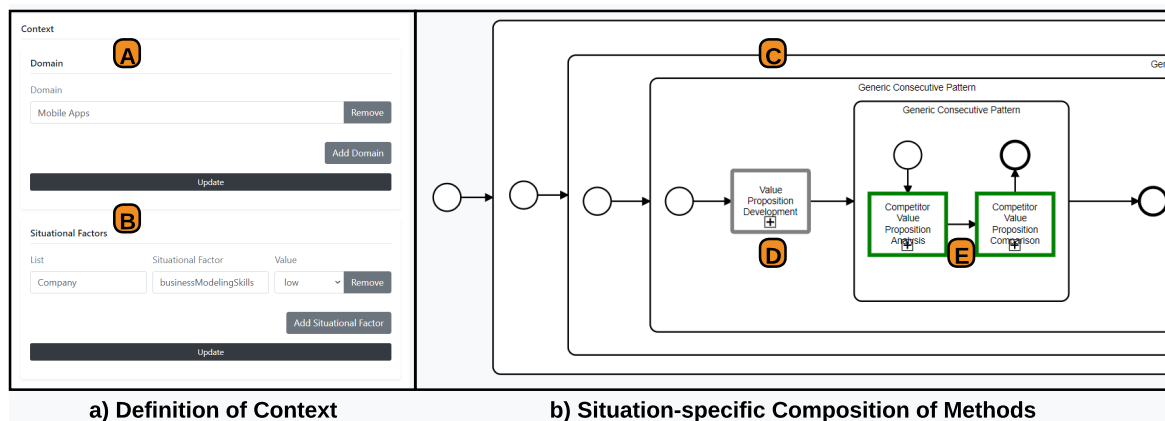


Fig. 8.6 Screenshots for Composition of Development Methods

Support of Development Steps In the third stage, we add flexible IT support to certain development steps. Here, as shown in Figure 8.8, we have the modularization of development support (see Section 7.2). In Figure 8.8 (a), we have the composition of modules, where we add different execution steps to a method building block (A). Those steps are combined so that they are executed in a row (B) and the outputs of a single step can be used as input in another one (C). In Figure 8.8 (b), the enactment of the modules is shown, where the steps are executed after each other (D). Moreover, the output of the execution steps can be stored in a new artifact or merged with an existing one (E).

Based on that modularization concept for supporting the development steps, we have implemented different modules to support the different phases of BMD.

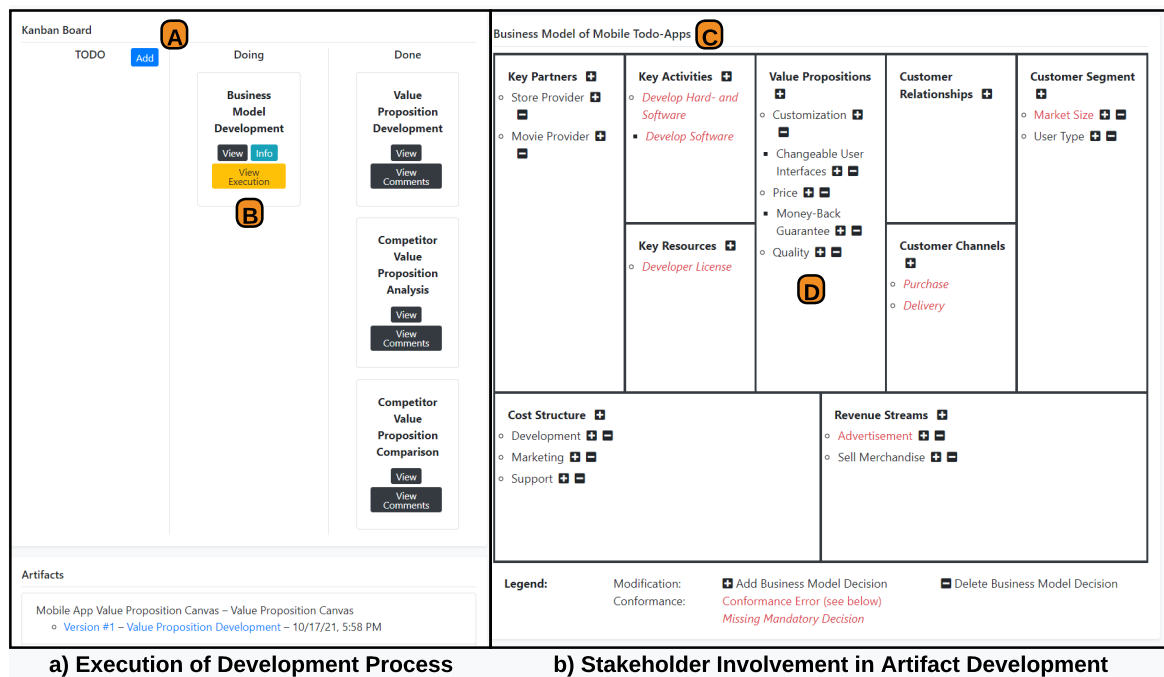


Fig. 8.7 Screenshots for Enactment of Development Methods

8.2.2 Developed Modules

Based on the modularization within the third stage of development support, we have implemented three exemplary modules to support the BMD. Those were the canvas module, the hypothesis modeling and mapping module, and the crowd-based prototype validation platform.

Canvas Module The integrated module was developed over the three design cycles and conceptualized in Section 7.3.1. For that, we use table structures to align the building blocks of the canvas models. Moreover, we have added CRUD operations for the elements, building blocks, and models together with using a mix of existing and modified algorithms for knowledge consolidation and conflict detection.

Parts of the implementation can be seen in Figure 8.9, where we provide the design support for such canvas models. Here, Figure 8.9 (a) shows the refinement step of the canvas models. For that, the *Business Developer* can activate different types of design support like strengths/weaknesses, hints, or patterns (A). Based on that, he receives information to support the design, like strengths (B), or selects a specific design support item to visualize (C). Moreover, Figure 8.9 (b) shows the competitor comparison step of different canvas models. Here, the differences between the canvas models are shown visually together with a

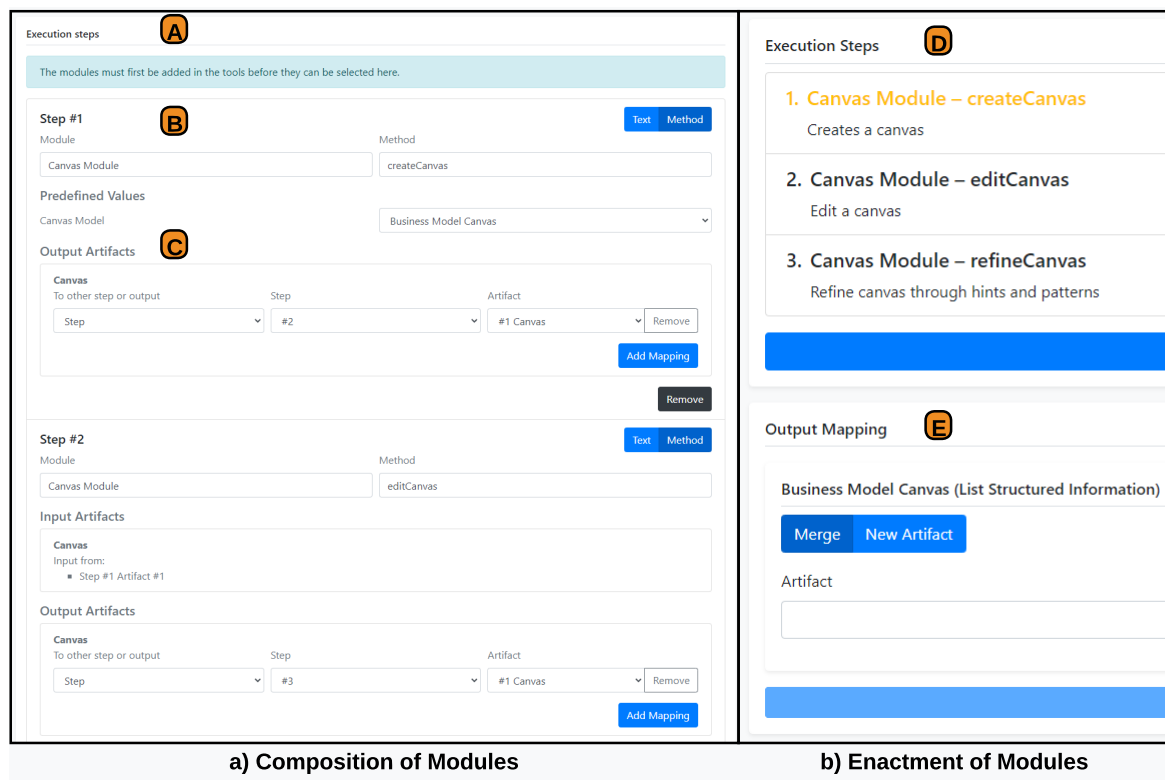


Fig. 8.8 Screenshots for Modularization of Development Support

comparison score (*D*). Here, the module supports the *Business Developer* in the design of new business models.

Hypothesis Modeling and Mapping (HypoMoMap) Module The internal module was developed in the first design cycle to support the validation phase. Similar to the canvas module, we use the underlying structure of a hierarchical tree together with CRUD operations for the hypotheses, experiments, and their mappings. The selection of the corresponding experiments is made through small designed algorithms.

Parts of the implementation can be seen in Figure 8.10, where we support the modification of the HypoMoMap artifacts. Here, Figure 8.10 (a) shows the hypotheses' modeling for validation. For that, the *Business Developer* structures those hypotheses in a tree (*A*), where each hypothesis has a starting priority and can be modified (*B*). Moreover, Figure 8.10 (b) shows the execution of experiments for validation. Here, based on the algorithm choice of the *Business Developer*, corresponding experiments are provided where one needs to be selected (*C*), and connected hypotheses need to be proofed/disapproved. Here, the module supports the *Business Developer* in the structured validation of the business model and the product features.

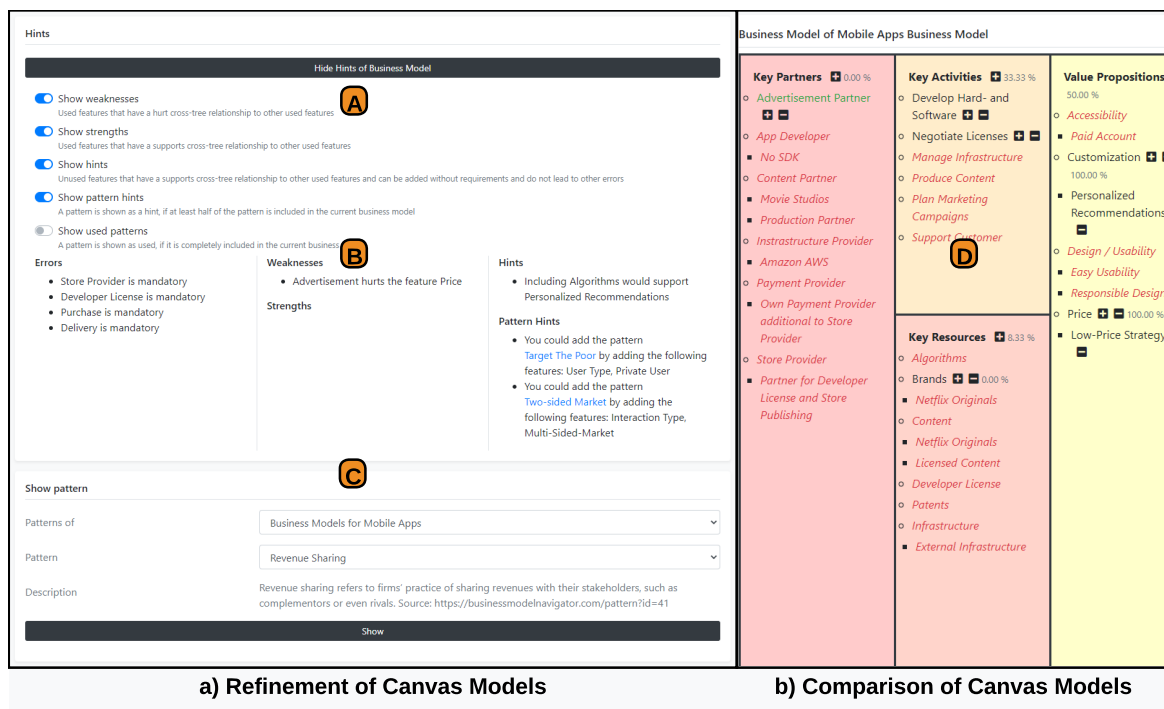


Fig. 8.9 Screenshots of the Canvas Module

Crowd-based Prototype Validation (CPBV) Platform The platform is the external support tool that we have conceptualized in Section 7.3.3 in the form of design principles (DPs). We represent the functionalities of our platform using design features (DFs) as explained in our research approach in Section 7.3.3. Here each DP is translated to a set of DFs that can be directly implemented in the platform.

For **DP 1: User Variety**, we allow the registration of developers and users both by a registration form (*DF 1*) and a single sign-on service (*DF 2*). Moreover, we provide user profiles with specific information like skills (*DF 3*) and a messaging system between different users (*DF 4*). As **DP 2: Task Iteration**, we provide the creation of tasks with essential information (*DF 5*), the provision of feedback with a questionnaire (*DF 6*), and the representation of feedback (*DF 7*). For **DP 3: Prototype Diversity**, we allow the provision of prototypes as textual descriptions (*DF 8*), uploaded images (*DF 9*), and integration of external prototyping tools (*DF 10*). In addition to that, **DP 4: Test Diversity** contains the test of single prototypes (*DF 11*), the comparison of multiple prototypes (*DF 12*), and the usage of split tests (*DF 13*). Here, the questionnaire also allows multiple questions like stars rating, thumbs-rating, radio buttons, and free text fields (*DF 14*).

For **DP 5: Filter Mechanisms**, we support adding required user profile criteria to the tasks (*DF 15*) and the shortlisting of tasks by the preferences of the users. As **DP**

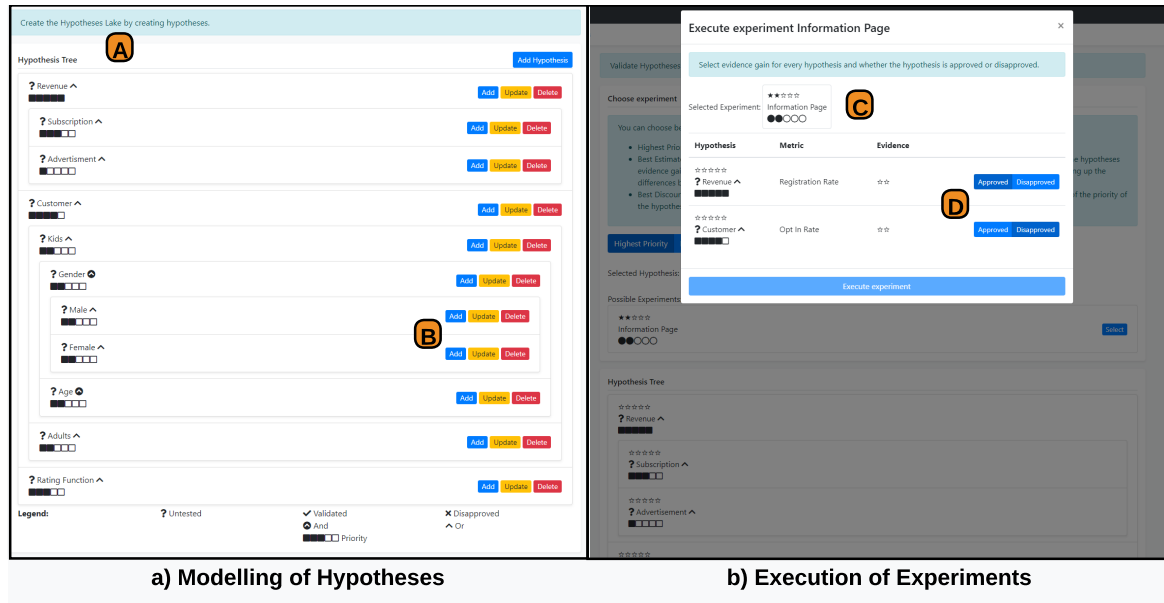


Fig. 8.10 Screenshots of the Hypotheses Modelling and Mapping Module

6: Aggregation Mechanism, we provide visualization charts of aggregatable answers to questions (*DF 16*), investigation of individual feedback of each user (*DF 17*), additional feedback for revealing unconsidered questions (*DF 18*), and the comparison of split-test results (*DF 19*). Based on that, **DP 7: Incentive Mechanism** allows the extrinsic motivation of sending virtual money (*DF 20*) and the intrinsic motivation of publicly displaying the user's trustworthiness (*DF 21*). As **DP 8: Non-Disclosure Mechanisms**, we support the direct approval of tasks to all users (*DF 22*), the deposit of a non-disclosure agreement that the users have to accept (*DF 23*), and the manual approval of users (*DF 24*). Finally, for **DP 9: Governance Mechanism**, we provide an admin control panel (*DF 25*) together with the reporting of tasks and users (*DF 26*).

The situated implementation of the CPBV is shown in Figure 8.11. As a technique, we have also used Angular to develop the frontend. Nevertheless, we used NestJS¹³ for the backend to provide an API for managing the platform and PostgreSQL¹⁴ for the database to provide structure for the received data in contrast to the initial software tool. Here, Figure 8.11 (a) shows the creation of a new task by the developers. For that, the developer selects a basic test selection as one possible test type (i.e., *A / DF 11*), uploads images of prototypes (i.e., *B / DF 9*), and provides questions to answer (i.e., *C / DF 14*). Figure 8.11 (b) shows the conduction of a task by the users. Here, the user executes the prototype by viewing the image (i.e., *D / DF 9*) and answering the questions of the questionnaire (i.e., *E / DF 14*).

¹³Website of NestJS: <https://nestjs.com/>

¹⁴Website of PostgreSQL: <https://www.postgresql.org/>

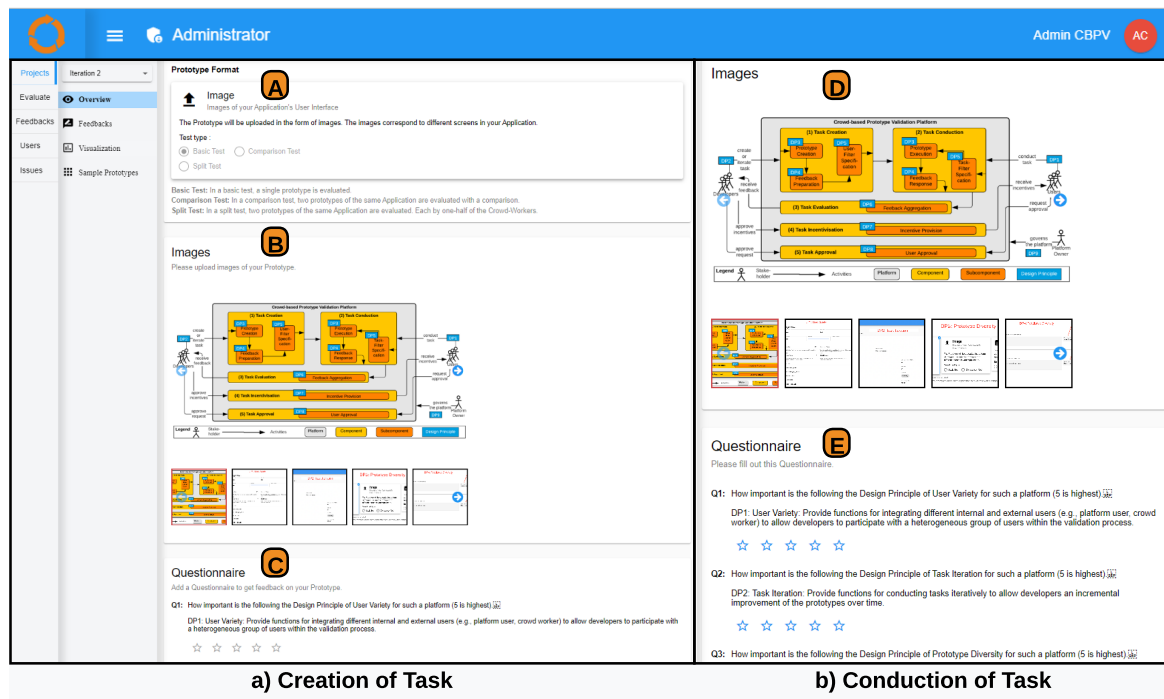


Fig. 8.11 Screenshots of the Crowd-based Prototype Validation Platform

8.3 Summary

In this chapter, we have provided the implementation for all three stages of our approach. For that, we have shown the modularized architecture on which the solution is based. Moreover, we have provided a technical implementation based on web technologies that can be used within the web browser.

For the **Modularized Architecture**, we have shown a component diagram of the most important aspects that the solution needs to fulfill. Based on that, we presented our used technologies around the Angular framework for the web-based implementation of the software tool out of the architecture.

For the **Software Tools**, we have shown different user interface aspects of the three implemented stages within the tool. Based on that, we presented interface parts of the three exemplary modules that were developed within this thesis.

Based on the implemented solution, we will show its evaluation and the associated improvements for the second and third design cycles in the next chapter. First, we will show a case study on developing business models for a local event platform. Second, we will present a user study on developing business models for mobile prototypes within student courses.

Chapter 9

Evaluation

In the previous chapter, we showed the modularized architecture together with the software tool of our solution. Based on that, this chapter evaluates the applicability of our situation-specific BMD approach. For that, we show the conduction of a case study on the OWL Live event platform and a user study in students' courses at Paderborn University.

9.1 Case Study on OWL Live

To evaluate our approach, we conducted a case study on developing possible business models for a local event platform called OWL Live. The case study is based on the evaluation of the second DSR cycle as proposed in Section 1.2, and, therefore, includes just the first two stages of our solution. The OWL Live platform is developed as part of a research project at Paderborn University and aims to provide an aggregated view of events in the local area of East Westphalia-Lippe. We base our case study on the well-accepted guidelines of Runeson and Höst [RH09] to increase the quality of the study results. The study aims to uncover possible difficulties in the composition and enactment of situation-specific business model development methods as proposed in RQ 2. Based on the guidelines, we divide the evaluation into the steps of forming the *Experimental Design* of the evaluation, followed by its *Execution*, the *Analysis* of the results, and their *Interpretation* concerning our approach.

