Domain-Specific Reasoning for Method Engineering based on Toulmin's Argumentation Theory

Sebastian Bittmann

Information Management and Information Systems
University Osnabrueck, Germany
E-mail:sebastian.bittmann@uni-osnabrueck.de*
*Corresponding author

Balbir Barn, Tony Clark

Middlesex University, London, UK
E-mail: b.barn@mdx.ac.uk
E-mail: t.n.clark@mdx.ac.uk

Abstract: Methods describe and embody a broad range of relevant knowledge of enterprises. Usually they have to account for requirements stated by a multitude of various stakeholders. These are typically those that are in charge of business related actions and those that are in charge to support such actions with an IT-Infrastructure. The statement of requirements as well as the validation of methods and in particular process models with respect to those requirements relies drastically on natural language. Natural language seems to be a substantial component to explain and to give an understanding about process models or certain aspects of it. This fact requires closing the gap between the natural language and the respective modelling language. This paper proposes argumentative method engineering for purposefully depicting design decisions and convictions for method engineering through arguments. The approach is derived from Toulmin's Argumentation Model and explicates the process of negotiating with various stakeholders. So, a model, depicting a method, specified by means of argumentative method engineering, not just includes the claims about a certain domain, it further justifies these claims by referring to already established knowledge. While it can't be ensured that certain requirements are considered in future project, if the reasons for design decisions of method engineering are transcribed in natural language text, but the semi-formalising of arguments regarding these methods allows such an assurance. So the argumentative approach enables the sophisticated management and reuse of knowledge during the development and extension of methods. The approach is evaluated using a case study, in which a software development method was outsourced to contractors.

Keywords: Method Engineering; Conceptual Modelling; Argumentation and Discourse; Toulmin

Reference to this paper should be made as follows: xxxx (xxxx) 'xxxx', *xxxx*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Sebastian Bittmann is a research assistant at the institute of Information and Business Management, which belongs to the University Osnabrueck, Germany. His research interests surround enterprise modelling, as he specifically focusses on supporting enterprise modelling as a creative and

reflective process of business planning and enterprise engineering.

Balbir Barn is Professor of Software Engineering and Deputy Dean in the School of Science and Technology at Middlesex University, UK. Balbir has over 15 years commercial research experience working in research labs at Texas Instruments and Sterling Software. His research is focused onmodel driven software engineering where the goal is to use models as abstractions and execution environments to support, for example, enterprise architecture and application integration using complex events. He has led numerous externally funded projects which apply model-driven principles to business processes, learning theories and more recently the theory building aspects to model-driven engineering.

Tony Clark is Professor of Informatics and Head of the Computer Science Department in the School of Science and Technology at Middlesex University. Tony has experience of working in both Academia and Industry on a range of software projects and consultancies. He has worked on languages for Object-Oriented specification and design including contributing to a range of Industry standards (UML 2.0, MDA, QVT, MOF) that led to a spin-out company which Tony co-founded in 2003 and served as Technical Director 2003-2008. The company sold UML-based meta-tools and acted as consultant to a number of blue-chip companies. Tony is an editorial board member of the SoSyM Journal and has edited several special issues including IEEE Software and SoSyM.

1 Introduction

Processes, describing the behaviour of enterprises, usually include activities performed by a variety of different stakeholders [1, 2]. Among other stakeholders, these are business-related individuals of the enterprise as well as those that are in charge for the support given by an IT-Infrastructure. Therefore, the execution of processes relies on the interpretation of the given description by a variety of stakeholders in order to complete their specific tasks. Regarding the development of software systems aiming at a purposefully support for the achievement of business goals, it is necessary that not just the business related individuals understand the intentions and the taken directions, but also those individuals that are in charge of providing an infrastructure for the execution of the business-related actions [3]. The participation of the stakeholders of the company is not just restricted towards the execution of a process. Furthermore it is necessary for them to validate the specification of a process in terms of the fulfilment of their specific requirements [4].

The general proceeding of negotiating the specific requirements and the accomplishing of a common understanding among stakeholders, is approached through discussions [5]. Discussions offer the opportunity to question certain design decisions and the proposals of responses and justifications. Usually all proposals given in a discussion are formed in terms of natural language. However, this provokes a tremendous threat towards the validation of process models, because process models will be validated by their relevant stakeholders only after their specification or a step of the specification is completed [4]. An explanation given by the specification is not available, since the product of the discussion does not incorporate the various reasons for design decisions. Requirements stated by stakeholders are translated in a form, educible by the respective modelling language. Moreover regarding the evaluation, the translation has to be understood by the various stakeholders after the