9.1.1 Experimental Design

In the beginning, we need to define the experimental design in which the case study takes place. By referring to the guidelines of Runeson and Höst [RH09], the design can be divided into the definition of a case, the initiation of a case study protocol, and the clarification of ethical considerations.

Definition of a Case Due to the flexibility of the case study design, preliminary and adjustable planning is an essential factor before the actual conduction of the study. For our study, we justify our plan, as explained in the next paragraphs, based on the elementary parts of the objectives to achieve, the research question to answer, the theories to concern, the case to study, the methods to rely on, and the selection strategy to use as discovered by Robson and McCartan [RM16].

The **Objective**, **Research Questions** and **Theories** of the study directly relate to the superior design science study in the thesis, which aims to develop a situation-specific BMD approach. The refined *Objective* of this study is based on the integrated concept of situation-specific development of business models. Here, our case study aims to develop potential business models for the local event platform and identify issues with our concept and the software tool that should be improved within the next design cycle. For that, we follow an exploratory purpose for seeking new insights and generating new ideas for possible improvements [RM16]. Based on this refined objective, we also refine our overall *Research Question* of how to enable the situation-specific development of business models for service providers within software ecosystems (see RQ in Section 1.2). This refinement leads to our new second question of how to handle the situation-specific composition and enactment of BMDMs by using utilized knowledge for the development methods and within the canvas modeling artifacts (see RQ 2). Moreover, we want to improve our approach to the existing first research question on how to utilize the knowledge about development methods and canvas modeling artifacts to support the BMD (see RQ 1). As *Theory*, we make use of the theories of boundary objects [SG89] and opportunity creation [ABA13] that are also used for our DSR study. Here, the boundary objects theory is applied to provide a common understanding among all stakeholders for providing the knowledge of methods and models, while the opportunity creation theory is applied to provide continuity in the composition and enactment of the development methods.

As **Case**, we provide a holistic case study, analysis of a unit in one case [Yin09], on developing possible business models for a local event platform called OWL Live. The platform is created as part of an ongoing research project and acts as a two-sided platform between event providers and event visitors. Here, the owner wants to use new data mining techniques to aggregate events from different event providers together with natural language processing to provide an enhanced recommendation system to the event visitors. By developing the business models in an ongoing research project, we also have two specialties that we need to consider during the development. First, the project is backed up by a consortium of public institutions (i.e., university, culture office) and a private company (i.e., development agency) that are involved in the platform. Second, the project partners have already discovered parts

of the business models as parts of ongoing discussions, a design thinking workshop, and a preliminary feasibility study on which the business model should base.

As **Methods** to collect data and corresponding **Selection Strategy**, we use a mix of direct and independent methods as characterized by Lethbridge et al. [LSS05]. As a direct method, we conduct virtual and physical interviews to gather information about the knowledge that should be used in methods and models, the project context for the composition of the development method, and the stakeholder information during the enactment of the development method. We use those interviews to gather the unfiltered information of the participants. As an independent method, we use Grey Literature Reviews (GLRs) [GFM19] to identify the knowledge that is used for the repositories of the methods and the models. GLRs, in turn, aim to review resources outside academic research, so-called grey literature, to gather insights on a particular domain. We use a GLR to identify knowledge that practitioners outside academia created. Moreover, we analyze the existing information generated through discussions, the design thinking workshop, and the feasibility study in the past. While some of the information is publicly accessible (e.g., the feasibility study), other information is confidential (e.g., current monetization calculation).

Initiation of a Case Study Protocol Before starting the conduction of our case study, we need to initiate our case study protocol. Here, the protocol is a continuously changed document that keeps track of the study's execution, analyzes the results, and draws conclusions regarding the interpretation. According to Maimbo and Pervan [MP05], we structure our protocol the following way: First, we justify the purpose of the protocol and information about the data storage. Second, a brief overview of the case and its research method are given. Third, the procedure of our case and its execution steps are explained in detail. Fourth, the research instruments are explained. Fifth, the data analysis is clarified. For our case study, a text document captures all information outside the stages of our approach. For the stages themselves, we are using the traceability feature of our software tool.

Clarification of Ethical Considerations Last, before conducting the case study, we need to consider ethical considerations. This, in turn, increases the trust between the participants and us. For this, we consider the following things: First, we explain the case study to every directly involved participant and inform them about the whole procedure. Second, we state our data collection and storage strategy. Third, we clarify the publication of parts of the study and omit confidential information.

9.1.2 Execution

During the conduction, we structured our procedure according to the two applied stages of knowledge provision of methods and models and the composition and enactment of development methods.

Knowledge Provision of Methods and Models For the first stage, we interviewed the project manager to get a scope in which the possible business models should be developed. Out of that context, we conduct a GLR to gather knowledge about business models for mobile apps in general, digital platforms, digital marketplaces, and content aggregators. Here, we followed the guidelines according to Garousi et al. [GFM19], who structured the GLR in the three phases of planning the review, conducting the review, and reporting the review.

In **(1) Planning the Reviews**, the need for a GLR needs to be motivated together with the explicit formulation of the RQ the study aims to answer. The search string and related inclusion and exclusion criteria are determined from the RQ. To motivate the need for the GLR, we have used the checklist of Garousi et al. [GFM19]. Here, we concluded the need for a GLR based on the subject's complexity and the lack of practical experience in the formal literature. Here, we used GLRs to utilize knowledge for the method repository and the canvas model repository.

For providing the *Method Repository*, we have defined the following RQ: *What are the main business model development steps that need to be done by a mobile app developer?* To answer this question, we have defined the following search string: *app AND (business model OR idea) AND (test OR validate OR develop)*. We include articles in English where the URL is accessible and directly connected to the RQ. We exclude articles that provide no BMDM or process, do not relate to mobile apps, are posted in forums, or are presented in non-textual form. With this exclusion, we aim to increase the quality of the knowledge in the method repository.

Similarly, for providing the *Canvas Model Repository*, we have defined the following RQ: *What are the main decisions for the business model that need to be made by a mobile app developer?* To answer this question, we have defined the following three search strings: *business model AND digital platform, business model AND digital market place, business model AND content aggregator*. We include articles in English where the URL is accessible and directly connected to the RQ. We exclude articles that provide no knowledge of the business models, are just landing pages for marketing of companies, are posted in forums, or are presented in non-textual form. With this exclusion, we aim to increase the quality of the knowledge in the canvas model repository.

In **(2) Conducting the Review**, the review needs to be conducted by considering the search process, the source selection, the quality assessment, the data extraction, and the data synthesis. Here, we conducted the GLRs for the knowledge of the method repository and the canvas model repository.

In the search process for the *Method Repository*, we applied the search string to the Google search engine on January 19th, 2021, and anonymized our browser data for maximum objective results. We exported the first 50 search results and manually scanned the inclusion and exclusion criteria, resulting in 38 articles. For the quality assessment, the essential criteria were the understandability of the presented business model development method or processes. Moreover, the *Method Repository* contains links to the articles to convince mobile app developers of the quality. In the data extraction, we divided the gained information into the different *Method Elements* (e.g., *Tasks*, *Stakeholders*). Out of that, in the data synthesis, we combined them into *Method Building Blocks* and structured them into *Method Patterns*.

Similarly, for the *Canvas Model Repository*, we applied our search string on June 18th, 2021, exported the first 25 results for each search, and, after applying the criteria, obtained eight results for the digital platform, 20 for the digital marketplace, and 15 for the content aggregator. In the data extraction, we divided the gained information into the different *Canvas Elements* (e.g., *Items*, *Patterns*). In the data synthesis, we arranged them in *Canvas Building Blocks* and mapped them to different *Canvas Models*. Those models were the existing Value Proposition Canvas and Business Model Canvas, together with a newly defined Feature Set Canvas for the product features.

In **(3) Reporting the Review**, we documented the review results. Here, a part of the *Method Repository*, which consists of examples for the *Method Elements*, *Method Building Blocks*, and *Method Patterns*, can be seen in Figure 9.1. For the *Method Elements*, we found *Situational Factors* with ordered (e.g., *businessModelingSkills* can be *low*, *medium*, *high*) and unordered (e.g., *marketSize* can be *niche* or *mass*) factors, *Types* for structuring the building blocks (e.g., *customerIdentification*) as well as nesting the patterns (e.g., *customerDeepDiscovery*), and *Tasks* (e.g., *Conduct Interview*). Moreover, we found different internal (e.g., *Business Developer*) and external (e.g., *Customer*) *Stakeholders*, textual (e.g., *Customer Information*) as well as canvas-based (e.g., *Value Proposition Canvas*) *Artifacts*, and *Tools* (e.g., *AppAnnie*). Out of those elements, we combine different *Method Building Blocks* like *Interview Customer* or *Create Landing Page*. For the *Interview Customer*, we have the output artifact of *Customer Information* by integrating the stakeholders of the *Business Developer* and the *Customer* together with *low* value for the factor of *customerValidity*. For the *Create Landing Page*, we have the output artifact of the *Landing Page* by integrating the *Marketing Manager* and *Designer* with *medium* value for the factor of *designSkills* and *medium* value

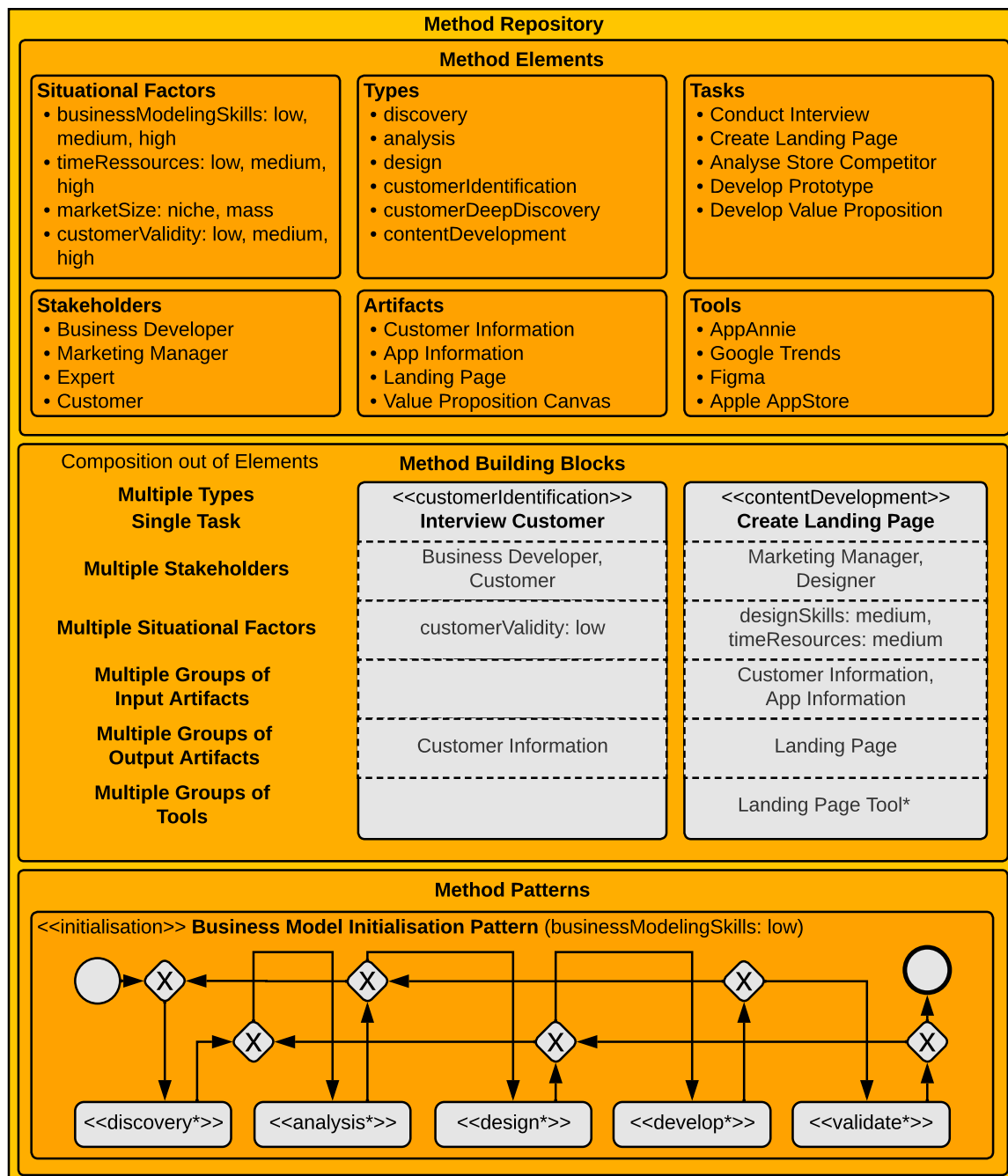


Fig. 9.1 Knowledge Provision: Examples of the Method Repository based on the GLR

for the factor of *timeResources* based on the input artifacts of *Customer Information* and *App Information* together with a flexible *Landing Page Tool*. Those building blocks are structured through *Method Patterns* like the *Business Model Initialisation Pattern*. Here, this pattern can then be used as a starting point of the development by a *low* value for the factor of

businessModelingSkills and provides a structure for all phases (*discovery, analysis, design, develop, validate*) of the BMD, which activities can be filled building blocks and patterns.

Moreover, a preliminary version of the whole method repository is provided inside the corresponding technical report [GYNE21c]. Furthermore, the results for both repositories can be accessed in our software tool. The installation of our tool is explained in Appendix B.

Composition and Enactment of Development Methods For the second stage, we needed to compose and enact a development method to develop potential business models.

For the **Composition of the Development Method**, we interviewed the project manager to gather information about the current state of the platform (e.g., target customer), the situation of the project (e.g., marketSize:mass), and the application domain of the app (e.g., content aggregation). Moreover, we gathered additional construction constraints that need to be included in the development method (e.g., closed beta tests, different target groups). Out of that context and the additional constraints, we composed the development method as shown in Figure 9.2. We structure those methods through our extracted *Business Model Initialisation Pattern*, see Figure 9.1, according to the five phases of discovery, analysis, design, development, and validation.

In the **(1) Discovery**, we suggested providing a *Target Audience Identification* to derive the different target groups that might use the platform. Based on that, we suggested a *Market Problem Observation* to analyze the pain points of the identified target groups together with a *Store Trend Analysis* to broaden the own ideation scope of features that the user might use. In the end, and with respect to the two-sided market, *Customer Interviews* support validation of the smaller target group of event providers, while *Social Media Survey Conduction* supports validation of the larger target group of event visitors.

In the **(2) Analysis**, we suggested running a *Market Potential Analysis* to estimate the size of the market that the platform could achieve. This should also support convincing external stakeholders that the platform has the potential to run successfully. Moreover, a *Store Competitor Analysis* inside the stores but also a *Competitor Analysis* outside the stores support the identification of requested features by the users or specific aspects of the business models. Those competitor analyses, in turn, support a focus on the most important features for distinguishment.

In the **(3) Design**, we gathered information on the last two steps for designing the platform. For that, a *Feature Set Creation* supports to structure of potential features of the platform. Moreover, the *Value Proposition Development* supports the identification of the target group and fosters their pains and gains. Those pains and gains are solved by specific value propositions inside the *Business Model Development*. Here, all *Canvas Artifacts* of

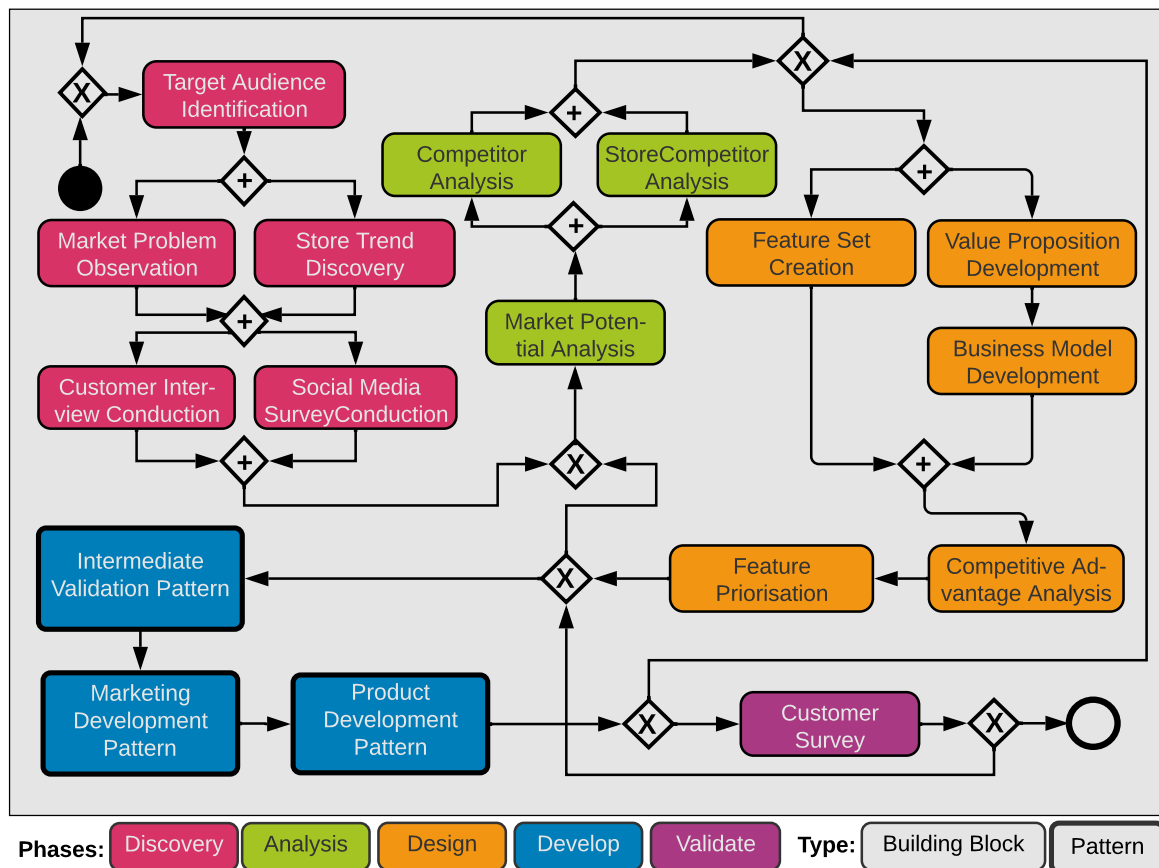


Fig. 9.2 Method Composition: BMDM for OWL Live based on the identified Context

the steps are linked to the *Canvas Models* (i.e., Value Proposition Canvas, Business Model Canvas, Feature Set Canvas) that are developed in the first stage, including the consolidation of the specific knowledge. The phase is completed by providing a *Competitive Advantage Analysis* of the own design against the competitors and using a *Feature Prioritisation* in order to derive those competitive advantages from the beginning.

In the **(4) Development** and with the additional information of the project manager, we suggested the development of a beta-version in front of the product development, which should be validated within the *Intermediate Validation Pattern*. Inside that pattern, the beta version will be tested with potential early adopters using user tests. Moreover, with the *Marketing Development Pattern*, we suggested conducting inbound marketing in parallel to the actual development of the MVP of the platform. Last, the *Product Development Pattern* consists of a selection of the used platforms, cost estimation of internal development, and the platform's first release.

Last, in the **(5) Validation**, we suggested the ongoing validation of both customer groups by using the scaleable approach of *Customer Surveys*. This can be done using external tools or directly integrating feedback possibilities within the platform.

For the **Enactment of the Development Method**, we used the ongoing interviewing of the project manager together with the already existing information (e.g., feasibility study, monetization calculation) of the project to develop the potential business models. Here, by referring to our constructed development method in Figure 9.2, we structure the enactment according to the discovery, analysis, design, and development phases.

During the **(1) Discovery**, the *Target Audience Identification* was used to identify the first groups of event providers and event visitors. Moreover, both groups were refined into different subgroups (e.g., culture actors for event providers and early adopters for event visitors). Moreover, a distinguishment between digitalized and non-digitalized providers as well as residents and tourists for visitors was made. For that, the results of a previous design thinking workshop and the information from an interview with the project manager were used. During the *Store Trend Analysis*, different trending apps (e.g., social networks like NextDoor, news aggregations like NewsBreak) were analyzed regarding possible features (e.g., invitation mechanism, social media connection) and business models (e.g., paid recommendations, affiliate marketing) of the platform. During the *Market Problem Analysis*, the main sources of event findings (e.g., word of mouth, posters) were identified, and occurring problems were analyzed. Those were divided into problems of the event provider (e.g., the manual effort for event provision, preselling of tickets) and the event visitor (e.g., the manual effort to find events, the missing of special events). While the *Customer Interview Conduction* for the event provider was already done in the design thinking workshop before, the *Social Media Survey Conduction* was done with an analysis of results from an existing social media website (i.e., Reddit) and the conduction of new polls on a local social media app (i.e., Jodel). Nevertheless, both development steps were instantiated again to gather more information over time.

During the **(2) Analysis**, the *Market Potential Analysis* was used to strengthen the results of the existing feasibility study. Here, key factors about the market (e.g., estimation of the number of events or locations) and the potential market groups (e.g., total numbers of groups or visited events) were identified. Moreover, the existing competitors in the market were identified and analyzed. The *StoreCompetitor Analysis* focused on general apps (e.g., ticket sellers, social networks with event features) in Apple's AppStore and Google's PlayStore. For example, Eventim already provides selling tickets for special events, Facebook Events can be used to discover various manual-created events, and Jodel can be used for special timely events. Moreover, the *Competitor Analysis* focused on the local providers (e.g., city

marketing, newspapers). Here, ErwinEvent provides information about local events and locations like the Paderborn theater, publish events on their website, and print posters.

During the **(3) Design** and based on the knowledge of the *Canvas Model Repository*, the Value Proposition Canvas, the Business Model Canvas, and the Feature Set Canvas were developed. The *Value Proposition Development* was done for the chosen target groups of event providers (e.g., already digitized providers) and event visitors (e.g., early adopters in the form of permanent residents). The *Business Model Development* was done for the three different ideas of a content aggregator (e.g., personalized advertisements, affiliate links to existing ticket sellers), an own ticket seller (e.g., personal arranged relationships, commission fee), and a sponsored platform (e.g., privacy-friendly usage, independent prioritization). Here, a simplified version of the business model of the content aggregator is shown in Figure 9.3. Within this business model, we have our two *Customer Segments* of *Permanent Residents* and *Digitalized Providers*. Moreover, we have defined different elements for the *Value Propositions* (e.g., *High Amount of Events*), the *Customer Relationships* (e.g., *Self Service*), the *Channels* (e.g., *Placements on existing Events*), the *Revenue Streams* (e.g., *Personalized Advertisements*), the *Key Partners* (e.g., *Event Platforms*), the *Key Activities* (e.g., *Develop WebApp*), the *Key Resources* (e.g., *API Gateways / WebCrawler*), and the *Cost Structures* (e.g., *Marketing / Branding*). Additionally, a new development step of SWOT analysis with its own SWOT Canvas was created to provide such an analysis for the different ideas of business models. For example, for the content aggregator, we have different strengths (e.g., good scalability), weaknesses (e.g., less personal contacts), opportunities (e.g., expansion in other regions), and threats (e.g., easy to copy) based on a canvas grid. Moreover, a corresponding *Feature Set Creation* for the platform was done. Here, we divided between basic features that are needed (e.g., search, filter, login) and advanced features that provide a competitive advantage (e.g., web crawler, recommendation algorithm). Out of those *Canvas Models*, a *Competitive Advantage Analysis* was done with the tool. Here, we added the competitors' information also in the canvas models and created instances of them. Last, we used the results to create a *Feature Prioritisation*, which divides between features that should be developed within the project and which not.

During the end of the case study, in the **(4) Development**, the app's beta version is developed in the *Intermediate Validation Pattern*, which should be in the future evaluated with test users in a user study. Moreover, during the development, the possible business models are presented to external stakeholders and, based on their feedback, could be further validated using ongoing *Customer Interview Conductions* and *Social Media Survey Conductions*. Here, we conducted an additional step by interviewing the project manager and the culture office employee on the relevance of the three business models. Last, inbound marketing

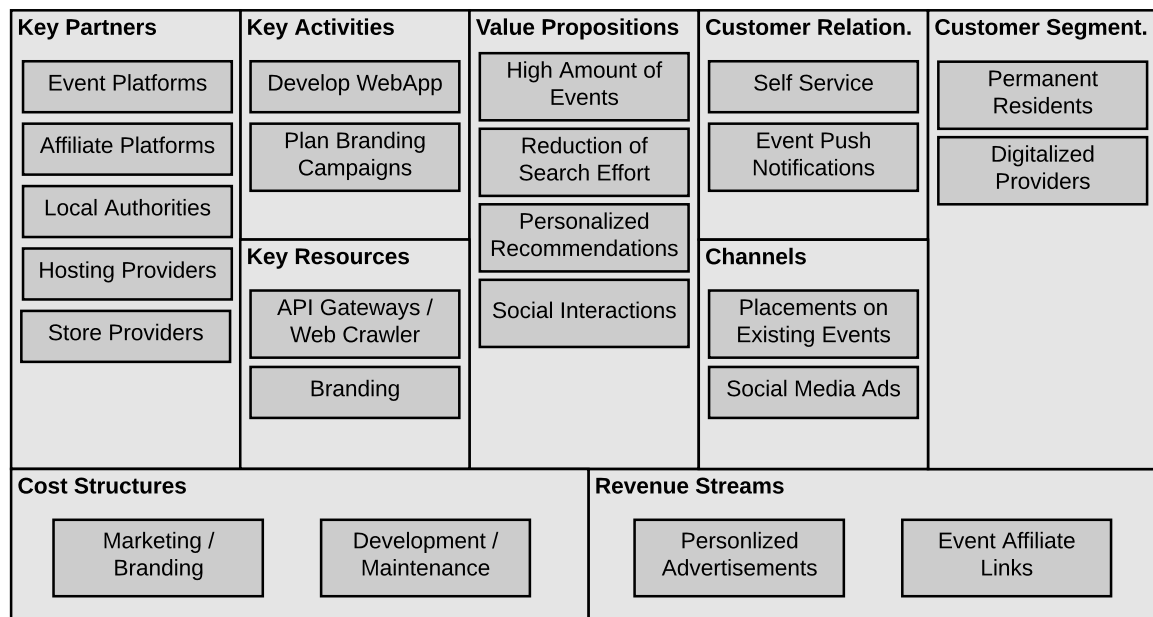


Fig. 9.3 Method Enactment: Example of the BMC for OWL Live

(e.g., landing page, social media posts) could be used in the future within the *Marketing Development Pattern* to ensure high traffic during the upcoming beta. Based on that execution, we conduct an analysis of our approach against the provided RQs.

9.1.3 Analysis

Our solution aims to develop a situation-specific BMD approach. By conducting a design science study, the purpose was to answer the overall RQ of how to enable the situation-specific development of business models. For the second cycle of DSR, we refined our RQ into the first question of how to utilize and store the knowledge of methods and models (RQ 1) and the second question of how to compose and enact development methods (RQ 2).

To evaluate the **Utilization and Storage of Knowledge** (RQ 1), we have conducted a GLR as an indirect method to create a *Method Repository* and a *Canvas Model Repository*. For the *Method Repository*, the identification of *Method Elements* and their combination into *Method Building Blocks* was straightforward. Here, arranging the *Method Patterns* was challenging due to the nested usage of patterns to allow specific orchestration of the development method during composition. For the *Canvas Model Repository*, the identification of single *Canvas Elements* was straightforward. Here, problematic was just the dimension of finding related elements for all building blocks of the canvas models and prioritizing which elements should be taken and which not. Moreover, the majority of digital platform results stayed on a high level without providing any insights. More challenging was the development of *Canvas*

Building Blocks. Here, it was hard to find the right granularity of the decomposition of the elements together with the mapping of item sets to instances and connecting items through relationships. Again, the presentation of building blocks to *Canvas Models* is simple due to a fixed matching of the building blocks. However, to allow the usage of the knowledge without external guidance, the textual descriptions of the *Method Elements* and the *Canvas Elements* need to be improved. Overall, the utilization and storage of knowledge were applicable concerning RQ 1. Nevertheless, it is advisable that a skilled *Method Engineer* does the steps of formalization of the knowledge instead of the *Domain Expert*.

To evaluate the **Composition and Enactment of Development Methods** (RQ 2), we developed the business model for a local event platform. Here, we have the *Composition of the Development Method* and the *Enactment of the Development Method*.

For the *Composition of the Development Method*, we interviewed the project manager to gather the context in which the development takes place. While the context factors can be easily defined (e.g., marketSize:mass), additional information as constraints (e.g., different customer groups) about the project must be considered to compose the development method. During the composition of the development method, we investigated that working with the patterns can be a challenging task. Here, especially working with the different nested patterns in each other and ensuring an artifact pipeline between input and output artifacts needs some previous experience. In contrast, the composition of models was quite straightforward with the limitation that the knowledge inside the models was just provided by us. However, a subsequent interview with the project manager revealed that the guided composition supports the identification of additional development steps.

For the *Enactment of the Development Method*, we executed the development method in close exchange with the project manager. During the case study, we gathered results for the first three stages because the platform was at this time developed within the fourth stage, and the aim of the study was to develop potential business models for the platform. Here, we followed all development steps of the composed development method but also created additional development steps during the execution. During the execution, the description of all development steps supported a common understanding among all stakeholders. Moreover, the tool provided traceability of the approach while the artifacts supported the collaboration and later transparency. The main results were the different developed canvas models in the third stage. Here, a discussion with the project manager and the culture office employee regarding the different business models revealed their relevance and the possibility of identifying new decisions (e.g., lock-in mechanism). However, to allow the usage of the composition and enactment without external guidance, the self-explanation of the software tool needs to be improved. Overall, the composition and enactment were applicable concerning RQ 2, and

the guidance shows potential both during the composition and enactment. Nevertheless, it is advisable that a skilled *Method Engineer* does the steps of the composition of methods and models instead of the *Business Developer*. Based on that analysis, we interpret those results for future improvements to the solution.

9.1.4 Interpretation

The reasons for our explorative case study during DSR are the evaluation of the second cycle and the derivation of suggestions for further improvement for the third cycle. For that, we analyzed our current threats to validity and derived further implications that we have already addressed during the third cycle.

In order to critically discuss the **Threats To Validity** of our evaluation, we use the framework of Yin et al. [Yin09], who divide the criteria of construct validity, internal validity, external validity, and reliability.

Construct Validity refers to guaranteeing that the most verifiable case study results are based on the RQ. To achieve that, we followed an explorative purpose of conducting both stages of knowledge provision of methods and models as well as composition and enactment of developments that were introduced during the second cycle of DSR. Moreover, we used a real-world case study [RH09] to guarantee the relevance of our results. However, the limitation is that our case study relies on a single unit in one case instead of multiple units in different contexts. Therefore, within the second cycle, we could not directly compare the impact of different contexts on the overall applicability.

Internal Validity refers to the establishment of trustworthiness due to casual relationships during the case conduction. To achieve that, we followed the guidelines of Runeson and Höst [RH09] to use a well-established procedure. Here, by clarifying the overall research design at the beginning of our case study and using the case study protocol as well as the software tool, we ensure transparency of the BMD. However, there is the limitation that we mainly developed the business model with the support of the project manager instead of the business model being independently developed by the project team. Therefore, within the second cycle, we were not able to analyze if the whole concept and the software tool are self-explaining enough for different kinds of stakeholders.

External Validity refers to the extent to which the results can be applied to other cases. To achieve that, we followed the DSR procedure of Vaishnavi and Kuechler [KV08], which aims to abstract the knowledge gained about the concept to solve most cases in the problem class. While this is done for the concept itself, our software tool is just a situated implementation based on the findings of our GLR. Moreover, we were limited by the single-unit case.

Therefore, the knowledge provision must be improved to provide applicability in other unrelated cases.

Reliability refers to the reproducibility of repeating the case study. To achieve that, we used a case study protocol and the traceability feature of our software tool for all stages. Moreover, we used a mix of direct and independent selection methods to increase the objectivity of our study. However, the BMD is a creative task, so repeated conduction by other stakeholders with different knowledge backgrounds can also lead to different results.

Out of the evaluation of the current DSR cycle, we found some **Implications** that we already addressed during the third cycle of DSR. We divide those implications into the three areas of knowledge generalization, complexity reduction, and multiple unit analysis.

For the *Knowledge Generalization*, we had just a base of knowledge for methods and models that focuses on mobile applications and, in particular, on event apps. Moreover, those knowledge repositories had just limited explanations for the elements of the development methods and the canvas models. This, in turn, limits the approach's applicability in other cases. Therefore, in the third cycle, we have extended the knowledge with a particular focus on models and methods for different types of mobile applications.

For the *Complexity Reduction*, we focused on the applicability of both stages of our concept inside the software tool and dismissed a user experience that is easy to understand. Moreover, all design support features of the model repository were applicable at the same time, which decreased the understandability of the design support. This, in turn, limited the usage of the tool by end-users. Therefore, we have increased the usability of the approach and reduced the complexity of our software tool using the concept of modularization.

For the *Multiple Unit Analysis*, we have applied the approach to the case of a local event platform. Here, we also had just a single context to analyze the applicability. This, in turn, limited the evaluation if the approach can be easily transferred to other scenarios. Therefore, we have validated the transferability by creating several business models in different contexts within a user study.

9.2 User Study in Student Courses

To evaluate our approach, we conducted a user study in a lean development seminar and an AR/VR lecture at Paderborn University. Here, within the seminar, we provided an overview of the concept of iterative development of mobile applications, including product features and the business model. During the seminar, we evaluated the third DSR cycle of our situation-specific BMD approach, as proposed in Section 1.2, and the second DSR cycle of our crowd-based prototype validation approach, as proposed in Section 7.3.3. Within the

lecture, we provided an overview of the systematic development of AR/VR applications. Here, we evaluated the third DSR cycle of our crowd-based prototype validation approach. We adapted the guidelines of Runeson and Höst [RH09] of case studies for the user study to increase the quality and reliability of our study. With our user study, we aim to analyze the application of situation-specific BMD in different contexts and the usage of crowd-sourcing for prototype validation. We divide the evaluation into the steps of forming the *Experimental Design* of the evaluation, followed by its *Execution*, the *Analysis* of the results, and their *Interpretation* concerning our approach.

9.2.1 Experimental Design

In the beginning, we need to define the experimental design in which the user study takes place. By adopting the guidelines of Runeson and Höst [RH09], the design can be divided into the definition of the study, the initiation of a user study protocol, and the clarification of ethical considerations.

Definition of the User Study Like in case studies, user studies can be flexibly designed, which makes preliminary and adjustable planning necessary before the actual conduction of the study. Similarly to our case study, we justify our plan based on Robson and McCartan [RM16] who divide it into objectives to achieve, the research question to answer, the theories to concern, the case to study, the methods to rely on, and the selection strategy to use.

The **Objectives**, **Research Questions** and **Theories** are divided through our approaches for situation-specific BMD and the crowd-based prototype validation. For the *Situation-specific Business Model Development*, our refined *Objective* is based on the modularized concept of situation-specific development of business models. Here, our user study aims to apply our concept and the software tool in multiple situations to identify the last issues that need to be resolved for the conclusion of our design study. Based on this refined object, we want to discover our third RQ on how to support the development of artifacts within the BMD steps using flexible adjusted software support (see RQ 3 in Section 1.2). Moreover, we want to improve the knowledge utilization for our approach (see RQ 1) and the composition and enactment of development methods (see RQ 2). Out of those questions, we answer our overall RQ on how to enable the situation-specific development of business models for service providers within software ecosystems (see RQ). For that, we again use the theories of boundary objects [SG89], and opportunity creation [ABA13] of the second cycle. For the *Crowd-based Prototype Validation*, our refined *Objective* is based on our overall solution design of a platform for crowd-based prototype validation. Here, our user study aims to use our situated implementation to identify missing DFs and evaluate our DPs for their

importance. Based on this refined object, we want to discover a separated RQ on how to design software to support the iterative validation of prototypes with the crowd (see Section 7.3.3). Here, we use the same opportunity creation theory as for situation-specific BMD. With both, we have an exploratory purpose of gaining new insights.

Also, the **Case** in our user study can be divided into the situation-specific BMD and the crowd-based prototype validation. For the *Situation-specific Business Model Development*, we use the analysis of multiple units in one case [Yin09] by developing business models for different mobile applications within a lean development seminar. Here, the seminar is part of the Bachelor of Computer Science at Paderborn University in winter term 21/22 and aims to provide an overview of the concept of iterative development of mobile applications. The seminar consists of four phases in which groups of two-three students with no experience in BMD develop the idea of a mobile app. First, a block seminar is used to divide the students into groups, give them an overview of the different steps of lean development, and explain the different situations that can lead to different steps for the developed methods (e.g., mockup vs. wireframe for prototype), and decision for modeling artifacts (e.g., advertisements vs. subscription as revenue streams). Moreover, each student has to present the first idea of a possible mobile app. Second, each group selects one of their ideas, elaborates on the app as a prototype and the business model, and provides an intermediate presentation of their ongoing results. Third, the groups improve their prototypes and business models based on feedback and provide a final presentation of their results. Fourth, the groups must present their conducted process and the developed artifacts in a final report. Here, the groups use our developed *Situational Business Model Developer* as the situated implementation of our situation-specific BMD within the second and third phases.

For the *Crowd-based Prototype Validation*, we use the situated implementation of our *Crowd-based Prototype Validation Platform* with the third phase in our seminar to provide additional development support for our situation-specific approach. Here, one student per group needs to upload their prototype with questions to the platform. Next, every student gives feedback on two predetermined prototypes by answering the questions that could be used to improve the prototypes. Last, the students evaluate the platform's prototype on the platform itself by rating the importance of the DPs together with feedback on the overall idea, the proposed solution, the current drawbacks of the platform, and additional feedback. Moreover, during the lecture in summer term 22/23, we conducted a similar user study just for the platform. Here, within a mini project, three-four students that were grouped needed to develop AR/VR prototypes, which initial idea we evaluated with the platform. Moreover, like in the seminar, each student provides a self-evaluation of the platform.

As **Methods** to collect data and corresponding **Selection Strategy**, we use a mix of indirect and independent methods as characterized by Lethbridge et al. [LSS05]. As an indirect method, we analyze the intermediate and final group results. This includes both given presentations, the submitted final report, and interactions within the software tool. Moreover, within the situated implementation of the platform, we provided a questionnaire for rating the DPs and open questions on the overall idea, the proposed solution, current drawbacks of the platform, and additional feedback. As an independent method, we improve the results of our GLR for the method repository with additional information. Moreover, we use Taxonomy Development (TD) [NVM13] to develop a canvas model repository out of a mix of existing literature and the analysis of existing mobile apps.

Initiation of a User Study Protocol Before starting the conduction of our user study, we need to initiate our user study protocol. Here, like in the case study, we follow the protocol structure of Maimbo and Pervan [MP05]. In the protocol, we store information about the block seminar together with the intermediate and final presentations. Moreover, additional information could be gathered from the traceability features of our software tool, the platform, and the final reports.

Clarification of Ethical Considerations Last, before conducting the user study, we need to consider ethical considerations. This, in turn, increases the trust between the users, here the students, and us. For this, we consider the following things: First, we explain the user study to every user and inform them about the procedure. Second, we state our data collection and storage strategy. Third, we clarify the publication of parts of the study and the omitting of personal information.

9.2.2 Execution

During the conduction, we structure our procedure according to the three applied stages of knowledge provision of methods and models, the composition and enactment of development methods, and the support of development steps.

Knowledge Provision of Methods and Models For the first stage, we needed to provide knowledge about business models for mobile apps for the method repository and the canvas model repository. For the *Method Repository*, we improved the results of our GLR within the case study in Section 9.1 with additional explanations and reference websites to ensure a common understanding by all students. For the *Canvas Model Repository*, we wanted to support the students with knowledge about possible business models and product features.

To gather that knowledge, we used a 3-phases extraction method based on the TD method by Nickerson [NVM13]. The method of Nickerson can be used to classify objects based on their common characteristics. First, we modeled each item of the canvas element for the business model and the product features as a characteristic of a mobile application within the taxonomy. Second, we combined each item with the extracted application domains, corresponding relationships, and identified instances. While we used our extraction method first on 11th February 2019 (see [GRE19c] for technical report), we improved our results on 14th September 2021 before the seminar. To use the method, we needed to define meta-characteristics and ending conditions together with empirical-to-conceptual and conceptional-to-empirical iteration steps. The meta-characteristics are the most comprehensive characteristics that can be used as the basis for the choices in the taxonomy. Based on these meta-characteristics, we were running combinations of empirical-to-conceptional and conceptual-to-empirical iterations. After each iteration, the taxonomy is checked against objective and subjective ending conditions. While objective ending conditions can be assessed, subjective ending conditions leave space for interpretation. We structured our applied extraction method into the three phases of planning the development, conducting the development, and summarizing the development.

During **(1) Planning the Development**, we needed to define the overall meta-characteristics and the ending conditions. To model the items of the business model, we used the nine building blocks of the BMC [OP10] as the most-comprehensive characteristics. We refined these blocks by the categories of the book *Business Model Generation* [OP10] to support the information extraction process. To model the items of the product features, we used a high-level classification of general, user, and interaction features as the most-comprehensive characteristics. The objective ending conditions were the examination of all selected applications and papers for the corresponding execution step. As subjective conditions, we wanted to create an appropriate and cross-application usable canvas model repository.

During **(2) Conducting the Development**, we derived the actual information that should be utilized with our canvas model repository. For that, we defined the three stages of studying existing material, analyzing existing applications, and extracting existing features.

During the *Studying of existing Material*, we analyzed the top listed applications of mobile ecosystems and related apps. Within the conceptual-to-empirical iteration, we analyzed selected literature [FSDN15, GWB10, HO11, LKH17, MVG⁺14, MKM11, RR16, TTP11] from an existing SLR of IT service markets [JPEK16]. Here, we discovered all papers classified as business model relevant and chose the ones that impact mobile developers and their apps. In the empirical-to-conceptual iteration, we looked at the information of 150 apps

from the top lists of Apple's AppStore and Google's PlayStore by discovering the 25 top free, top paid, and top grossing applications. After the elimination of duplicates, there were 126 applications left, which information in the store we have discovered. Out of both iterations, we refined our taxonomies for the business models and the product features.

During the *Analysis of existing Applications*, we conducted a deeper analysis of the product features of selected apps and their business models. For that, we grouped our list of applications into the six categories of messengers, social networks, streaming services, trading services, information processing, and games which are also selected as our application domains. Out of each category, we chose three applications with major differences in their business model that are also selected as our instances. In the conceptual-to-empirical iteration, we reviewed information (papers, analyses, news articles), which we obtained using Google Search. Within the empirical-to-conceptual iteration, we executed the selected apps and analyzed their business models. To support the analysis, we used the patterns of the St. Gallen Business Model Navigator [GFC14] that are also used as instances. Out of both iterations, we refined our taxonomies for the business models and the product features.

During the *Abstraction of existing Features*, we abstracted the characteristics of the business models and product features to create a taxonomy. For that, we derived generic value propositions from the existing value propositions of the first stage and the discovered value propositions of the mobile applications of the second stage. After that, we refined our taxonomies by providing the same hierarchical level for the lowest characteristics.

During **(3) Summarizing the Development**, we combined our gathered information and initiated the *Canvas Model Repository* with the *Canvas Elements*, *Canvas Building Blocks*, and *Canvas Models*. For that, we first defined the *Canvas Elements*. Here, we used the characteristics of both taxonomies as *Items*, the categories of the apps as *Application Domains*, the predefined connections as *Relationships*, and the analyzed apps and identified patterns as *Instances*. Out of that, we defined the *Canvas Building Blocks* by using the hierarchy of the *Items* in the taxonomy combined with existing *Relationships* and *Instances*. Last, we mapped them to *Canvas Models* for the business model and the feature set. A part of our *Canvas Model Repository* can be seen in Figure 9.4. For the *Canvas Elements*, we had exemplary *Application Domains* (e.g., *Mobile Apps*, *Messengers*), *Items* (e.g., *Network*, *Interact with Everyone*), *Relationships* (e.g., *requires*, *excludes*), and *Instances* (e.g., *Spotify*, *Multi-Sided Market*). Those were structured to *Canvas Building Blocks* (e.g., *Mobile App Business Model Knowledge*) with multiple *Applications Domains* (e.g., *Mobile Apps*). Here, the hierarchy of *Items* (e.g., *InteractionType* was decomposed to *Single-Sided Market* and *Multi-Sided Market*), multiple *Relationships* (e.g., *Interact with Everyone supports Multi-Sided Market*), and multiple *Instances* (e.g., *YouTube uses Interact with Everyone and Multi-Sided Market*)

were modeled. Moreover, those blocks were mapped to the *Canvas Models* of the existing *Business Model Canvas* and the newly created *Feature Set Canvas*.