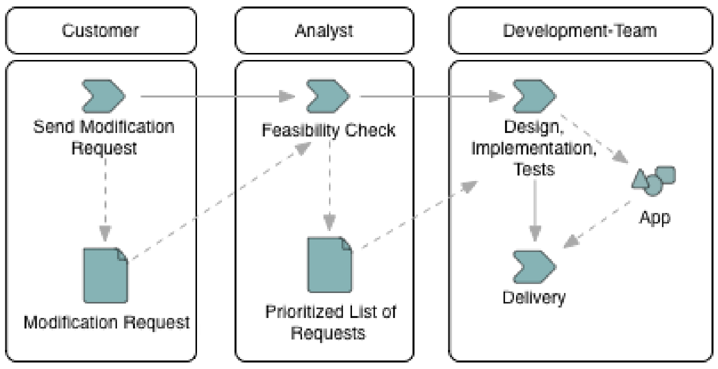
Customer
Analyst
Development-Team
Send Modification Request
Feasibility Check
Design, Implementation, Tests
App
Modification Request
Prioritized List of Requests
Delivery

Figure 1 An exemplary process model defined with SPEM

specification. Therefore stakeholders have to restrict their validation to the final product of a complex and highly structured reasoning process [6]. So a serious gap between the modelling language and natural language has to be overcome in every specification of a model. A possible way to close the elucidated gap between the modelling language and natural language is to hold discussions by means of the modelling language. The arguments given in a discussion would then be well formed with a reference towards a certain well-defined modelling language and the contribution by a given argument, whether it is valid or not, can then be derived automatically based on the language's syntax. Such an approach holds three valuable promises. *First*, the evolution of the supplemented knowledge that is created in a specific universe of discourse becomes traceable through the explicit made arguments. *Second*, the underlying design decisions can be explained with a reference to a given argument. *Third*, the knowledge, which is used to support an argument and therefore a design decision, is explicitly depicted. Hence the actual convictions that have led to a certain process model are made explicit and explainable. Respectively, it will be shown that these attributes of modelling lead to a better management and maintenance of models, especially of large enterprise models. Additionally, the benefits for design decision traceability will be outlined during the illustration of the research. Ultimately, it will be discussed how the design rationale behind a model can be reused in other models, e.g. to prevent redundant wrong decisions, by the specific application of the ArgML.

During this paper it will be shown how the creation, the progress and the maintenance of method engineering can be supported with respect to the Argumentation Modelling Language, which is capable of discussing conceptual models by means of a modelling language. The next section 2 discusses various approaches that are aiming at enabling of discussions about conceptual models by meaningful languages. In section 3 the Software & Systems Process Engineering Meta-Model will be introduced, whose concepts will be used for the specification of process models. The following section 4 will give the specification of the Argumentation Modelling Language, whose concepts will be used for structuring discussion about process models. Section 5 elucidates the application of the Argumentation Modelling Language with the Software & Systems Process Engineering Meta-Model as a underlying language and how from such, a process model can be derived. The developed approach will be evaluated with a case study from the domain of method engineering in section 6. The paper will end with a conclusion in section 7.

2 Related Work

The construction of conceptual models as part of the requirements analysis phase of system design is described in [5, 7], in which the authors address the dialogue between stakeholders and modellers. To foster this dialogue, some approaches recommend to provide different representation for different stakeholders [8, 9]. A notable omission in these proposals is a justification of design decisions and how to account for them in the specification process. In particular, clarification of relationships between design decisions is not addressed. Most approaches represent requirements in natural language and don't take the different stakeholder domains into account [10]. To cope with these issues, specific models have been developed to clarify dependencies relations in more detail [11, 10]. However, while these approaches certainly provide a manner for requirements elicitation, they are impeded on a conceptual scale. While these approaches certainly can provide a certain level of soundness, the implications on the actual to-be developed artefact are contempt to any logical form of requirement relations. Thereby, *requirements engineering* approaches, regardless their design-rationale foundation, are not in support of the actual development is not given and the solely interpretation of the stated requirements still lead to ambiguity, misinterpretation, incompleteness and conflict, without providing a certain and holistic form of discourse of design decision in artefacts.

Elsewhere research has tried to include the actual requirements given by stakeholders in a conceptual model, or in the modelling process, so that it becomes possible to capture design decisions leading from a requirement to a solution [12, 13, 14]. Unfortunately these approaches concentrate primarily on the management and change of requirements; the need to translate from natural language to the modelling language remains. Additionally there are approaches, like the waterfall model and its derivatives [15, 16], which can be used to guide the process of modelling. However, with the strict focus on the actual process, the generality of these approaches restrains them to be used with general purpose languages, such as programming languages. Thereby, specific peculiarities of different conceptual models, respective problem domains, cannot be considered and therefore they provide no usage for a detailed explanation. Especially, artefacts that depend on the interpretation by stakeholders, which are not necessarily common with the field of computer science, these *process models of software engineering* cannot be used for revealing the design rationale behind the artefact for its wide audience.