A preliminary version of the whole canvas model repository can be accessed inside the corresponding technical report [GRE19c]. Moreover, the results for both repositories can be accessed in our software tool.

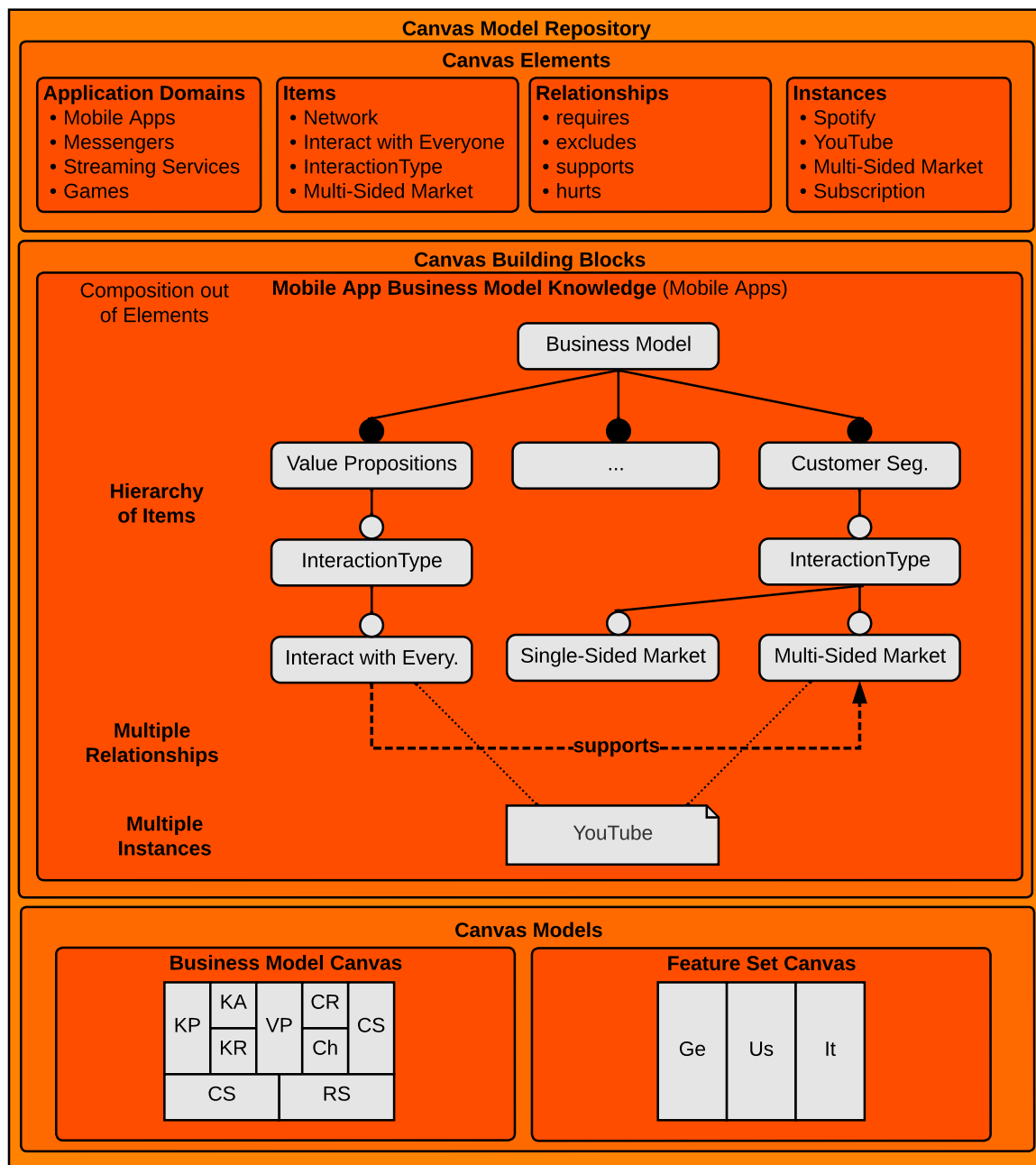


Fig. 9.4 Knowledge Provision: Examples of the Canvas Model Repository based on the TD

Composition and Enactment of Development Methods For the second stage, the 14 students of the seminar were divided into six groups. After every student had presented one idea, each group decided on one of their ideas to work on for the rest of the seminar. These were (1) a recipe app that combines meals out of already purchased food, (2) a student planner app that organizes different courses and submissions, (3) a booking app that allows customers to book appointments at service providers, (4) a social app, that supports the friend matching based on common memes, (5) a clothing app, the recombines clothes from the wardrobe, and (6) a gaming support app, that supports the dungeons-and-dragons roleplay with custom character sheets. The groups used our software tool to compose and enact a situation-specific BMDM to develop business models for their mobile apps. These results of that development were presented within the intermediate as well as the final representation. Moreover, within the final report, they described their conducted development steps and created artifacts.

To use the *Situational Business Model Developer*, we have presented the students the most important tool features during the block seminar together with an exemplary composed development method. With this information, the student groups could take the part of the method engineer to compose a development method or could use our support as method engineer for the composition. For both, we had the constraints that the situational factor of *businessModelingSkills* needs to be set to *low* and our derived *Business Model Initialization Pattern* needs to be used as the first pattern during the initialization. Therefore, the composition and enactment of the development methods could be structured according to the five phases of discovery, analysis, design, development, and validation.

In the **(1) Discovery**, the students discovered the actual problem which they wanted to solve together with the customers they wanted to address. To discover the problem, they often used the *Market Problem Discovery* to gather information about a particular market or the *Own Problem Discovery* to derive the market from their own problems. Here, for example, the group of the booking app used the analysis of the market problem of the time-efficient booking process of regular appointments. Moreover, the group of the recipe app used the observation of the own problem of leftover food that should be reused in new dishes. That discovery was enhanced by additional development steps like *Persona Creation* or *Customer Journal Map Creation*. Here, the group of the clothing app used personas to illustrate potential users of the app, and the group of the booking app used customer journeys to connect contact points of the end-users and the companies. For the discovery of the customers, the *Target Audience Identification* was mostly used as a standard development step. Here, for example, the group of the gaming support app refined their target group as dungeon and dragons players. Moreover, that information was verified through the first

Customer Interviews or *Online Community Observations*. Here, for example, the group of the planner app interviewed other students on their opinions, and the group of the recipe app looked up their market in existing cooking communities. The discovery phase of two different development methods can be seen in Figure 9.5. While the group of the planner app used a single identification of the target audience for the students together with an observation of its own problems, the group of the booking app used two identifications for both market sides together with an observation of the market problem.

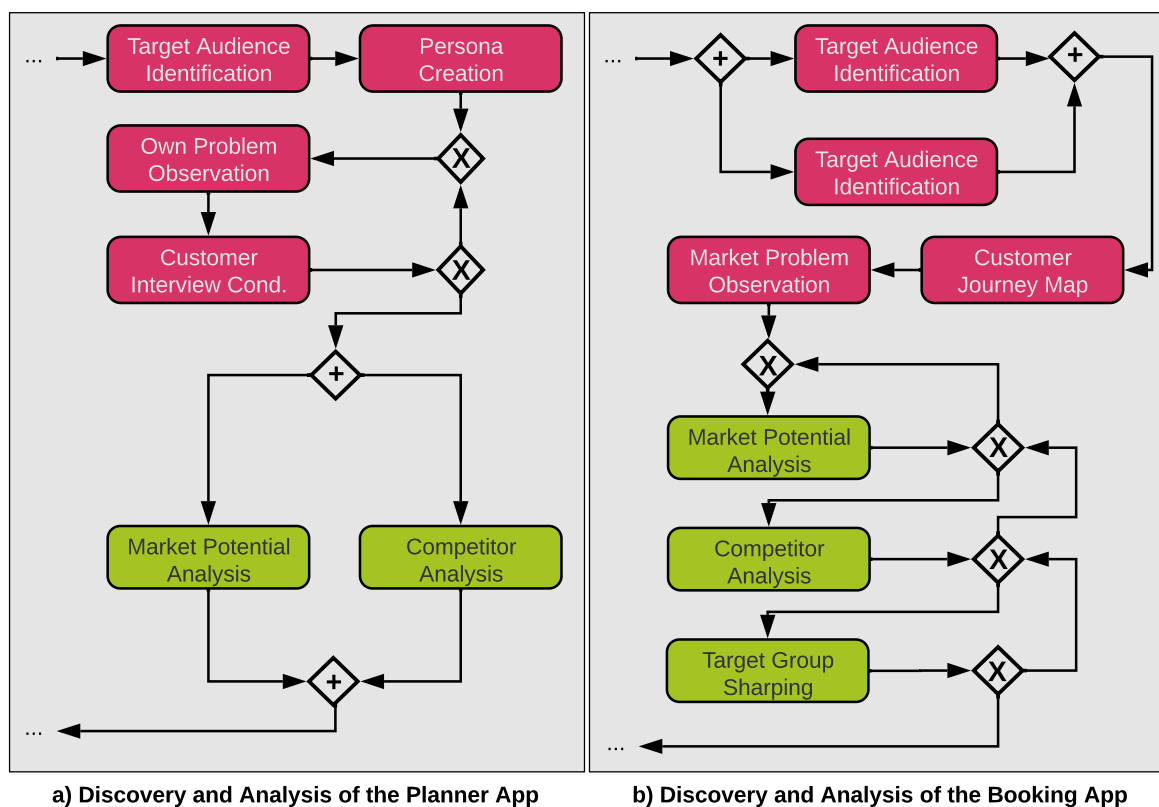


Fig. 9.5 Method Composition: Parts of the BMDMs for the Mobile Apps

In the **(2) Analysis**, the students analyzed the market with the existing competitors they wanted to address. Here, all groups used the *Market Potential Analysis* together with some sort of *Competitor Analysis* or *StoreCompetitor Analysis*. Here, for example, the group of the recipe app analyzed the potential market based on statistics of cooking together with the analysis of existing websites and apps for recipes. Moreover, sometimes the *Target Group Sharpening* was used to refine the target audience after the analysis. For example, the group of the booking app revisited their target audience after a detailed analysis of the market and the competitors. The analysis phase of two different development methods can be seen in Figure 9.5. While the group of the planner app used the parallel conduction of the market

potential analysis and the competitor analysis, the group of the booking app used an iterative sequence of both development steps together with the sharpening of the target audience.

In the **(3) Design**, the students designed the value proposition, business model, and feature set for their mobile apps. For that, they used the required development steps of *Value Proposition Development*, *Business Model Development*, and *Feature Set Development*, where each step is connected to a canvas model and, therefore, supported by the internal *Canvas Module*. While the value proposition was developed without additional knowledge support, the business model and feature set could be supported with the provided knowledge of the canvas model repository. Here the knowledge was used to suggest possible items and patterns, analyze the relationships between those items, and compare the own modeled canvas against other businesses. Here, the groups used the repository to gather possible items for the business model and the feature set. Examples of that can be seen in Figure 9.3 (e.g., *Personalized Recommendations*, *Single-sided Market*). Moreover, the developed business models looked like that if the analyzed relationships were used to improve the possible business models (e.g., solving conformance errors against the knowledge in the repository). However, it seemed that the students did not use the identification of patterns and the comparison against other businesses. For example, parts of the value propositions and customer segments for the cooking app and the gaming support app can be seen in Figure 9.6. Here, both groups used a mix of their own knowledge (e.g., *Young Adults* in *Customer Segments* for cooking app, *Sheet Customization* in *Value Propositions* for game support app) and existing knowledge (e.g., *Personalized Recommendations* in *Value Propositions* for cooking app, *Single-sided Market* in *Customer Segments* for game support app). Moreover, the development steps of *Competitive Advantage Analysis*, *SWOT Analysis*, and *Feature Set Prioritisation* are used each once. Here, the group of the recipe app used analysis of competitive advantage due to many other apps in the same field, and the group of the booking app used feature prioritization due to their large feature set.

In the **(4) Development**, the students developed an artifact of their mobile app. Here, for the first iteration and the intermediate presentation, most groups chose *Mockup Development* for the first artifact. Here, for example, the group of the booking app used Figma to create different mockups of their mobile app. Later, some groups decided to switch to the *Prototype Development*, and one group used a *Wireframe Development*. Here, for example, the group of the gaming support app used the Android SDK to develop the first prototype of their app.

In the **(5) Validation**, the students should validate the artifact of their mobile app with potential users. For the first iteration and the intermediate presentation, most groups chose the *Customer Interviews* with friends as the first validation. Here, for example, the group of the planner app asked their student colleagues for initial feedback on their developed

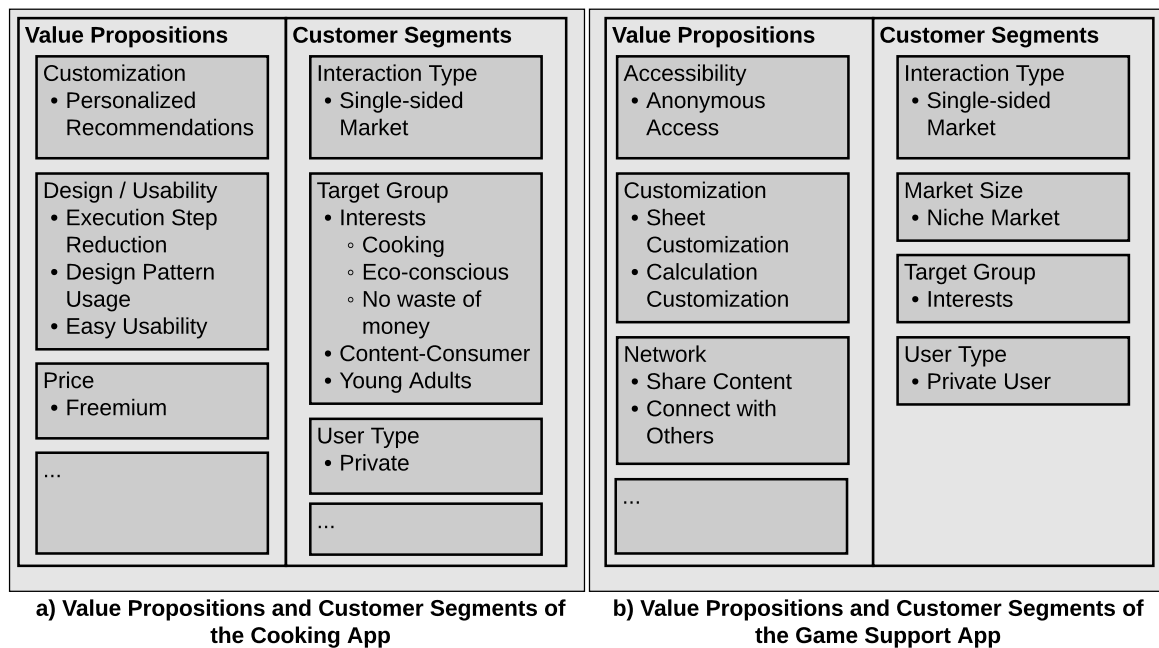


Fig. 9.6 Method Enactment: Parts of the BMC for the Mobile Apps

mockups. Later, for the final evaluation, the *Crowd-based Prototype Validation Platform* was used in the *Crowd Prototype Validation* to cross-validate the artifacts around the seminar as presented in support of the development steps.

Support of Development Steps Within the seminar, the 14 students had less experience in BMD and lean development. Therefore, we aimed to support the development steps of the business model and the feature set with our integrated *Canvas Module* and the validation of the prototype with the external *Crowd-based Prototype Validation Platform*. While the usage of the *Canvas Module* was already explained in the design phase, here we focus on the crowd-validation support.

With the *Crowd-based Prototype Validation Platform*, it is possible to support the validation of prototypes with the crowd. Here, the business developer could upload screenshots or link external prototypes together with a questionnaire. This questionnaire, in turn, could be filled out by potential users of the crowd. Moreover, the results of the questionnaire are aggregated and made available to the business developer to improve his prototype iteratively. In our setting, each group selects one business developer who published a prototype of their mobile app on the platform together with a questionnaire containing the business model (e.g., price offering of the user) and the product features (e.g., missing functionalities) related questions. After that, each student evaluated two prototypes of other groups by filling out their questionnaires. Therefore, each group got feedback from four to six other students that

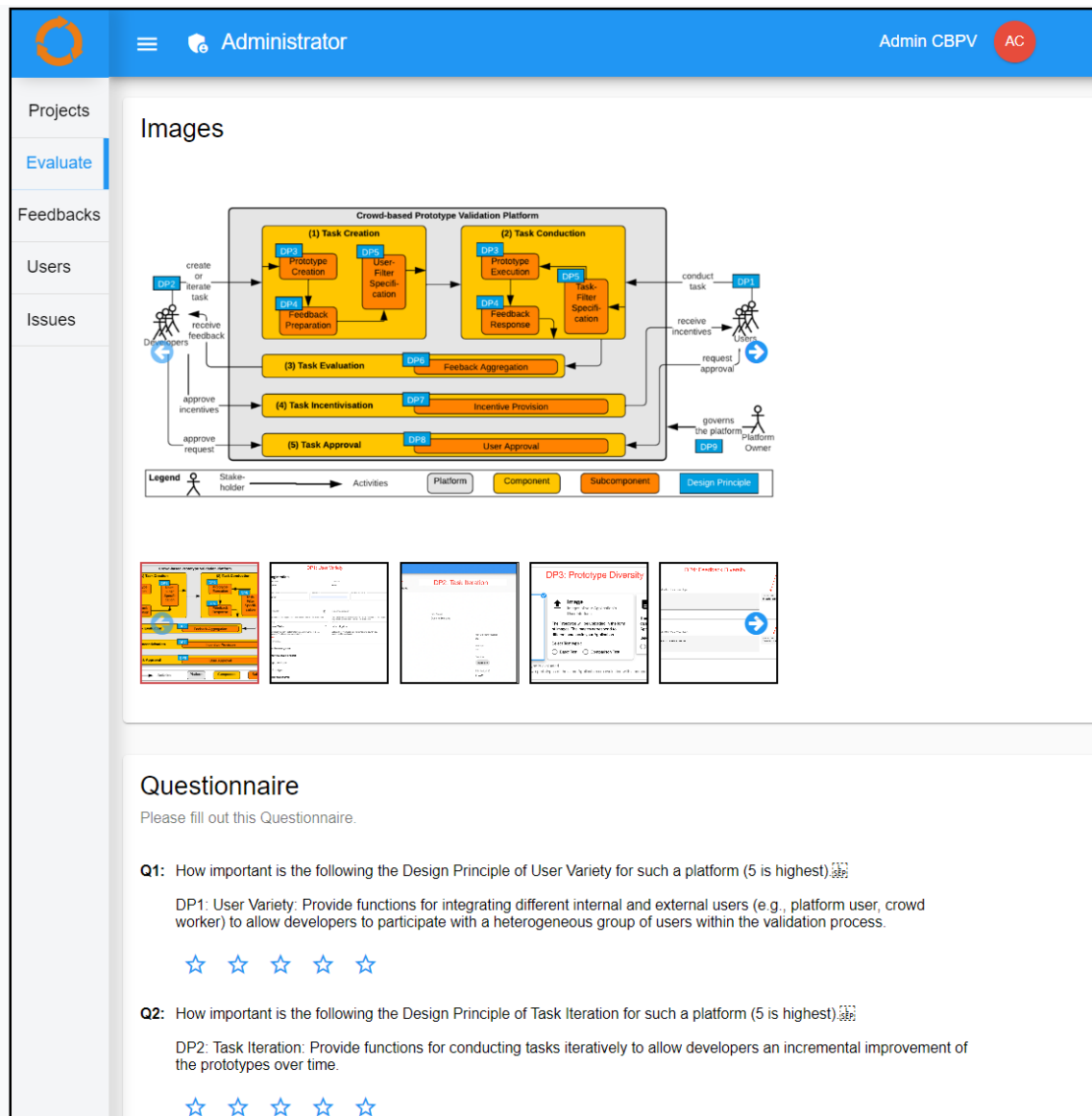


Fig. 9.7 Development Support: Self-evaluation of the Platform

they could use within the validation phase. After that, we provided a self-evaluation of the platform. For that, we have added the platform itself as a prototype, as shown in Figure 9.7. Here, the prototype consisted of images of the platform where the instantiations of the DPs on the platform are labeled. Moreover, the questionnaire provided a 5-star rating question for each DP and four free text questions on the overall idea, the proposed solution, current drawbacks of the platform, and additional feedback.

Moreover, we repeated the study with updated DPs and improved situated implementation in a student lecture on the systematic development of AR/VR applications. Here, the 26

students were divided into eight groups of three to four, where one of each group published an idea (e.g., a JengaVR game, a smARt note app, an ARmomix cooking app) for the application on the platform. After that, all students evaluated two other prototypes and the platform as in the seminar.

9.2.3 Analysis

The first goal of our work was to develop a situation-specific BMD approach. By conducting a design science study, the purpose was to answer the overall RQ of how to enable the situation-specific development of business models (RQ). For the third cycle of DSR, we focus on our third question of supporting the artifact development with software (RQ 3) and refine the first question on how to utilize and store the knowledge of method and models (RQ 1) and the second question on how to compose and enact development methods (RQ 2). Moreover, the second goal was to design a crowd-based prototype validation. By conducting a separate DSR study, the purpose was to answer the RQ of how to design software to support the iterative validation of prototypes with the crowd. We used this software support within our third stage of situation-specific BMD to support development steps.

For the **Utilization and Storage of Knowledge** (RQ 1), we have improved the results of the GLR as an indirect method to create a *Method Repository*. Here, we have added additional explanations to the *Method Elements*, *Method Building Blocks*, and *Method Patterns* to allow a common understanding without external explanations by domain experts. Moreover, we have added explicit execution steps for the modularized support of designing the canvas models in supporting the development steps. Here, understanding the forwarding of the artifacts through the execution steps was tricky. Moreover, for the *Canvas Model Repository*, we have used an extraction method based on TD. Here, identifying the *Canvas Elements* was the major challenge based on the granularity of the items and the analysis of additional sources for the business models of the mobile applications. In contrast to that, the *Canvas Building Blocks* could be directly adopted by the developed taxonomy and mapped to the *Canvas Models* of the existing BMC and the newly developed FSC.

Overall, the utilization and storage of knowledge were applicable concerning RQ 1 and improved in contrast to the second design cycle. However, again a background knowledge of the approach is needed to allow the utilization and storage concerning the reusability aspect. Moreover, by providing the development support for the *Method Building Blocks* also, RQ 3 was applicable to the approach. However, background knowledge of the approach is needed to allow the connection of the execution steps.