Proposals from the field of *artificial intelligence* exist that try to introduce a certain logical language for formulating arguments [17, 18, 19, 20, 21, 22, 23, 24, 25]. These approaches enable derivations of contradictions in arguments within a discussion. However every proposition given in an argument has to be specified with the general-purpose language defined by the approach. So it is necessary to translate the domain-specific arguments given by stakeholders to a general-purpose representation. In order to generate any implications and finally the conceptual model, it is further necessary to translate the result of the discussion back to the actual problem domain. Any domain-specific knowledge is therefore lost.

Finally, recent approaches try to introduce the reflections of a model, by means of extended concepts of the respective modelling language [26, 27]. These approaches are aware of the participation by multiple stakeholders and the need for discussions and reasoning. Unfortunately the concepts, which they try to introduce, are unaware of the problems with natural language. Furthermore actual discussions are not fully supported, since the concepts only offer to include certain reasons for a given design decision, but the ability for reasoning about a design decision is not possible, due to the ambiguity of natural language. Moreover as soon as the reasons are stated, there is no facility to challenge them, as they are captured only for explanatory reasons.

Certain issues are targeted by ideas of *collaborative modelling* and *design rationale*. Propositions exist that try to guide mostly virtual discussions [28, 29, 30, 31]. In particular, these approaches try to capture the design rationale that result based from the discussion. However, while these approaches enable a structured discussion and reference to TAM, they miss to outline the relation to the actual artefact and in that sense enable a certain fluid process of argumentation, but miss to consider a needed goal relation of the discussion [32, 33]. Thereby, the generally applicability of these approaches impedes the enrichment of the used language for forming arguments.

In conclusion, a structured proceeding of modelling is widely recommended, as the relevance of traceability is wideley accepted. Additionally, the form of argument presented by Stephen Toulmin is also widely recognised and respected as a basis for achieving such a structured proceeding. However, on the one hand approaches exists that only reuse the provided structure and remain with naturally language expressed statements and thereby the reach for a wide audience is given, but the actual software development is postponed [34]. On the other hand, there are logically sound language designs that try to to translate the Toulmin's Argumentation Model to a calculus, which is capable of testing the requirements, but the transition to the to-be designed artefact is restrained [35], due to the focus of the usage of restrictive languages. Further, these approaches impede the collaborative sense through the usage of restrictive languages and the implied low level of accessibility [30].

3 Software & Systems Process Engineering

The Software & Systems Process Engineering Meta-Model (SPEM) [36] is a modelling language, which enables the specification of process models regarding software development. The SPEM is specified by means of the UML. It is supportive to UML Class Diagrams, which are static and SPEM shall be used for expressing dynamic behaviour, such as software development processes [37]. The advantage of the embedment of the SPEM within the UML is that UML related concepts are usable for the specification of a process or method defined with SPEM. The structure of processes within SPEM is mainly characterised by three different concepts [38, 39]. These are the activity, the role and the work product. Detailed descriptions of the concepts can be retrieved from table 1.

The SPEM was selected, due to its suitability and specificity to the domain of software development. The engineering perspective on software processes makes it superior to classical process modeling approaches such as the EPC [40], BPMN [41] or IDEF0 [42]. While proper extensions probably exists for both process modeling languages, SPEM was selected due to its straight and its compatibility of its initial state without the need for considering extensions. Specifically the clear structure of the language and the support of documents and artifacts, respectively the work product that is well specified and standardized [36], was the main aspect for choosing it, while the insights remain transferrable.

A specific method is depicted in figure 1. This method is constituted by the process of selecting feature requests by users and providing these features through extended software artefacts. Initially the users, respectively the customers of an enterprise, send modification request. Following, an analyst checks these request for their feasibility. Afterwards, the features will be implemented by the development team and delivered. Although such a

Table 1 Conceptually description of the Software & Systems Process Engineering Meta-Model

Concept	Description
Activity	An activity is part of a process and represents a step of that certain process. Furthermore there is typically one individual responsible for an activity. During the execution, the availability of certain work products may be required. The execution usually completes with a resulting work product.
Role	Usually one specific role is responsible for an activity. Further one or more roles, which may include the owner, are in charge of executing the activity. Furthermore a role is responsible for certain work products, which are created during the execution of activities, either by the role itself or others.
Work Product	A work product is the result of the execution of an activity. Work products may be composed from other work products as well as they contribute to compositions.

specification might be regarded as simple, the specification of such a process model comes with a high effort for negotiating. During the process specification, various participants and their specific tasks and requirements have to be considered. So the respective modeller needs to account for the various perspectives the variety of stakeholders might have even on a simple process. Furthermore regulations for the application of the modelling language have to be considered [38]. For example, enterprise-specific rules have to be considered during the application of the SPEM.