For the **Composition and Enactment of Development Methods** (RQ 2), we have improved the composition of the development method with additional explanations and the

enactment of the method with additional collaboration features. For the *Composition of the Development Methods*, we had the six student groups with their own individual development method, which were composed by the groups or us under some restrictions. While for us, the composition was straightforward, the students had some issues regarding the nested usage of the patterns and the additional choices of groups for the stakeholders, artifacts, and tools. However, the issue of nested patterns could be directly related to the first usage of the composition, and the issue of additional choices could be improved with default groups. For the *Enactment of the Development Method*, the six student groups executed their individual methods to develop the business models of their mobile apps. For that, the groups conducted the corresponding development steps. Here, two groups used the collaboration features to discuss the different development steps, three groups used just the traceability feature of the different artifacts, and one group executed the steps but stored the artifact information externally. During that, we got the feedback that the development steps should get more guidance, artifacts should be creatable without development steps, and parts of the *Canvas Module* are hard to understand. During the execution of the execution steps from the *Canvas Module*, the groups used the predefined items and relationships but not the existing instances.

Overall, the composition and enactment of development methods were applicable concerning RQ 2 and improved in contrast to the second design cycle. However, again a background knowledge of the approach is needed to allow the composition of the development method. Overall, the support of development steps (i.e., support of canvas design) was applicable concerning RQ 3. However, the modules' execution steps need to be explained in more detail to allow easy usage.

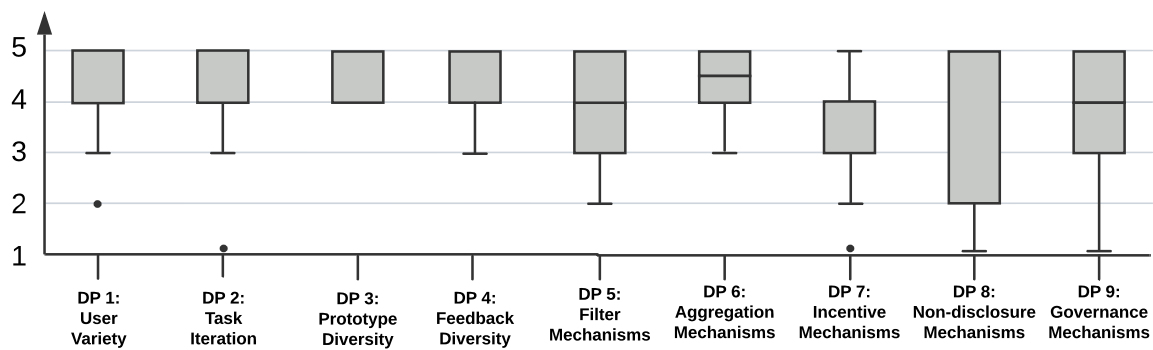


Fig. 9.8 Evaluation of the Design Principles (DPs) on the CBPV platform

For the **Support of Development Steps** (RQ 3), we already analyzed the composition of the development support and their execution for the *Canvas Module* in the last paragraphs. Here, we focus on the analysis of the crowd-based prototype validation, which we divide into

the final quantitative and qualitative results of our questionnaire (n=20) of the third design cycle.

For the *Quantitative Results*, we have answers for the 5-star rating questions for the nine DPs that mostly relate to the abstracted design knowledge of the crowd-based prototype validation. An overview of those results as box plots is shown in Figure 9.8. As an overall impression, we see that nearly every DP is rating as crucial for such a platform. The variety of users (i.e., *DP 1*) and iteration of tasks (i.e., *DP 2*) should be provided by every platform. Also, the diversity of prototypes (i.e., *DP 3*) and feedback (i.e., *DP 4*), together with the aggregation of feedback (i.e., *DP 6*) that are specific for prototype crowd-validation, are rated as essential. The same holds for providing overall governance (i.e., *DP 9*). The function for filtering (i.e., *DP 5*) and incentives (i.e., *DP 7*) are rated lower by the students as they got predetermined prototypes to validate and no additional incentives for the validations. Last, a higher discrepancy exists for the non-closure agreements (i.e., *DP 8*), which some students could interpret as just additional overhead.

For the *Qualitative Results*, we have answers for the free text fields of the additional questions for the concept that are mostly related to the situated implementation of the platform prototype but can partially also be abstracted to the design knowledge. An overall impression of that feedback was that most of the students liked the overall idea of the platform. Just one student was curious if the additional effort would be worth the feedback, and one student commented that crowd validation should be just done in addition to regular customer interviews. Most of the feedback was regarding some general issues with the current version of the platform prototype, like better support for mobile web browsers, UI issues, simplified account management, or bug fixes. However, some feedback also suggested improvements to the most important features of the users, the prototypes, and the feedback. For the users, there was feedback to create groups for collaborative working on the prototypes and invite links to share the prototype with colleagues. For the prototypes, there was feedback to add additional types of non-visual prototypes and directly create clickable mockups on the platform. For the questionnaire, there was feedback for "if not, why" questions and blocks for Likert scale questions. Last, there was the wish to integrate the prototypes and the questions deeper.

Based on that analysis, we interpret those results for the situation-specific BMD and the crowd-based prototype validation for future improvements to the solution.

9.2.4 Interpretation

The reasons for our user study during DSR were the evaluation of the third cycle of our situation-specific BMD approach in different contexts and the third cycle of our crowd-based

prototype validation approach. For that, we analyzed our current threats to validity and derived further implications that we have already addressed within our approaches.

To critically discuss the **Threats To Validity** of our evaluation, we again use the framework of Yin et al. [Yin09], who divide the criteria of construct validity, internal validity, external validity, and reliability.

For *Construct Validity*, we followed an explorative purpose by conducting all three stages of the approach. Here, we use a user study with mobile apps in different contexts to discover the applicability of the whole approach and the self-explanation of the composed development method. However, there is the limitation that for the applicability of the support of development steps, we were just able to test two different exemplary modules from the side of the business developers. This means that we can not directly anticipate the applicability of all possible support modules. Moreover, for the platform, we clarify the goal and purpose to the students and provide additional explanations of the platform together with an email address for solving occurring problems. Nevertheless, there can be misunderstandings of the purpose, especially on the transferability of the DPs to different application areas.

For *Internal Validity*, we transferred the guidelines of Runeson and Höst [RH09] from the case study to the user study to support the comparability of both studies. Here, we clarify the overall research design at the beginning of the study and use the user study protocol, the presentations, the final reports, and the information within the software tool as sources to maximize objectivity and transparency. However, one limitation is that we used a homogeneous group of computer science students with less experience in BMD for the user study. This means we can not directly anticipate the applicability of all possible user groups. Moreover, for the platform, a threat here is the non-systematic literature review in the first cycle. While we cover different areas and use a technique like snowballing to reduce that threat, we can not completely ensure missing some literature. However, those issues should be reduced by conducting multiple design cycles.

For *External Validity*, we followed the design science paradigm of Vaishnavi and Kuechler [KV08] to gain abstract knowledge that can be used to solve a class of problems. Here, we elaborated on the specific problem of developing business models for mobile apps, where we applied our approach in different contexts within the user study. However, there are limitations to the similarities within the specific situations of the students and the application domain of mobile apps. This means that we can not fully anticipate the applicability to all situations and application domains. Moreover, for the platform, a threat here is the evaluation within student courses because of the biased view of the students. Nevertheless, this should less affect the DPs due to the additional interviewing of experts.

For *Reliability*, we maximize the traceability by using a user study protocol, the presentations, the final reports, and the information within the software tool. Moreover, we used a mix of direct and independent selection methods to increase the objectivity of our study. However, there is the limitation that the provision of knowledge repositories and the development of business models are creative processes that highly depend on the actual stakeholders. This means that the repetition of the knowledge provision and business model development will also lead to different results. Moreover, for the platform, we record the whole expert workshop and export the raw data of all data created in the two user studies. While this increases the reliability of the study result, it could also harm the experts and students in providing negative feedback.

Out of the evaluation of the third DSR cycle, we found some **Implications** that we divide into the situation-specific BMD and the crowd-based prototype validation.

For the *Situation-specific Business Model Development*, we directly addressed those implications during the conclusion of our design science study. However, further improvements are presented in the future work of our thesis, as presented in Section 10.3. We divide those implications into the three areas of guidance improvements, approach complexity reduction, and tool complexity reduction.

For the *Guidance Improvements*, we noted that the students with less business model experience had issues with the given guidance. This includes the different functionalities of the execution steps of the canvas module but also the high-level descriptions for the tasks without separate execution steps. This, in turn, also limits the applicability to similar stakeholders with less business model experience. Therefore, we improved the canvas module's functionalities and extended tasks without modules with additional task steps.

For the *Approach Complexity Reduction*, we found out that the students had issues with the predefined composed method during the enactment. This includes the composition itself and the need for method building blocks to create the artifacts. This, in turn, limits the flexible usage by the stakeholders. Therefore, we reduced the complexity by allowing the BMD without a predefined composed development method, phase-based construction, and the direct creation of artifacts.

For the *Tool Complexity Reduction*, we investigated whether the students had an overall usability problem with the software tool. This includes the explanations for the different stages but also the layout of the tool. This, in turn, limits the acceptance of the stakeholders. Therefore, we improved the explanations and the layout of the software tool.

For the *Crowd-based Prototype Validation*, we divide those implications into the abstracted design knowledge of the DPs and the situated implementation of the platform.

The *Abstracted Design Knowledge* refers to the developed design principles and the overall solution design. We currently see no major issues with the current set of principles. However, some DPs could be slightly improved in the future. For the variety of the users (i.e., *DP 1*), the external users, and in the iteration of tasks (i.e., *DP 2*), the incremental improvements could be described more precisely. Moreover, the deeper integration of prototypes (i.e., *DP 3*) and feedback (i.e., *DP 4*) could be mentioned. Next, the reasoning for the incentivization (i.e., *DP 7*) and the non-disclosure agreements (i.e., *DP 8*) could be improved. Last, based on the analysis of the created prototypes, a minor design principle that could be investigated in the future would be providing guidance in creating a task (e.g., choosing the best type of prototype, generating good questions for feedback).

The *Situated Implementation* refers to the design features and the implemented platform prototype. Here, we currently see a need to fix the current bugs that were identified by the students. Moreover, we want to work on specific features that were mentioned during the evaluation. For that, in addition to single sign-on services (i.e., *DF 2*), we want to allow the sending of invitation links for concrete task evaluations. Moreover, we want to allow the sharing of prototypes with other developers at the task creation (i.e., *DF 5*). To improve the diversity of the prototypes, we want to add an internal prototyping tool in addition to the external one (i.e., *DF 10*). Furthermore, the diversity of the feedback should be supported by multi-questions based on Lickert scales (i.e., *DF 14*). As a larger project, we want to combine the creation of prototypes and the provision of questions deeper based on the integrated prototyping tool. Last, we want to implement the guidance in task creation, as mentioned as a possible DP above.

9.3 Summary

Within this chapter, we have provided the evaluation of our second and third design science cycle. For that, we have conducted a case study on developing possible business models for the OWL Live event platform and a user study on developing possible business models and prototypes within students' courses. For both studies, we have shown the overall experimental design, the execution of the study, the analysis of the results, and their interpretation. Moreover, we provided an evaluation of the crowd-based prototype validation that was integrated into our third design cycle.

For the **Case Study on OWL Live**, we developed possible business models for a local event platform. Here, we conducted a GLR to provide the knowledge for the method repository and the canvas model repository. Out of that knowledge and with the support of the project manager, we have composed a situation-specific BMDM. By enacting that

development method, we have developed the three different business models of a content aggregator, a ticket seller, and a sponsored platform, together with a swot analysis of each business model.

For the **User Study in Student Courses**, we organized a student seminar in which the groups of students developed business models and mobile app prototypes iteratively. Here, we improved our GLR to provide the knowledge of the method repository and conducted a TD to provide the knowledge for the canvas model repository. Out of that knowledge and with additional support from us, the student groups have composed their situation-specific BMDMs. By enacting the development methods, the students have developed the business models and prototypes of a recipe app, a student planner app, a booking app, a social app, a clothing app, and a gaming support app. Moreover, we showed the evaluation of our crowd-based prototype validation within a student lecture on developing AR/VR applications.

Based on the evaluation, we conclude our approach in the next chapter. For this, we summarize the main contributions of our approach, revisit the stated HRs and point out future work.

Chapter 10

Conclusion and Future Work

In the previous chapter, we evaluated our solution based on a case and a user study. Based on that, we give a conclusion on our approach for the situation-specific business model development within software ecosystems and show additional research points which can be addressed in the future. For that, we first summarize the main contributions of our approach (10.1). After that, we revisit the extracted high-level requirements behind our solution (10.2). Last, we point out research work that could be addressed in the future (10.3).

10.1 Contribution Summary

The development of new and innovation of existing business models is a challenging task for both startups and companies. Here, the steps in the development method, as well as the information in the modeling artifacts, need to fit the context in which the business model should be developed. For that, we support both the development method and the modeling artifacts with the utilized knowledge of different domain experts. Here, the knowledge is selected situation-specific to the changeable context. To solve that challenge of context awareness, we have developed an approach for the situation-specific development of business models. Based on DSR, we conducted three design cycles which we evaluated with a feasibility study, a case study, and a user study. The outcomes of this research are the operational principles of the evaluated concept of situation-specific BMD together with the situated implementation of a software tool called *Situational Business Model Developer*. Our concept consists of three stages. In the first stage, we utilize and store the knowledge of different development methods and supporting modeling artifacts in two repositories. In the second stage, we compose the development method out of both repositories based on the context and enact that development method to develop different artifacts, including the business model. In the third stage, we support single development

steps within those development methods with customizable development support to assist the artifact development. In the following, we summarize the main contributions of each stage.

The goal of the **Knowledge Provision of Methods and Models** is to utilize and store reusable knowledge about development methods and modeling artifacts from domain experts. To solve that, we provide a *Method Repository* for the method fragments of the development methods and a *Canvas Model Repository* with canvas modeling fragments for the canvas modeling artifacts. As method fragments, we have the atomic *Method Elements*, including artifacts and tools, that are combined to *Method Building Blocks* that transform input artifacts into output artifacts. Moreover, the order of those building blocks is optionally structured by *Method Patterns*. As canvas modeling fragments, we have the atomic *Canvas Elements*, including items and relationships, that are hierarchically refined as *Canvas Building Blocks*. Moreover, those building blocks are visually represented through *Canvas Models*. Last, both repositories are connected by canvas artifacts that relate to specific canvas models. That knowledge is made reusable to compose different development methods in the next stage.

The goal of the **Composition and Enactment of Development Methods** is to construct and execute development methods based on freely definable and changeable contexts. For the construction of the development method, we introduce pattern-based and phase-based construction. In the pattern-based construction, we nest different *Method Patterns* into each other and inserted *Method Building Blocks* in them. In the phase-based construction, we divide the development method into different phases and select *Method Building Blocks* for them. In both constructions, the *Method Building Blocks* are recommended by comparing the given situation of the organization and the defined situation of the building block. Moreover, we connect each canvas artifact to a *Canvas Model*. Here, we support the consolidated usage of different *Canvas Building Blocks*. For that, we introduce the feature-based and taxonomy-based consolidation of those building blocks. For the feature-based consolidation, we merge the different building blocks with their items and relationships, detecting conflicts of relationships between items of different building blocks. For the taxonomy-based consolidation, we support just items that can be merged without conflicts between different building blocks. Based on that, we provide an execution engine for the composed development methods or new ad-hoc created development methods. Here, the *Method Building Blocks* of the development method are interpreted as development steps with guidelines for their conduction. During this conduction, different stakeholders collaborate to create and modify different (*Canvas*) *Modeling Artifacts*, including the business model. Moreover, each conduction of a development step might lead to a change of the context and an adaption of the development method. Last, specific development steps are supported by IT assistance in the next stage.

The goal of the **Support of Development Steps** is to provide software assistance for specific development steps during the BMD. To provide that assistance, we introduce the usage of *Support Modules* for providing such flexible development support. In the beginning, we need to connect each module to the process engine. Here, each module might contain different *Support Tools* with atomic *Support Steps*. Those support steps are used to create and modify different *Support Artifacts* that are specified through *Support Meta Artifacts*. Those meta artifacts, in turn, are used to ensure the interpretability of the support artifact by support tools of different support modules. After that, we compose the development support for specific *Method Building Blocks* by combining different *Support Steps* and choosing the *Support Artifacts* as input and/or output artifacts of the building block. After that, we enact the development support by executing the development step that is connected to the *Method Building Blocks*. Here, each *Support Step* is connected to a customized code block in the *Support Module* to support the creation and modification of the *Support Artifacts*. To show the applicability of our solution, we provide three different support modules. The first one is the *Canvas Module*, which provides design support for canvas models based on the *Canvas Model Repository*. The second one is the *HypoMoMap Module*, which provides validation support for the assumptions of business models and product features. The third one is the *CBPV Platform*, which provides validation support for software prototypes using the crowd for feedback.

10.2 High-level Requirements Revisited

For developing our situation-specific BMD approach, we have conducted a review of the literature in BMD and an analysis of tools on BMDSSs in Section 1.3 to derive the underlying HRs as a foundation of our solution. During this section, we revisit those nine requirements and their fulfillment within the approach.

The **HR 1: Knowledge Utilization** states that the solution should allow the utilization of knowledge about BMDMs and within the canvas artifacts. Within our approach, we utilize the knowledge from different domain experts about methods to use and the canvas artifacts to rely on with the support of a *Method Repository* and a *Canvas Model Repository* that both interrelate to each other. Here, the utilized knowledge can be used to compose a development method and provide suggestions during the enactment during the conduction of development steps together with supporting single development steps with design support.

The **HR 2: Method Comprehensiveness** states that the solution should allow the comprehensive development of BMDMs for all phases. Within our approach, we support all phases of BMD by providing a *Method Repository* and the support for pattern-based and

phase-based construction of the development method. For the pattern-based construction, the nesting of *Method Patterns* ensures comprehensiveness. For the phase-based construction, the selection of the phases leads to it. Moreover, during the enactment, it is possible to flexibly execute additional *Method Building Blocks* as development steps for supporting those phases.

The **HR 3: Model Visualisation** states that the solution should allow visual representations of the business model. With our approach, we visualize the business model by providing a *Canvas Model Repository*, where those models can be flexibly represented through *Canvas Models*. Those *Canvas Models*, in turn, can be used during the composition to consolidate different *Canvas Building Blocks* and the enactment to visualize the structure behind the consolidated *Canvas Building Blocks*. Moreover, during the support of development steps, the instances of the items might be used to provide additional design support.

The **HR 4: Context Awareness** states that the solution should be aware of the context in which the business model is developed. Within our approach, we integrate the context by defining the *Situational Factors* in the *Method Repository* and the *Application Domains* in the *Canvas Model Repository*. During the composition of the development, we use the *Situational Factors* to choose the *Method Building Blocks* and optionally *Method Patterns* or *Phases* together with the *Application Domains* to choose the *Canvas Building Blocks*. Here, fragments of both repositories are connected to support the development methods as well as the canvas modeling artifacts. Moreover, that context can be flexibly adjusted during the enactment of the development method to provide an adaptation of methods and models.

The **HR 5: Selection Assistance** states that the solution should assist the development of business models with the selection of BMDMs and canvas artifacts. Within our approach, we assist in selecting knowledge from the *Method Repository* and the *Canvas Model Repository* based on the given context. During the composition, we guide the construction of the development method out of the *Method Building Blocks* and optionally *Method Patterns* or *Phases* and the consolidation of the canvas models from the *Canvas Building Blocks*. Moreover, during the enactment, we support the conduction of development steps with knowledge of the *Method Building Blocks* and the development of canvas artifacts with knowledge of the *Canvas Building Blocks*. *Support Modules* can further improve the development of (canvas) artifacts during the support of development steps.

The **HR 6: Development Continuity** states that the solution should allow changes in the business model and the business model development method during the whole development. Within our approach, we provide continuity for the development method and the canvas model during the whole development. For the development methods, we allow the adaptation of the development method based on changes in the context after the execution of every development

step. For the canvas models, we allow the creation and modification of new canvas artifacts during each development step. Moreover, both the execution of new development steps and the creation of new canvas artifacts is possible during the development.

The **HR 7: Stakeholder Collaboration** states that the solution should allow the collaboration of different stakeholders during the BMD. Within our approach, we support the collaboration and communication of different stakeholders. Here, we allow the definition of stakeholders responsible for specific tasks within the *Method Building Blocks*. During the conduction of development steps, the information about involved stakeholders might be directly used to assign them to tasks. Moreover, different stakeholders could communicate within each development step using a discussion board and collaborate in developing different (*Canvas*) *Artifacts*.

The **HR 8: Artifact Management** states that the solution should provide management of all artifacts that are created during the BMD. Within our approach, we provide that management by tracing the creation and modification of all artifacts. For that, general *Artifacts* are defined in the *Method Repository*, while specific *Canvas Artifacts* are defined in the *Canvas Model Repository*. Moreover, within the *Support Modules*, additional *Support Meta Artifacts* can be defined to manage the instantiated *Support Artifacts*. During the conduction of development steps, existing artifacts are used as inputs or outputs of each step. Moreover, different *Support Tools* with *Support Steps* might be used to support the creation and modification of those artifacts.

The **HR 9: Decision Support** states that the solution should provide decision support during BMD. Our approach provides different development support techniques with the *Support Modules*. Here, we allow the composition of *Support Steps* of *Support Tools* within the *Method Building Blocks* that are used during the conduction of development steps within the enactment of the development method. For that, we provide the initial modules for designing the canvas with the *Canvas Module*, validating hypotheses with the *HypoMoMap Module*, and validating prototypes with the *CBPV Platform*.

The **HR 10: Tool Support** states that the solution should provide a software tool to support the BMD. Within our approach, we developed a situated implementation of our approach called *Situational Business Model Developer* and published it under open-source for future extensions. Within the tool, we provide support for all three stages consisting of the knowledge provision of methods and models, the composition and enactment of development methods, and the support of development steps. Moreover, the tool can be flexibly extended using additional *Support Modules* for single development steps.

10.3 Future Work

Within this thesis, we have presented a situation-specific BMD approach based on DSR. During the work on the thesis, we identified different points for extensions of the stages of our approach as well as concrete ideas for follow-up work that could be addressed in future design cycles. We divide those extensions into the stages of knowledge provision, composition and enactment, and development support together with their evaluation and follow-up work in modeling business ecosystems, tailoring business model workshops, simulating business outcomes, and connecting source code.

For the **Knowledge Provision Extensions**, we currently base the structuring of our method repository on BPMN and the canvas model repository on canvas models that were both initialized with knowledge from domain experts during the evaluations. However, here both the structuring of the repositories and the initialization of knowledge could be extended in the future. While additional lightweight modeling languages like Case Management Model and Notation could be discovered for the structuring of the method repository, model repositories could provide more complex modeling languages like the e3-Value Model for modeling also value networks around the business models. Moreover, for the initialization of knowledge, additional sources for method and modeling artifact knowledge in specific areas like business ecosystems or digital platforms could be derived and provided within the tool. That knowledge could be gathered manually by conducting SLRs, GLRs, and expert interviews, or (semi)automatic by analyzing developed business models or mining existing business resources.

For the **Composition and Enactment Extensions**, we currently compose the method out of patterns and phases together with the modeling artifacts out of feature models and taxonomies. Both are enacted on a Kanban board and (canvas) modeling artifacts. However, simplifications of both the methods and modeling artifacts during the composition and enactment could be discovered in the future. While, for the composition of models, the usage of configuration-based situational method engineering with configuration points for the different goals and phases could be discovered, the modeling artifacts could use a simplified library without hierarchies if the enhanced design support and refinements for the canvas models are not needed. This, in turn, can also lead to the potential effect that the enactment might be directly combined with the composition in the most simplified cases, like for our defined ad-hoc creation of development methods. Integrating both makes it possible to adjust better the roles of the method engineer and the business developer.

For the **Development Support Extensions**, we currently provide simplified step-wise design support together with three exemplary support modules to show its applicability. However, both the step-wise design support and the number of modules could be extended

in the future. For the step-wise design support, the composition could be extended so that more complex executions sequences of steps and a more fine-granular selection of modeling artifacts is possible. This, in turn, improves the possible use cases of the approach but also increases the complexity of integrating new modules into the software tool. Moreover, it would be possible to add modules to support all different phases of BMD. Ideas for such modules could be derived from our SLR on BMDSSs. Here, examples would be discovering business model information from existing ERP systems or validating possible business models by creating digital twins.

For the **Design Evaluation Extensions**, we currently evaluate the applicability of our approach using a feasibility study, a case study, and a user study. However, in the future, the applicability of using other evaluation techniques and the measuring of improvements could be discovered in additional design cycles. For applicability, it would be possible to conduct expert workshops with companies and startups or prototype presentations on online events and trade fairs to gather industrial feedback on our approach. For measuring improvements, it would be possible to conduct small and locked controlled experiments of single features of our approach against other existing approaches like pen-and-paper workshops or simple visualization tools.

For the **Follow-Up Work: Modeling of Business Ecosystems**, it could be a future investigation to model more complex networks of business ecosystems with different stakeholders through our approach. For that, we first need to develop a (lightweight) modeling language to design such business ecosystems. This can be done by extending our canvas model repository to present modeling language models or developing an additional meta artifact within a support module. After that, both the existing method repository and the extended canvas model repository can be filled with knowledge on the development methods and modeling artifacts of business ecosystems. Depending on the developed modeling artifact, the composition and enactment of the models need also be changed before the usage of the models is possible. Moreover, developing an additional support tool within a support module to support the different development phases of business ecosystems with unique collaboration and communication features among different organizations could make sense. We already presented the idea behind this follow-up work in [VGK⁺22].

For the **Follow-Up Work: Tailoring of Business Model Development Workshops**, it would be worth an investigation to adapt our approach for different online and offline workshop settings. Here, different stakeholders simultaneously work on developing different ideas for business models. For that, we need to extend our software tool with a user interface for large and interactive screens for the offline setting and live synchronization and real-time communication for the online setting. Moreover, we need to develop an extension for our

repositories or create a new meta artifact to provide visual representations like template boards of the single development steps. Those visualizations, in turn, should also be used during the enactment of the development process. Furthermore, gamification elements within a support tool of a support module could support single development steps. We already presented the idea behind this follow-up work in [GYNE22d].

For the **Follow-Up Work: Simulation of Business Outcomes**, a future investigation would be to combine our reference modeling using feature models to (semi-) automatically simulate the outcome of different variants of business models. For that, we need to extend the feature representation in our canvas model repository or create a new meta artifact to deal with an advanced calculation language like system dynamics or agent-based modeling. During the enactment, the space of possible business models could be chosen by selecting a subset of all features that should be investigated. To calculate different business models simultaneously, a support tool of a support module for development support needs to be developed where different parameters for the calculation can be set. We already presented the idea behind this follow-up work in [VG21].

For the **Follow-Up Work: Connection to Source Code**, it would be beneficial to investigate if our model-based approach can be turned into a model-driven approach by directly connecting our developed models to the source code of a software product. With that, split tests of the product features and the business model can be directly tested in possible prototypes by adjusting those to the formal representation. For that, we need to develop a meta artifact that can be used within a software product. Moreover, we need to develop an external code connection tool within a support module to achieve split-testing, receiving results, and further analysis of the results. We already presented the idea behind this follow-up work in [GYE22].

References

- [ABA13] Sharon A. Alvarez, Jay B. Barney, and Philip Anderson. Forming and Exploiting Opportunities: The Implications of Discovery and Creation Processes for Entrepreneurial and Organizational Research. *Organization Science*, 24(1):301–317, 2013.
- [ABKS13] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines*. Springer, Heidelberg, 2013.
- [ADEHA08] Mutaz Al-Debei, Ramzi El-Haddadeh, and David Avison. Defining the Business Model in the New World of Digital Business. In *Proceedings of AMCIS 2008*. 2008.
- [AdR18] Alexia Athanasopoulou and Mark de Reuver. Designing business model tooling for business model exploration: An experimental design for evaluation. In *Proceedings of BLED eConference 2018*, pages 477–489. University of Maribor Press, 2018.
- [AdR20] Alexia Athanasopoulou and Mark de Reuver. How do business model tools facilitate business model exploration? Evidence from action research. *Electronic Markets*, 30(3):495–508, 2020.
- [ADv13] Petra Andries, Koenraad Debackere, and Bart van Looy. Simultaneous Experimentation as a Learning Strategy: Business Model Development Under Uncertainty. *Strategic Entrepreneurship Journal*, 7(4):288–310, 2013.
- [AF17] Dominik Augenstein and Christian Fleig. Exploring Design Principles for a Business Model Mining Tool. In *Proceedings of ICIS 2017*. AIS, 2017.
- [AF18] Dominik Augenstein and Christian Fleig. Towards Increased Business Model Comprehension - Principles for an Advanced Business Model Tool. In *Proceedings of ICIS 2018*. AIS, 2018.
- [AFD18] Dominik Augenstein, Christian Fleig, and Dominik Dellermann. Towards Value Proposition Mining - Exploration of Design Principles. In *Proceedings of ICIS 2018*. AIS, 2018.
- [AFM18] Dominik Augenstein, Christian Fleig, and Alexander Maedche. Development of a Data-Driven Business Model Transformation Tool. In *Designing for a Digital and Globalized World*, volume 10844, pages 205–217. Springer, Cham, 2018.

- [AG03] J. M. Akkermans and Jaap Gordijn. Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, 2003.
- [AHdR18a] Alexia Athanasopoulou, Timber Haaker, and Mark de Reuver. Designing digital tooling for business model exploration for the Internet-of-Things. In *Proceedings of DESRIST 2018*. 2018.
- [AHdR18b] Alexia Athanasopoulou, Timber Haaker, and Mark de Reuver. Tooling for Internet-ofThings Business Model Exploration: A Design Science Research Approach. In *Proceedings of ECIS 2018*. 2018.
- [AIB07] Jörn Altmann, Mihaela Ion, and Ashraf Adel Bany Mohammed. Taxonomy of Grid Business Models. In *Grid Economics and Business Models*, volume 4685, pages 29–43. Springer, Cham, 2007.
- [ALB⁺09] Shail Arora, Gary Leavens, Bernd Bruegge, Yvonne Coady, and Simon Peyton-Jones, editors. *Proceeding of the 24th ACM SIGPLAN conference companion*. ACM, 2009.
- [All00] Verna Allee. Reconfiguring the value network. *Journal of Business Strategy*, 21(4):36–39, 2000.
- [AM17] Dominik Augenstein and Alexander Mädche. Exploring Design Principles for Business Model Transformation Tools. In *Proceedings of ICIS 2017*. 2017.
- [App21] App Annie Inc. *The State of Mobile 2021*. 2021.
- [ATO⁺20] O. Ege Adali, Oktay Türetken, Baris Ozkan, Rick Gilsing, and Paul Grefen. A Multi-concern Method for Identifying Business Services: A Situational Method Engineering Study. In *Enterprise, Business-Process and Information Systems Modeling*, volume 387, pages 227–241. Springer, Cham, 2020.
- [BBS10] Jan Bosch and Petra Bosch-Sijtsema. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67–76, 2010.
- [BCW17] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-Driven Software Engineering in Practice: Second Edition. *Synthesis Lectures on Software Engineering*, 3(1):1–207, 2017.
- [BdRHF20] Harry Bouwman, Mark de Reuver, Marikka Heikkilä, and Erwin Fielt. Business model tooling: where research and practice meet. *Electronic Markets*, 30(3):413–419, 2020.
- [BdRS⁺12] Harry Bouwman, Mark de Reuver, Sam Solaimani, Dave Daas, and Timber Haaker. Business Models Tooling and a Research Agenda. In *BLED 2012 – Special Issue*. 2012.
- [Béz05] Jean Bézin. On the unification power of models. *Software & Systems Modeling*, 4(2):171–188, 2005.

- [Béz06] Jean Bézivin. Model Driven Engineering: An Emerging Technical Space. In *Generative and Transformational Techniques in Software Engineering*, volume 4143, pages 36–64. Springer, Heidelberg, 2006.
- [BH95] Sjaak Brinkkemper and Frank Harmsen. Design and implementation of a method base management system for a situational CASE environment. In *Proceedings of the Asia Pacific Software Engineering Conference*, pages 430–438. IEEE Comput. Soc. Press, 1995.
- [BHH⁺18] Harry Bouwman, Jukka Heikkilä, Marikka Heikkilä, Carlo Leopold, and Timber Haaker. Achieving agility using business model stress testing. *Electronic Markets*, 28(2):149–162, 2018.
- [BHP00] Phillip A. Bernstein, Alon Y. Halevy, and Rachel A. Pottinger. A vision for management of complex models. *ACM SIGMOD Record*, 29(4):55–63, 2000.
- [Bla13] Steve Blank. Why the lean start-up changes everything. *Harvard Business Review*, (91):63–72, 2013.
- [Bla20] Steve Blank. *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. JohnWiley, Boston, 5th edition edition, 2020.
- [BM14] Steve Boßelmann and Tiziana Margaria. Domain-Specific Business Modeling with the Business Model Developer. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, volume 8803, pages 545–560. Springer, Heidelberg, 2014.
- [BM17] Steve Boßelmann and Tiziana Margaria. Guided Business Modeling and Analysis for Business Professionals. In *Service Business Model Innovation in Healthcare and Hospital Management*, pages 195–211. Springer, Cham, 2017.
- [BO20] David J. Bland and Alexander Osterwalder. *Testing business ideas*. John Wiley & Sons, Hoboken, 2020.
- [Bri96] Sjaak Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280, 1996.
- [BS07] Katrin Burmeister and Christian Schade. Are entrepreneurs’ decisions more biased? An experimental investigation of the susceptibility to status quo bias. *Journal of Business Venturing*, 22(3):340–362, 2007.
- [BSRP16] Hans Berends, Armand Smits, Isabelle Reymen, and Ksenia Podoynitsyna. Learning while (re)configuring: Business model innovation processes in established firms. *Strategic organization*, 14(3):181–219, 2016.
- [BWRP15] Michael Blaschke, Maurus L. Wuetherich, Uwe V. Riss, and Petros Papakostas. Business Model Management System Design: Manifesto, Requirements, and Prototype. In *Proceedings of the 3rd International Conference on Enterprise Systems (ES)*, pages 153–160. IEEE, 2015.
- [CBI19] CBInsights. *CB Insights 2019: Top 20 Reasons Why Startups Fail*. 2019.

- [Che10] Henry Chesbrough. Business Model Innovation: Opportunities and Barriers. *Long Range Planning*, 43(2-3):354–363, 2010.
- [CN09] Paul Clements and Linda Northrop. *Software product lines: Practices and patterns*. Addison-Wesley, Boston, 7th edition, 2009.
- [CN18] Federico Cosenz and Guido Noto. A dynamic business modelling approach to design and experiment new business venture strategies. *Long Range Planning*, 51(1):127–140, 2018.
- [DEWL13] Christina Di Valentin, Andreas Emrich, Dirk Werth, and Peter Loos. Business Modeling in the Software Industry: Conceptual Design of an Assistance System. In *Practice-Driven Research on Enterprise Transformation*, volume 151, pages 34–45. Springer, 2013.
- [DFH16] Fabiano Dalpiaz, Xavier Franch, and Jennifer Horkoff. *istar 2.0 language guide*, 2016.
- [DHOB13] Dave Daas, Toine Hurkmans, Sietse Overbeek, and Harry Bouwman. Developing a decision support system for business model design. *Electronic Markets*, 23(3):251–265, 2013.
- [DLE17] Dominik Dellermann, Nikolaus Lipusch, and Philipp Ebel. Developing Design Principles for a Crowd-Based Business Model Validation System. In *Designing the Digital Transformation*, volume 10243, pages 163–178. Springer, 2017.
- [DLEL19] Dominik Dellermann, Nikolaus Lipusch, Philipp Ebel, and Jan Marco Leimeister. Design principles for a hybrid intelligence decision support system for business model validation. *Electronic Markets*, 29(3):423–441, 2019.
- [DLL18] Dominik Dellermann, Nikolaus Lipusch, and Mahei Manhai Li. Combining Humans and Machine Learning: A Novel Approach for Evaluating Crowdsourcing Contributions in Idea Contests. In *Proceedings of Multikonferenz Wirtschaftsinformatik 2018*. 2018.
- [dRAH⁺16] Mark de Reuver, Alexia Athanasopoulou, Timber Haaker, Melissa Roelfsema, and Angelika Riedl. Designing an ICT tooling platform to support SME business model innovation: Results of a first design cycle. In *Proceedings of BLED eConference 2016*. 2016.
- [DWL15] Christina Di Valentin, Dirk Werth, and Peter Loos. Analysis of IT-Business Models Towards Theory Development of Business Model Transformation and Monitoring. In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design (IS-BMSD)*, pages 171–177. SCITEPRESS, 2015.
- [EBL16] Philipp Ebel, Ulrich Bretschneider, and Jan Marco Leimeister. Leveraging virtual business model innovation: a framework for designing business model development tools. *Information Systems Journal*, 26(5):519–550, 2016.

- [EHB11] Martin J.. Eppler, Friederieke Hoffmann, and Sabrina Bresciani. New business models through collaborative idea generation. *International Journal of Innovation Management*, 15(06):1323–1341, 2011.
- [ES16] David S. Evans and Richard Schmalensee. *Matchmakers: The new economics of multisided platforms*. Harvard Business Review Press, Boston, 2016.
- [EW17] Nesat Efendioglu and Robert Woitsch. A Modelling Method for Digital Service Design and Intellectual Property Management Towards Industry 4.0: CAXMan Case. In *Serviceology for Services*, volume 10371, pages 153–163. Springer, Cham, 2017.
- [FAM18] Christian Fleig, Dominik Augenstein, and Alexander Maedche. Tell Me What’s My Business - Development of a Business Model Mining Software. In *Information Systems in the Big Data Era*, volume 317, pages 105–113. Springer, Cham, 2018.
- [FB16] Masud Fazal-Baqaie. *Project-specific Software Engineering Methods: Composition, Enactment, and Quality Assurance*. Paderborn University, 2016.
- [FCP12] Abiola O. Fanimokun, Gary Castrogiovanni, and Mark F. Peterson. Developing High-Tech Ventures: Entrepreneurs, Advisors, and the Use of Non-Disclosure Agreements (NDAs). *Journal of Small Business & Entrepreneurship*, 25(1):103–119, 2012.
- [FK15] Hans-Georg Fill and Dimitris Karagiannis. On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. *Enterprise Modelling and Information Systems Architectures*, (8):4–25, 2015.
- [FLR⁺18] Gilbert Fridgen, Jannik Lockl, Sven Radszuwill, Alexander Rieger, Andre Schweizer, and Nils Urbach. A Solution in Search of a Problem: A Method for the Development of Blockchain Use. In *Proceedings of AMCIS 2018*. AIS, 2018.
- [FP10] Boris Fritscher and Yves Pigneur. Supporting Business Model Modelling: A Compromise between Creativity and Constraints. In *Task Models and Diagrams for User Interface Design*, volume 5963, pages 28–43. Springer, Cham, 2010.
- [FP14a] Boris Fritscher and Yves Pigneur. Computer Aided Business Model Design: Analysis of Key Features Adopted by Users. In *Proceedings of HICCS 2014*, pages 3929–3938. IEEE, 2014.
- [FP14b] Boris Fritscher and Yves Pigneur. Visualizing Business Model Evolution with the Business Model Canvas: Concept and Tool. In *Proceedings on the 16th Conference on Business Informatics (CBI)*, pages 151–158. IEEE, 2014.
- [FP16] Boris Fritscher and Yves Pigneur. Classifying Business Model Canvas Usage from Novice to Master: A Dynamic Perspective. In *Business Modeling and Software Design*, volume 257, pages 134–151. Springer, Cham, 2016.

- [Fra13] Ulrich Frank. Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In *Domain Engineering*, pages 133–157. Springer, Berlin, 2013.
- [FSDN15] Awdren de Lima Fontao, Rodrigo Pereira dos Santos, and Arilo Claudio Dias-Neto. Mobile Software Ecosystem (MSECO): A Systematic Mapping Study. In *Proceedings of the Annual Computer Software and Applications Conference (COMSAC)*, pages 653–658. IEEE, 2015.
- [FSMM17] Fabian Fagerholm, Alejandro Sanchez Guinea, Hanna Mäenpää, and Jürgen Münch. The RIGHT model for Continuous Experimentation. *J. Syst. Softw.*, 123:292–305, 2017.
- [FWCG13] Karolin Frankenberger, Tobias Weiblen, Michaela Csik, and Oliver Gassmann. The 4I-framework of business model innovation: a structured view on process phases and challenges. *International Journal of Product Development*, 18(3/4):249, 2013.
- [GA01] Jaap Gordijn and Hans Akkermans. Designing and evaluating e-business models. *IEEE Intelligent Systems*, 16(4):11–17, 2001.
- [GAYE21] Sebastian Gottschalk, Muhammad Suffyan Aziz, Enes Yigitbas, and Gregor Engels. Design Principles for a Crowd-Based Prototype Validation Platform. In *Software Business*, volume 434, pages 205–220. Springer, Cham, 2021.
- [GBH16] Martin Geissdoerfer, Nancy M.P. Bocken, and Erik Jan Hultink. Design thinking to enhance the sustainable business modelling process – A workshop based on a value mapping process. *Journal of Cleaner Production*, 135:1218–1232, 2016.
- [GBW⁺22] Sebastian Gottschalk, Rakshit Bhat, Nils Weidmann, Jonas Kirchhoff, and Gregor Engels. Low-Code Experimentation on Software Products. In *Proceedings of 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion)*. ACM, 2022.
- [Gen18] General Electric Inc. *GE Global Innovation Barometer 2018*. 2018.
- [GFC14] Oliver Gassmann, Karolin Frankenberger, and Michaela Csik. *The business model navigator: 55 models that will revolutionise your business*. Pearson, Harlow, 2014.
- [GFCL] Andrea Giessman, Alexander Fritz, Simon Caton, and Christine Legner. A Method For Simulating Cloud Business Models: A Case Study On Platform As A Service. In *Proceedings of the ECIS 2013*.
- [GFM19] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101–121, 2019.
- [GH13] Shirley Gregor and Alan R. Hevner. Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, 37(2):337–355, 2013.

- [GKE21] Sebastian Gottschalk, Jonas Kirchhoff, and Gregor Engels. Extending Business Model Development Tools with Consolidated Expert Knowledge. In *Business Modeling and Software Design*, volume 422, pages 3–21. Springer, Cham, 2021.
- [GKS20] Shirley Gregor, Leona Kruse, and Stefan Seidel. Research Perspectives: The Anatomy of a Design Principle. *Journal of the Association for Information Systems*, 21:1622–1652, 2020.
- [GM03] John Gross and Kenneth Mcinnis. *Kanban Made Simple: Demystifying and applying toyota’s legendary manufacturing process*. AMACOM, New York City, 2003.
- [Got21a] Sebastian Gottschalk. Decision Support Systems for Business Model Development. In *European Computer Science Summit*. Informatics Europe, 2021.
- [Got21b] Sebastian Gottschalk. Situation-specific Development of Business Models for Service Providers in Software Ecosystems. In *CAISE Doctorial Consortium*, volume 2906, pages 99–108. CEUR-WS, 2021.
- [GPYE22] Sebastian Gottschalk, Sarmad Parvez, Enes Yigitbas, and Gregor Engels. Designing Platforms for Crowd-based Software Prototype Validation: A Design Science Study. In *Product-Focused Software Process Improvement*, volume 13709. Springer, Cham, 2022.
- [GRE19a] Sebastian Gottschalk, Florian Rittmeier, and Gregor Engels. Business Models of Store-Oriented Software Ecosystems: A Variability Modeling Approach. In *Business Modeling and Software Design*, volume 356, pages 153–169. Springer, 2019.
- [GRE19b] Sebastian Gottschalk, Florian Rittmeier, and Gregor Engels. Intertwined Development of Business Model and Product Functions for Mobile Applications: A Twin Peak Feature Modeling Approach. In *Software Business*, pages 192–207. Springer, Cham, 2019.
- [GRE19c] Sebastian Gottschalk, Florian Rittmeier, and Gregor Engels. *Intertwined Development of Business Model and Product Functions for Mobile Applications: A Twin Peak Feature Modeling Approach: Technical Report*. 2019.
- [GRE20] Sebastian Gottschalk, Florian Rittmeier, and Gregor Engels. Hypothesis-driven Adaptation of Business Models based on Product Line Engineering. In *Proceedings of the International Conference on Business Informatics (CBI)*. IEEE, 2020.
- [Gri16] Marvin Grieger. *Model-Driven Software Modernization: Concept-Based Engineering of Situation-Specific Methods*. Paderborn University, 2016.
- [GSE17] Martin Geissdoerfer, Paulo Savaget, and Steve Evans. The Cambridge Business Model Innovation Process. *Procedia Manufacturing*, 8:262–269, 2017.

- [GSS⁺11] David Geiger, Stefan Seedorf, Thimo Schulze, Robert C. Nickerson, and Martin Schader. Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes. In *Proceedings of AMCIS 2011*. AIS, 2011.
- [GT18] Görkem Giray and Bedir Tekinerdogan. Situational Method Engineering for Constructing Internet of Things Development Methods. In *Business Modeling and Software Design*, volume 319, pages 221–239. Springer, Cham, 2018.
- [GWB10] Vânia Gonçalves, Nils Walravens, and Pieter Ballon. “How about an App Store?” Enablers and Constraints in Platform Strategies for Mobile Network Operators. In *Proceedings of the Ninth International Conference on Mobile Business and Ninth Global Mobility Roundtable (ICMB-GMR)*, pages 66–73. IEEE, 2010.
- [GY21] Sebastian Gottschalk and Enes Yigitbas. Von datenbasierter zu datengetriebener Geschäftsmodellentwicklung: Ein Überblick über Software-Tools und deren Datennutzung. In *WI-MAW-Rundbrief*. Gesellschaft für Informatik, 2021.
- [GYE20] Sebastian Gottschalk, Enes Yigitbas, and Gregor Engels. Model-Based Hypothesis Engineering for Supporting Adaptation to Uncertain Customer Needs. In *Business Modeling and Software Design*, volume 391, pages 276–286. Springer, Cham, 2020.
- [GYE22] Sebastian Gottschalk, Enes Yigitbas, and Gregor Engels. Model-driven Continuous Experimentation on Component-based Software Architectures. In *Proceedings of the 19th International Conference on Software Architectures*. IEEE, 2022.
- [GYNE21a] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Situation- and Domain-Specific Composition and Enactment of Business Model Development Methods. In *Product-Focused Software Process Improvement*, volume 13126, pages 103–118. Springer, Cham, 2021.
- [GYNE21b] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Situation-specific Business Model Development Methods for Mobile App Developers. In *Enterprise, Business-Process and Information Systems Modeling*, volume 421, pages 262–276. Springer, 2021.
- [GYNE21c] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. *Situation-specific Business Model Development Methods for Mobile App Developers: Technical Report*. 2021.
- [GYNE22a] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Continuous Situation-specific Development of Business Models: Knowledge Provision, Method Composition, Method Enactment. In *Journal of Software and Systems*. Springer, 2022.
- [GYNE22b] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Don’t Start from Scratch: A Modularized Architecture for Business Model Development Tools. In *Poster Presentation @ International Conference on*

- Software Business and South Tyrol Free Software Conference*. CEUR-WS, 2022.
- [GYNE22c] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Situational Business Model Developer: A Tool-support for Situation-specific Business Model Development. In *Proceedings of the Wirtschaftsinformatik*. AIS, Nuremberg, 2022.
- [GYNE22d] Sebastian Gottschalk, Enes Yigitbas, Alexander Nowosad, and Gregor Engels. Towards Software Support for Situation-specific Cross-organizational Design Thinking Processes. In *Proceedings of the 5th International Workshop on Software-intensive Business*. IEEE, 2022.
- [GYSE20a] Sebastian Gottschalk, Enes Yigitbas, Eugen Schmidt, and Gregor Engels. Model-Based Product Configuration in Augmented Reality Applications. In *Human-Centered Software Engineering*, volume 12481, pages 84–104. Springer, Cham, 2020.
- [GYSE20b] Sebastian Gottschalk, Enes Yigitbas, Eugen Schmidt, and Gregor Engels. ProConAR: A Tool Support for Model-Based AR Product Configuration. In *Human-Centered Software Engineering*, volume 12481, pages 207–215. Springer, Cham, 2020.
- [HBJdR17] Timber Haaker, Harry Bouwman, Wil Janssen, and Mark de Reuver. Business model stress testing: A practical approach to test the robustness of a business model. *Futures*, 89:14–25, 2017.
- [HBO94] Frank Hamsen, Sjaak Brinkkemper, and J. L. Han Oei. Situational method engineering for informational system project approaches. In *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle*, pages 169–194. 1994.
- [Hev07] Alan Hevner. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 2007.
- [HH12] Larissa Hammon and Hajo Hippner. Crowdsourcing. *Business & Information Systems Engineering*, 4(3):163–166, 2012.
- [HKB18] Basma Hamrouni, Ahmed Korichi, and Abdelhabib Bourouis. IDSS-BM. In *Proceedings of the 7th International Conference on Software Engineering and New Technologies*, pages 1–5, New York City, 2018. ACM.
- [HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, (28), 2004.
- [HO11] Adrian Holzer and Jan Ondrus. Mobile application market: A developer’s perspective. *Telematics and Informatics*, 28(1):22–31, 2011.
- [HSGP10] Brian Henderson-Sellers and Cesar Gonzalez-Perez. Granularity in Conceptual Modelling: Application to Metamodels. In *Conceptual Modeling – ER 2010*, volume 6412, pages 219–232. Springer, Berlin, 2010.

- [HSRÅR14] Brian Henderson-Sellers, Jolita Ralyté, Pär J. Ågerfalk, and Matti Rossi. *Situational Method Engineering*. Springer, Heidelberg, 2014.
- [HZFN16] Philipp Max Hartmann, Mohamed Zaki, Niels Feldmann, and Andy Neely. Capturing value from big data – a taxonomy of data-driven business models used by start-up firms. *International Journal of Operations & Production Management*, 36(10):1382–1406, 2016.
- [JKS17] Thomas John, Dennis Kundisch, and Daniel Szopinski. Visual Languages for Modeling Business Models: A Critical Review and Future Research Directions. In *Proceedings of ICIS 2017*. AIS, 2017.
- [Joh16] Thomas John. Supporting Business Model Idea Generation Through Machine-generated Ideas: A Design Theory. In *Proceedings of ICIS 2016*. AIS, 2016.
- [JPEK16] Bahar Jazayeri, Marie C. Platenius, Gregor Engels, and Dennis Kundisch. Features of IT Service Markets: A Systematic Literature Review. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, volume 9936, pages 301–316. Springer, 2016.
- [JZE⁺18] Bahar Jazayeri, Olaf Zimmermann, Gregor Engels, Jochen Küster, Dennis Kundisch, and Daniel Szopinski. Design Options of Store-Oriented Software Ecosystems: An Investigation of Business Decisions. In *Business Modeling and Software Design*, volume 319, pages 390–400. Springer, Cham, 2018.
- [KB10] Björn Kijl and Durk Boersma. Developing a business model engineering & experimentation tool - the quest for scalable ‘lollapalooza confluence patterns’. In *Proceedings of AMCIS 2010*. AIS, 2010.
- [KBS19] Ralf Knackstedt, Sebastian Bräuer, and Thorsten Schoormann. Tool Support for Designing Innovative Sustainable Business Models. In *The Art of Structuring*, pages 87–100. Springer, Cham, 2019.
- [KBWL12] Julian Krumeich, Thomas Burkhart, Dirk Werth, and Peter Loos. Towards a Component-based Description of Business Models: A State-of-the-Art Analysis. In *Proceedings of the AMCIS 2012*. 2012.
- [KCS08] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, page 453. ACM, 2008.
- [KGE22] Jonas Kirchhoff, Sebastian Gottschalk, and Gregor Engels. Detecting Data Incompatibilities in Process-Driven Decision Support Systems. In *Business Modeling and Software Design*. Springer, 2022.
- [Kit04] Barbara Kitchenham. *Procedures for Performing Systematic Reviews: Technical Report TR/SE- 0401*. 2004.
- [KK02] Dimitris Karagiannis and Harald Kühn. Metamodelling Platforms. In *E-Commerce and Web Technologies*, volume 2455, page 182. Springer, Berlin, 2002.

- [KV08] Bill Kuechler and Vijay Vaishnavi. On theory development in design science research: anatomy of a research project. *European Journal of Information Systems*, 17(5):489–504, 2008.
- [LCK⁺11] Julie Linsey, Emily Clauss, Tolga Kurtoglu, Jeremy Murphy, Kristin Lee Wood, and Arthur B. Markman. An Experimental Study of Group Idea Generation Techniques: Understanding the Roles of Idea Representation and Viewing Methods. *Journal of Mechanical Design*, 133(3), 2011.
- [Lei12] Jan Marco Leimeister. Crowdsourcing. *Controlling & Management*, 56(6):388–392, 2012.
- [LFBBM19] Florian Lüdeke-Freund, René Bohnsack, Henning Breuer, and Lorenzo Massa. Research on Sustainable Business Model Patterns: Status quo, Methodological Issues, and a Research Agenda. In *Sustainable Business Models*, pages 25–60. Springer, Cham, 2019.
- [LFCJ⁺18] Florian Lüdeke-Freund, Sarah Carroux, Alexandre Joyce, Lorenzo Massa, and Henning Breuer. The sustainable business model pattern taxonomy—45 patterns to support sustainability-oriented business model innovation. *Sustainable Production and Consumption*, 15:145–162, 2018.
- [LKH17] Sang M. Lee, Na Rang Kim, and Soon Goo Hong. Key success factors for mobile app platform activation. *Service Business*, 11(1):207–227, 2017.
- [LM16] Eveliina Lindgren and Jürgen Münch. Raising the odds of success: the current state of experimentation in product development. *Inf. Softw. Technol.*, 77:80–91, 2016.
- [LSS05] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. Studying Software Engineers: Data Collection Techniques for Software Field Studies. *Empirical Software Engineering*, 10(3):311–341, 2005.
- [MCF⁺95] Richard J. Mayer, John W. Crump, Ronald Fernandes, Arthur Keen, and Michael K. Painter. *Information Integration for Concurrent Engineering (IICE): Compendium on Method Reports*. 1995.
- [McG10] Rita Gunther McGrath. Business Models: A Discovery Driven Approach. *Long Range Planning*, (43):247–261, 2010.
- [MHHW18] Benedikt Morschheuser, Lobna Hassan, Karl Werder, and Juho Hamari. How to design gamification? A method for engineering gamified software. *Information and Software Technology*, 95:219–237, 2018.
- [MIN⁺12] Lucas Onno Meertens, Maria Eugenia Iacob, Bart Nieuwenhuis, J. M. van Sinderen, Henk Jonkers, and Dick Quartel. Mapping the Business Model Canvas to ArchiMate. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC)*. ACM, 2012.

- [MKM11] Roland M. Müller, Björn Kijl, and Josef K. J. Martens. A Comparison of Inter-Organizational Business Models of Mobile App Stores: There is more than Open vs. Closed. *Journal of theoretical and applied electronic commerce research*, 6(2):13–14, 2011.
- [MNJR16] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe. Toward Data-Driven Requirements Engineering. *IEEE Software*, 33(1):48–54, 2016.
- [MP05] Hilangwa Maimbo and Graham Pervan. Designing a Case Study Protocol for Application in IS Research. In *Proceedings of PACIS 2005*. 2005.
- [MTA17] Lorenzo Massa, Christopher L. Tucci, and Allan Afuah. A Critical Assessment of Business Model Research. *Academy of Management Annals*, 11(1):73–104, 2017.
- [MVG⁺14] Andreas Menychtas, Jürgen Vogel, Andrea Giessmann, Anna Gatzidou, Sergio Garcia Gomez, Vrettos Moulos, Frederic Junker, Mathias Müller, Dimosthenis Kyriazis, Katarina Stanoevska-Slabeva, and Theodora Varvarigou. 4CaaS marketplace: An advanced business environment for trading cloud services. *Future Generation Computer Systems*, 41:104–120, 2014.
- [MWA19] Jorge Melegati, Xiaofeng Wang, and Pekka Abrahamsson. Hypotheses Engineering: First Essential Steps of Experiment-Driven Software Development. In *RCoSE/DDrEE*, pages 16–19. IEEE, 2019.
- [MYW⁺15] Ke Mao, Ye Yang, Qing Wang, Yue Jia, and Mark Harman. Developer Recommendation for Crowdsourced Software Development Tasks. In *Proceedings of the IEEE Symposium on Service-Oriented System Engineering*, pages 347–356. IEEE, 2015.
- [Nam17] Satish Nambisan. Digital Entrepreneurship: Toward a Digital Technology Perspective of Entrepreneurship. *Entrepreneurship Theory and Practice*, 41(6):1029–1055, 2017.
- [NVM13] Robert C. Nickerson, Upkar Varshney, and Jan Muntermann. A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22(3):336–359, 2013.
- [OB14] Helena Holmström Olsson and Jan Bosch. The HYPEX Model: From Opinions to Data-Driven Software Development. In *Continuous Software Engineering*, volume 14, pages 155–164. Springer, 2014.
- [OB15] Helena Holmström Olsson and Jan Bosch. Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation. In *Software Business*, volume 210, pages 154–166. Springer, 2015.
- [Obj08] Object Management Group. *Software & Systems Process Engineering Meta-model (SPEM)*. 2008.
- [Obj10] Object Management Group. *Business Process Model And Notation (BPMN)*. 2010.

- [Obj16] Object Management Group. *Meta Object Facility (MOF)*. 2016.
- [Obj17] Object Management Group. *Unified Modeling Language (UML)*. 2017.
- [OP10] Alexander Osterwalder and Yves Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons, Hoboken, 2010.
- [OP13a] Alexander Osterwalder and Yves Pigneur. *Business model generation: A handbook for visionaries, game changers, and challengers*. Wiley&Sons, New York, 2013.
- [OP13b] Alexander Osterwalder and Yves Pigneur. Designing Business Models and Similar Strategic Objects: The Contribution of IS. *Journal of the Association for Information Systems*, 14(5):237–244, 2013.
- [OPB⁺14] Alexander Osterwalder, Yves Pigneur, Greg Bernarda, Alan Smith, and Patricia Papadakos. *Value proposition design: How to create products and services customers want. Get started with*. Strategyzer series. John Wiley & Sons, Hoboken, 2014.
- [OPT05] Alexander Osterwalder, Yves Pigneur, and Christopher L. Tucci. Clarifying Business Models: Origins, Present, and Future of the Concept. *Communications of the Association for Information Systems*, 16, 2005.
- [Ost04] Alexander Osterwalder. *The Business Model Ontology - A Proposition in a Design Science Approach*. Lausanne University, 2004.
- [Par72] David Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.
- [PF14] Yves Pigneur and Boris Fritscher. Business Model Design: An Evaluation of Paper-based and Computer-Aided Canvases. In *Proceedings of the Fourth International Symposium on Business Modeling and Software Design (IS-BMSD)*, pages 236–244. SCITEPRESS, 2014.
- [PHS08] Mikko Pynnonen, Jukka Hallikas, and Petri Savolainen. Mapping business: value stream-based analysis of business models and resources in Information and Communications Technology service business. *International Journal of Business and Systems Research*, 2(3):305, 2008.
- [PvC16] Geoffrey Parker, Marshall van Alstyne, and Sangeet Paul Choudary. *Platform revolution: Platform Revolution: How Networked Markets Are Transforming the Economy - and How to Make Them Work for You*. W.W. Norton & Company, New York, 2016.
- [Ral04] Jolita Ralyté. Towards situational methods for information systems development: engineering reusable method chunks. In *Proceedings of the 13th international conference on information systems development. Advances in theory, practice and education*. 2004.

- [RBLL⁺22] Susanne Robra-Bissantz, Christoph Lattemann, Ralf Laue, Raphaela Leonhard-Pfleger, Luisa Wagner, Oliver Gerundt, Ricarda Schlimbach, Sabine Baumann, Christian Vorbohle, Sebastian Gottschalk, Dennis Kundisch, Gregor Engels, Nancy Wunderlich, Volker Nissen, Lisa Lohrenz, and Simon Michalke. Methoden zum Design digitaler Plattformen, Geschäftsmodelle und Service-Ökosysteme. *HMD Praxis der Wirtschaftsinformatik*, 2022.
- [RH09] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.
- [RHNH⁺18] Kira Rambow-Hoeschele, Anna Nagl, David K. Harrison, Bruce M. Wood, Karlheinz Bozem, Kevin Braun, and Peter Hoch. Creation of a Digital Business Model Builder. In *Proceedings of the IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–7. IEEE, 2018.
- [RHRH⁺19] Kira Rambow-Hoeschele, Nick Giani Rambow, Matthias Michael Hampel, David Keith Harrison, and Bruce MacLeod Wood. Creating a Digital Twin: Simulation of a Business Model Design Tool. *Advances in Science, Technology and Engineering Systems Journal*, 4(6):53–60, 2019.
- [RHTK17] Gerrit Remane, Andre Hanelt, Jan Tesch, and Lutz Kolbe. The Business Model Database - A Tool for Systematic Business Model Innovation. *International Journal of Innovation Management*, 21(01), 2017.
- [Rie14] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, USA, 2014.
- [RM16] Colin Robson and Kieran McCartan. *Real world research: A resource for users of social research methods in applied settings*. Wiley, Chichester, 4th edition edition, 2016.
- [RR16] Paolo Roma and Daniele Ragaglia. Revenue models, in-app purchase, and the app performance: Evidence from Apple's App Store and Google Play. *Electronic Commerce Research and Applications*, 17:173–190, 2016.
- [RRE⁺19] Sarah Rübel, Adrian Rebmann, Andreas Emrich, Sabine Klein, and Peter Loos. Improving Business Model Configuration through a Question-based Approach. In *Proceedings of the International Conference on Wirtschaftsinformatik (WI)*. AIS, 2019.
- [RVS15] Maria Camila Romero, Jorge Villalobos, and Mario Sanchez. Simulating the business model canvas using system dynamics. In *Proceedings of the 10th Computing Colombian Conference (10CCC)*, pages 527–534. IEEE, 2015.
- [Sau11] Stefan Sauer. *Systematic Development of Model-based Software Engineering Methods*. Paderborn University, 2011.
- [SBK18a] Thorsten Schoormann, Dennis Behrens, and Ralf Knackstedt. Design Principles for Leveraging Sustainability in Business Modelling Tools. In *Proceedings of ECIS 2018*. AIS, 2018.

- [SBK18b] Thorsten Schoormann, Dennis Behrens, and Ralf Knackstedt. The Noblest Way to Learn Wisdom is by Reflection: Designing Software Tools for Reflecting Sustainability in Business Models. In *Proceedings of the ICIS 2018*. 2018.
- [Sch14] Markus Schief. Software Business Model Tool. In *Business Models in the Software Industry*, pages 169–182. Springer, Wiesbaden, 2014.
- [SELK20] Norman Schaffer, Martin Engert, Girts Leontjevs, and Helmut Krcmar. A Tool to Model and Simulate Dynamic Business Models. In *Proceedings of BLED eConference 2020*. 2020.
- [SEP⁺19] Benedikt Simmert, Philipp Alexander Ebel, Christoph Peters, Eva Alice Christiane Bittner, and Jan Marco Leimeister. Conquering the Challenge of Continuous Business Model Improvement. *Business & Information Systems Engineering*, 61(4):451–468, 2019.
- [SG89] Susan Leigh Star and James R. Griesemer. Institutional Ecology, ‘Translations’ and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3):387–420, 1989.
- [SHB⁺20] Thorsten Schoormann, Simon Hagen, Jonas Brinker, Sebastian Wildau, Oliver Thomas, and Ralf Knackstedt. Towards Aligning Business Models with Business Processes: A Tool-based Approach. In *Proceedings of Modellierung 2020*. 2020.
- [SL20] Johannes S. Schwarz and Christine Legner. Business model tools at the boundary: exploring communities of practice and knowledge boundaries in business model innovation. *Electronic Markets*, 30(3):421–445, 2020.
- [SLF17] Tina Saebi, Lasse Lien, and Nicolai J. Foss. What Drives Business Model Adaptation? The Impact of Opportunities, Threats and Strategic Orientation. *Long Range Planning*, 50(5):567–581, 2017.
- [SP09] Juneseuk Shin and Yongtae Park. On the creation and evaluation of e-business model variants: The case of auction. *Industrial Marketing Management*, 38(3):324–337, 2009.
- [SSJ⁺19] Daniel Szopinski, Thorsten Schoormann, Thomas John, Ralf Knackstedt, and Dennis Kundisch. Software tools for business model innovation: current state and future challenges. *Electronic Markets*, 60(11):2794, 2019.
- [SSSV19] Krista Sorri, Marko Seppänen, Kaisa Still, and Katri Valkokari. Business Model Innovation with Platform Canvas. *Journal of Business Models*, 7(2), 2019.
- [Sti14] Volker Stiehl. *Process-Driven Applications with BPMN*. Springer International Publishing, Cham, 2014.
- [STRV10] Marc Sosna, Rosa Nelly Trevinyo-Rodríguez, and S. Ramakrishna Velamuri. Business Model Innovation through Trial-and-Error Learning. *Long Range Planning*, 43(2-3):383–407, 2010.

- [SWS20] Norman Schaffer, Jörg Weking, and Olivia Stähler. Requirements and Design Principles for Business Model Tools. In *Proceedings of the AMCIS 2020*. 2020.
- [SYT09] Reza Samavi, Eric Yu, and Thodoros Topaloglou. Strategic reasoning about business models: a conceptual modeling approach. *Inf Syst E-Bus Manage*, 7(2):171–198, 2009.
- [Szo19a] Daniel Szopinski. Can Stimuli Improve Business Model Idea Generation? In *Proceedings of 12th Conference on Creativity and Cognition*, pages 547–555, New York City, 2019. ACM.
- [Szo19b] Daniel Szopinski. Jumping, dumping, and pumping: Three mental principles for idea Jumping, dumping, and pumping: Three mental principles for idea generation to activate software-based tools in business model generation to activate software-based tools in business model innovation. In *Proceedings of the BLED eConference 2019*. 2019.
- [TA17] Karl Täuscher and Nizar Abdelkafi. Visual tools for business model innovation: Recommendations from a cognitive perspective. *Creativity and Innovation Management*, 26(2):160–174, 2017.
- [Tee10] David J. Teece. Business Models, Business Strategy and Innovation. *Long Range Planning*, 43(2-3):172–194, 2010.
- [TK09] Juha-Pekka Tolvanen and Steven Kelly. MetaEdit+. In Shail Arora, Gary Leavens, Bernd Bruegge, Yvonne Coady, and Simon Peyton-Jones, editors, *Proceeding of the 24th ACM SIGPLAN conference companion*, page 819. ACM, 2009.
- [TM11] Katja Thoring and Roland M. Müller. Understanding Design Thinking: A Process Model Based on Method Engineering. In *Proceedings of the International Conference on Engineering and Product Design Education*. E&PDE, 2011.
- [TSLE17] Nicola Terrenghi, Johannes Schwarz, Christine Legner, and Uli Eisert. Business Model Management: Current Practices, Required Activities and IT Support. In *Proceedings of the Conference on Wirtschaftsinformatik (WI)*, pages 972–986. AIS, 2017.
- [TTP11] Virpi Kristiina Tuunainen, Tuure Tuunainen, and Jouni Piispanen. Mobile Service Platforms: Comparing Nokia OVI and Apple App Store with the IISIn Model. In *Proceedings of the International Conference on Mobile Business (ICMB)*, pages 74–83. IEEE, 2011.
- [van01] Alex van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In *International Symposium on Requirements Engineering*, pages 249–262. IEEE, 2001.
- [VCB⁺14] Daniel Veit, Eric Clemons, Alexander Benlian, Peter Buxmann, Thomas Hess, Dennis Kundisch, Jan Marco Leimeister, Peter Loos, and Martin Spann. Business Models: An Information Systems Research Agenda. *Business & Information Systems Engineering*, 6(1):45–53, 2014.

- [VG21] Christian Vorbohle and Sebastian Gottschalk. Towards Visualizing and Simulating Business Models in Dynamic Platform Ecosystems. In *Proceedings of ECIS 2021*. AIS, 2021.
- [VGK⁺22] Christian Vorbohle, Sebastian Gottschalk, Dennis Kundisch, Gregor Engels, and Nancy Wunderlich. A Procedure Model for Enhancing Ideation in the Collaborative Development of Business Ecosystems. In *Digital Business Ecosystems Workshop @ Wirtschaftsinformatik 2022*. AIS, 2022.
- [Vog17] Peter Vogel. From Venture Idea to Venture Opportunity. *Entrepreneurship Theory and Practice*, 41(6):943–971, 2017.
- [Völ13] Markus Völter. *DSL engineering: Designing, implementing and using domain-specific languages*. CreateSpace Independent Publishing Platform, Lexington, KY, 2013.
- [VOPN13] Matthias Voigt, Kevin Ortbach, Ralf Plattfaut, and Björn Niehaves. Developing Creative Business Models – The OctoProz Tool. In *Design Science at the Intersection of Physical and Virtual Design*, volume 7939, pages 456–462. Springer, Heidelberg, 2013.
- [VPO⁺13] Matthias Voigt, Ralf Plattfaut, Kevin Ortbach, Andrea Malsbender, and Björn Niehaves. Evaluating Business Modeling Tools from a Creativity Support System Perspective - Results from a Focus Group in the Software Development Industry. In *Proceedings of the PACIS 2013 2013*. 2013.
- [vvR19] Marlies van Steenberghe, Jeroen van Grondelle, and Lars Rieser. A Situational Approach to Data-Driven Service Innovation. In *Enterprise, Business-Process and Information Systems Modeling*, volume 352, pages 156–168. Springer, Cham, 2019.
- [WF20] Michael Wieland and Hans-Georg Fill. A Domain-Specific Modeling Method for Supporting the Generation of Business Plans. In *Proceedings of Modellierung 2020*, pages 45–60. GI, 2020.
- [WW11] Nico Weiner and Anette Weisbecker. A Business Model Framework for the Design and Evaluation of Business Models in the Internet of Services. In *Proceedings of the Annual SRII Global Conference*, pages 21–33. IEEE, 2011.
- [XHB14] Anbang Xu, Shih-Wen Huang, and Brian Bailey. Voyant: generating structured feedback on visual designs using a crowd of non-experts. In *ACM conference on Computer supported cooperative work & social computing*, pages 1433–1444. ACM, 2014.
- [Yin09] Robert K. Yin. *Case study research: Design and methods*, volume 5 of *Applied social research methods series*. Sage, Los Angeles, 2009.
- [ZSDM14] Marin Zec, Alexander W. Schneider, Peter Dürr, and Florian Matthes. Improving Computer-Support for Collaborative Business Model Design and Exploration. In *Proceedings of the Fourth International Symposium on Business Modeling and Software Design (IS-BMSD)*, pages 29–37. SCITEPRESS, 2014.

Abbreviations

AR	Augmented Reality
BA	Business Approaches
BM	Business Model
BMC	Business Model Canvas
BM	Business Model Development
BMI	Business Model Innovation
BMO	Business Model Ontology
BMDM	Business Model Development Method
BMDSS	Business Model Decision Support System
BMDT	Business Model Development Tool
BMML	Business Model Modeling Languages
BPMN	Business Process Model and Notation
BT	Business Tools
CBPV	Crowd-based Prototype Validation
CLI	Command Line Interface
CRUD	Create, Read, Update, Delete
DAR	Development Assistance Requirement
DB	Data Base

DF	Design Feature
DP	Design Principle
DR	Design Requirement
DMR	Development Method Requirement
DSL	Domain-specific Language
DSR	Design Science Research
DSS	Decision Support System
DTD	Document Type Definition
ERP	Enterprise Resource Planning
FM	Feature Model
FSC	Feature Set Canvas
GPL	General Purpose Language
HR	High-level Requirement
HTML	Hyper Text Markup Language
HypoMoMap	Hypotheses Modeling and Mapping
JS	JavaScript
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KPR	Knowledge Provision Requirements
ME	Method Engineering
MO	Model Engineering
MOF	Meta Object Facility
NPM	Node Package Manager
OMG	Object Management Group

RQ	Research Question
SA	Situational Approaches
SBMD	Situational Business Model Developer
SDK	Software Development Kit
SE	Software Ecosystem
SLR	Systematic Literature Review
SME	Situational Method Engineering
SPEM	Software and Systems Process Engineering
SPL	Software Product Line
SQL	Structured Query Language
SR	Solution Requirement
TD	Taxonomy Development
UML	Unified Modeling Language
URL	Uniform Resource Locator
VPC	Value Proposition Canvas
VR	Virtual Reality
WYSIWYG	What-You-See-Is-What-You-Get
XML	Extensible Markup Language

Appendix A

SLR on Business Model Decision Support Systems

The development of business models is a complex task that can be supported using decision support systems (DSSs). These DSSs, which are partly referred to as BMDTs, can support the different phases of BMD. Inside this appendix, we provide an SLR on those DSSs that are used to derive our HRs (see Section 1.3), provide parts of our related work (see Section 3.3) and modularize the development support of development steps (see Section 7.2). In the following, we show our research approach used and the derived results.

Conducted Research Approach

To collect the developed approaches for DSSs for BMD from the literature, we use an SLR as proposed by Kitchenham [Kit04]. Here, the SLR is a procedure to provide a systematic overview of a specific theme or topic in research. The results can be used to identify gaps in the current research to suggest further investigations [Kit04]. The method consists of the three steps of planning the review, conducting the review, and reporting the review.

In *Planning the Reviews*, an SLR needs to be motivated together with the explicit formulation of the RQ the study aims to answer. Out of the RQ, the search string and related inclusion and exclusion criteria are determined and stated within a research protocol. The motivation for the SLR is justified by our DSR study, in which we create a situation-specific BMD approach. Here, we want to use the results to foster the requirements from the relevance cycle and the grounding from the rigor cycle [Hev07]. Together with a literature review on BMD, we identify the corresponding research gap and derive our HRs.

To identify the corresponding BMDSSs, we have defined the following research question:

- **RQ:** What decision support systems for business model development exist in the literature and how do they provide decision support?

To answer this question and by testing different search terms with the terminology in BMD, we have defined the following search string:

- **Search String:** business model and (decision support system or tool)

Moreover, we provide the following inclusion and exclusion criteria for the publications:

- **Inclusion:**
 - The publication is written in English.
 - The publication is a peer-reviewed workshop paper, a conference paper, a journal article or a book chapter.
 - The publication is written between January 2010 and August 2020 because the interest in business models significantly increased with the publication of the Business Model Canvas [OP10].
- **Exclusion:**
 - The publication provides no decision support for business model development.
 - The publication does not relate to the topic of business models (e.g., business processes).
 - The publication just applied existing techniques for decision support to business models (e.g., system dynamics, agent-based modeling).

In *Conducting the Review*, the review needs to be conducted by considering the search process, the source selection, the quality assessment, the data extraction, and the data synthesis. In the search process, we apply the search string to the five libraries of SpringerLink¹, AIS eLibrary², ACM Digital Library³, IEEE Xplore⁴, and ScienceDirect⁵ on September 17th, 2020, and follow the stages A-F from Figure A.1. At the beginning (A), we apply the search string on all meta-data (i.e., title, abstract, keywords) of the publication to focus only on papers that either provide a DSS or software tool for business modeling. We analyze

¹Website of SpringerLink: <https://link.springer.com/>

²Website of AIS eLibrary: <https://aisel.aisnet.org/>

³Website of ACM Digital Library: <https://dl.acm.org/>

⁴Website of IEEE Xplore: <https://ieeexplore.ieee.org/>

⁵Website of ScienceDirect: <https://www.sciencedirect.com/>

the meta-data of these papers (B) and the full texts (C) to identify interesting approaches by considering the inclusion and exclusion criteria. Then we apply backward search (D) and forward search (E) using Google Scholar⁶ to cover all iterations of the design cycles of these publications and discover further publications. In the end, we group the 54 identified publications into 31 different approaches.

Sources	Stages											
SpringerLink	(A) Keyword Search	371	(B) Meta-Data Analysis	37	(C) Fulltext Analysis	13	(D) Backward Search		(E) Forward Search		(F) Grouping	13
AIS eLibrary		51		22		13						+ 13
ACM DL		18		4		2						+ 2
IEEE XPlore		108		23		4						+3
Science Direct		338		10		1						+ 1
Snowballing								12				+ 12
Citation Check										10		+ 10
Grouping												54 in 31 Approaches

Fig. A.1 Extracted Number of Publications per Stage of the SLR

In *Reporting the Review*, we categorize those approaches around the three categories of modeling & configuration, analysis & simulation, and evolution & validation. We describe them shortly and use parts of them to derive our HRs (see Section 1.3), provide our related work (see Section 3.3), and provide our development support (see Section 7.2).

Derived Results

In this section, we present the derived approaches out of our SLR. We divide them into the categories of model & configuration, analysis & simulation, and evolution & validation.

The category of *Modeling & Configuration* refers to tools that provide decision support on the first creation of a business model. Common examples of decision support in these tools are the usage of business model patterns, a wizard for the configuration of a business model, or creativity enhancement through guiding questions.

- **[moby:designer]bm**: The approach [WW11] focuses on the development of different business model alternatives together with their evaluation. For that, they use existing

⁶Website of Google Scholar: <https://scholar.google.com/>

BMMLs to develop their own BMO to represent different business models together with a graphical tool to guide the configuration process.

- **Computer-Aided Business Model Design:** Under the term of computer-aided BMD, the authors propose different concepts [FP10, FP14b, FP14a, FP16, PF14] to support BMD. Here, they provide the idea of visualizing the evolution of a business model [FP14b] and provide a list of key features for computer-aided development tools [FP14a]. Moreover, they compare computer-aided and paper-based business model design [PF14] and state that different user maturity levels also have an impact on the BMD process [FP16].
- **OctoProz:** The approach [VOPN13, VPO⁺13] describes itself as a creativity support system for creating process-oriented business models. For that, they provide collaboration between different stakeholders, guidance in the business model creation, simple financial calculations, and ratings [VOPN13]. Moreover, they provide an evaluation with expert focus groups [VPO⁺13].
- **Collaborative Business Model Design and Exploration:** The authors provide a concept [ZSDM14] for the collaborative business model design. They use a business model configuration where the business model components are broken into smaller, ideally mutually exclusive, and collectively exhaustive business model elements.
- **Business Model Management System:** The solution [BWRP15] provides the requirements for an information system that can be used to manage business models among the whole company. They divide between different short-term and long-term requirements and use a design study to develop the first prototype.
- **Business Model Miner:** The approach [AF17, AFD18, FAM18] is the concept of a tool to automatically derive the own company's business model from an existing ERP system and corresponding business models of other companies from crawled web pages. They have worked on the mining of the value proposition [AFD18] and the business model [AF17]. Moreover, they have integrated their approach into the ERP of SAP [FAM18].
- **Active Business Model Development Tool:** The approach [Szo19b, Szo19a] provides creativity support by adding external stimuli to the BMD. For that, they applied the psychological model of searching for ideas in associated memory to the business modeling domain [Szo19b] and used what-if questions to support the ideation process of new business models [Szo19a].

- **Business Modeling Infrastructure:** The approach [EW17] develops a tool for configuring value propositions and business models based on the combination of different modeling languages and the identification of KPIs for successful business models.
- **Question-based Business Model Configurator:** The approach [RRE⁺19] supports the comparability of business models by using a questionnaire-based initialization process of new business models based on a predefined taxonomy of business model elements.
- **BusinessMakeOver:** The approach [dRAH⁺16] is an online platform that provides small and medium-sized enterprises decision support in finding proper business model tools for different phases of BMD. These tools were primarily visualizations, together with an explanation of how to use them.
- **Smart Business Modeler:** The approach [LFBBM19] is a tool that supports the development of sustainable business models based on different business model pattern packs. These different packs can be chosen out of a library during the configuration process of the business model.

The category of *Analysis & Simulation* refers to tools that provide different analyses and simulation techniques on a created business model. Common examples of decision support in these tools are the comparison of different business models, the financial assessment of different business model configurations, or the planning of best- and worst-case scenarios.

- **e3-Value Modeler:** The approach [GA01, AG03] can be used to define a value network of multiple stakeholders based on the e3-value modeling language. Out of this modeling, the outcome of different scenarios can be calculated.
- **Business Model Engineering and Experimentation Tool:** The approach [KB10] provides a performance analysis of the business models based on different scenarios. For that, they develop a seven-step process from obtaining key variables to the analysis of the strengths and weaknesses of a business model.
- **Business Model Stress Tester:** The approach [BHH⁺18, HBJdR17, BdRS⁺12] combines the modeling of the business model and uncertainties to analyze the robustness of the company. For that, they identify the stress factors of the business model and visualize them with the support of a heat map.
- **Software Business Model Tool:** The approach [Sch14] combines a step-by-step configuration of the company's business model with the analysis of the market, the

financial situation, and the firm performance. For that, we develop an architecture that derives those data out of different repositories and uses them for the process steps of configuration, comparison, and analysis.

- **Business Model Assistance System:** The approach [DEWL13, KBWL12, DWL15] provides recommendations for business model adaptations based on an existing database of business models from competitors. Moreover, it introduces a connection to an ERP system to analyze the current information and material flows of the company.
- **Business Model Developer:** The approach [BM17, BM14] uses cluster analysis to compare the own business with other companies in the same market based on the customization of the underlying business modeling structure together with a guided configuration process.
- **Business Model Simulator:** The approach [GFCL] provides a process model for the simulation of business models based on a case study on platforms as a service. The model consists of the three phases of method setup and data acquisition, business model design, and business model simulation.
- **Business Model Decision Support System:** The approach [DHOB13] supports the market analysis of different business model configurations by modeling critical design issues and success factors. This information can also be used to estimate the consequences of the different configurations for the company.
- **Business Model Analyzer:** The approach [AFM18, AM17, AF18] focuses on transforming a current business model into a new target one. For this, they model semantic relationships between the models and quantify the impact of changes to a set of alternative business models.
- **Digital Business Model Builder:** The approach [RHRH⁺19, RHNH⁺18] provides a digital twin of a business model that can be used to simulate the changes of KPIs based on changing market conditions. For that, they divide the business ideas, the qualitative description of a business model, and the business case.

The category of *Evolution & Validation* refers to tools that provide traceability in the evolution steps of business modeling and validate the different business model decisions with external stakeholders. Common examples of decision support in these tools are visualization of the business model evolution steps, deriving new steps based on the decision history, and validating business models through crowd worker platforms.

- **Business Model Idea Generators:** The approach [Joh16] refers to a new type of information system where business model ideas are generated through a combination of existing computer knowledge and new human ideas. For that, new ideas are iteratively generated by the systems and evaluated by crowd users.
- **IoT Business Model Change:** The approach [AHdR18b, AdR18, AHdR18a] is a concept for the systematic exploration of different business models. For that, it develops an iterative approach for creating and testing business models by using business model patterns.
- **Virtual Business Model Innovation:** The approach [EBL16] provides design patterns for a unified framework for BMDTs. They focus on the whole process of analyzing, designing, implementing, and managing business models that can be supported by shared material about business models and a community of users.
- **Framework for Analysis of Business Model Management:** The approach [TSLE17] provides a framework for business model management consisting of the phases of analysis, design, implementation, and control. Here, every phase should be provided with corresponding IT support.
- **Intelligent Decision Support System for Business Models:** The approach [HKB18] provides the concept of DSSs that reasoned its decisions to the company's management, which provides trust in computer-based systems. With this approach, they represent the robustness of the business model based on a heat map.
- **Crowd-Based Business Model Validation System:** The approach [DLE17] reduces uncertainties of the business model by offering a crowd-based validation with potential customers. Using an iterative process, they submit business model ideas to a crowd platform and evaluate the corresponding feedback.
- **Hybrid Intelligence Decision Support System:** The approach [DLL18, DLEL19] combines the knowledge of computers and humans to improve the validation of new business model ideas. For that, the feedback of the crowd platform is systematically recombined to generate new ideas.
- **Green Business Modeling Editor:** The approach [SBK18a, SBK18b, KBS19] is a tool for sustainable business model development by allowing the collaboration of different users. For that, it extends the BMC with custom building blocks and tracks the reasons for changes in the business model.

- **BM-BPM Alignment Tool:** The approach [SHB⁺20] aligns the business models with the business processes to improve the decision-making process. This alignment can be done on the whole BMC, single components, or single elements.
- **BMDL Feature Modeler⁷:** The approach [GRE19b, GRE20] provides a meta-modeling structure of business elements based on feature models. Here a set of those features can be selected and adapted over time by conducting experiments with potential customers.
- **Modesk:** The approach [SELK20, SWS20] models and simulates dynamic business models. For that, they add interdependencies between business components and use system dynamics for simulating the outcome.

⁷The approach is a preliminary version of our tool after the first design cycle.

Appendix B

Installation of the Situational Business Model Developer

The *Situational Business Model Developer (SBMD)*, as shown in Figure B.1, is our situated implementation within our design science study. It supports all three stages of our concept for situation-specific BMD. For that, the tool can be used online¹ or the source code can be downloaded and installed locally².

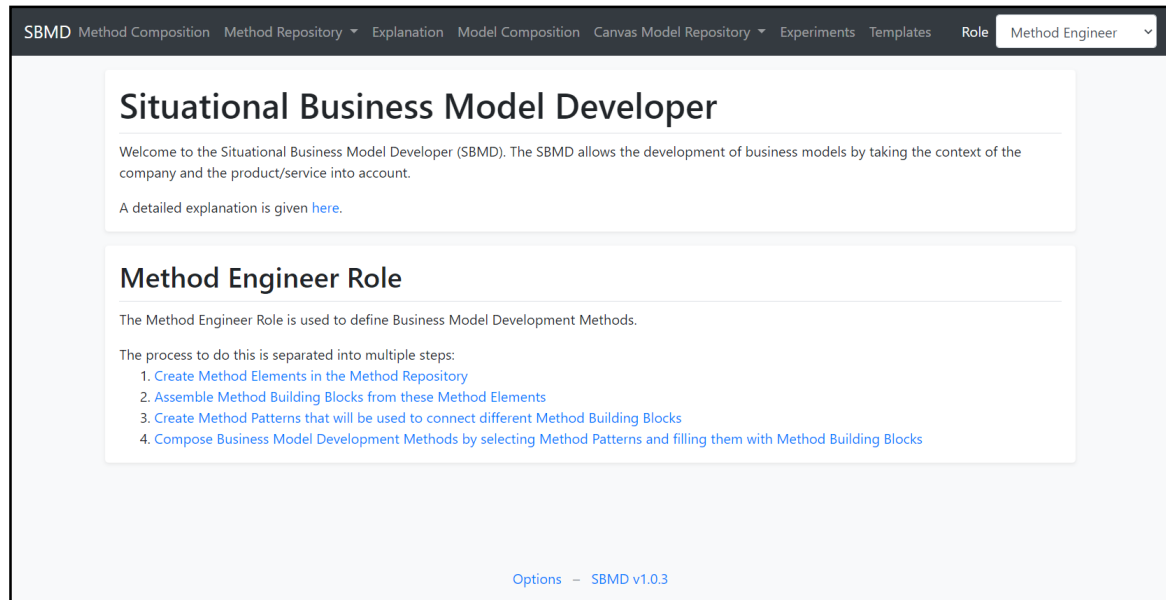


Fig. B.1 Tool Overview for the Method Engineer

¹Online Version of the SBMD: <http://sebastiangtts.github.io/situational-business-model-developer/>

²Source Code of the SBMD: <https://github.com/sebastiangtts/situational-business-model-developer>

The SBMD is based on NodeJS³ and the Angular CLI⁴ that need to be downloaded and installed before the tool itself. Here, NodeJS is a runtime environment to execute JS code that comes with the Node Package Manager (NPM) to include external JS code packages in the own software. AngularCLI is a command line interface (CLI) to develop and maintain Angular applications. Moreover, optionally CouchDB⁵ can be installed as persistent database storage and GitHub Desktop⁶ to clone the repository. Based on those tools, the SBMD can be used with the following guidelines:

1. Get the current version of the SBMD from Github
 - (a) Clone the GitHub repository with a GitHub Desktop
 - (b) Download the repository directly on GitHub
2. Install the needed NPM packages by running *npm install* in the repository
3. Configure the used database
 - (a) Internal Database: As default, the application uses the PouchDB within the web browser. The name of the default database is *bmdl-feature-modeler*. In the *src/app/pouchdb.service.ts*, you can change the variable *databaseName*.
 - (b) External Database: Optional, a CouchDB can be used as persistent storage. For that, the *databaseName* in the *src/app/pouchdb.service.ts* needs to be changed to *http://localhost:4200/database*. Moreover, the URL to the CouchDB needs to be added as *target* in the *proxy.conf.json*.
4. Run the SBMD application
 - (a) Internal Database: Run the SBMD with *ng serve*
 - (b) External Database: Run the SBMD with *npm start*
5. Use the SBMD to develop your own business models by open *http://localhost:4200* in your web browser

³Website of NodeJS: <https://nodejs.org/>

⁴Website of AngularCLI: <https://angular.io/cli>

⁵Website of CouchDB: <https://couchdb.apache.org/>

⁶Website of GitHub Desktop: <https://desktop.github.com/>