However, the perception of a specific process might change, while the requirements regarding this process develop. The change of requirements follow a set of requirements that are still reasonable, new requirements and requirements, which have become irrelevant. If for example the depicted process in figure 1 has to be adapted, because the process should be outsourced and executed by an external company, there will be new requirements, which are caused by the externalisation of the process as well as there will be requirements that are still valid, e.g. regarding quality criteria, although the process is outsourced.

The current way of adapting process models towards such a change is mainly informal and unevaluated. Participants of a previously held discussion about process models do not necessarily participate in future discussions, although they are the representatives for certain valid requirements. So a validation by these stakeholders is omitted. Furthermore adaptations by upcoming modellers might lack in quality, since the necessity understanding of a model often can't be established through and necessity justifications of design decision can't be inferred from the model. Processes may be purposefully represented, but a sophisticated manner for explanation of the model and participation in the act of modelling missing [26].

4 An Enhancement of Toulmin's Argumentation Model

4.1 Toulmin's Argumentation Model

Toulmin's Argumentation Model (TAM) [removed for review] [43] is a general classification scheme for structured discussions and its concepts can be used for examining arguments.

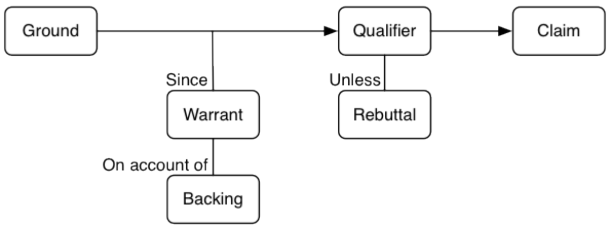

Figure 2 Toulmin's Argumentation Model [43]

Table 2 Concepts of the Toulmin's Argumentations Theory

Concept	Description
Claim	A claim comprises propositions, whose underlying knowledge is sought to be established. The discussion is primarily driven by the required justification of the claims with respect to the already established knowledge.
Ground	Contrasting to the claim, a ground embodies already established knowledge. Grounds illustrate the common knowledge, which can be used to justify claims. Therefore a relation between the claim and its supporting grounds should be revealed.
Warrant	The relation between a claim and its supporting grounds is expressed through a warrant. Warrants describe the implication that leads from the ground to the claim. A warrant is mainly based on convictions and therefore requires the acceptance by the various participating stakeholders. Contrary to a claim and a ground, which address particular entities, a warrant embodies a more general statement.
Rebuttal	A rebuttal is a statement that contrasts a claim. It is constituted by propositions, which can't be aligned with the rebutted claim. A rebuttal introduces a further case that may be valid when the rebutted claim is not. Therefore a rebuttal is a claim, which contrasts at least one further claim.
Qualifier	Whenever multiple contrasting claims exist, there is the need for a decision among them. The qualifier is in charge of aligning its qualifying claim towards those claims that may rebut it. It defines the conditions for the claim to be perceived as valid. Accordingly the qualifier is in charge of deciding when the claim is derivable from its supporting warrants and grounds.
Backing	A backing establishes further knowledge in order to support the respective warrant and its acceptance by the various stakeholders.

Therefore it will be used as a classification scheme for structured discussions about process models, which guide the application of methods. The conceptualisation of an argument by the TAM enables the classification of the various statements given in a discussion. The different concepts of the TAM are the claim, the ground, the warrant, the rebuttal, the qualifier and the backing. These concepts are more explicitly described in table 2. Furthermore the relations between those concepts are depicted in figure 2.

Table 3 Concepts of the ArgML

Concept	Description
Argument	Any argument is a container for a multitude of claims and their rebuttals.
Language	A specific language is uniquely chosen to form the claims contained in one argument.
Concept	Any language, which is used in an argument, consists out of multiple concepts. The warrants and qualifiers can refer these concepts.
Model	Any model is an instance of the chosen language of an argument for expressing the designated claims.
Claim	Every claim embodies knowledge, which shall be established. It is expressed through a model, which follows the syntax of the argument's associated language.
Ground	Grounds delegate claims, which originate from already settled arguments. They are used to justify newly proposed claims by means of already settled claims.
Qualifier	Qualifiers define the validity of the qualifying claim. If the qualifier is satisfied, the claim is valid. A qualifier is expressed through OCL.
Warrant	Warrants define the implication for deriving the claims from the grounds. They are universal expressions, which are specified by means of the OCL.

4.2 TAM applied to Conceptual Modelling

Although the TAM supports a clearer insight of a discussion, two tremendous problems can be identified considering the dependency on natural language. First, the actual propositions still embody possible misinterpretations by the various stakeholders in a high and serious manner. Hence, the understanding of propositions during discussions still comes with a remarkable effort. Second, the actual impact of a discussion regarding the body of knowledge is difficult to infer. Although it might be that after a discussion certain convictions are spread, the actual impact on the universe of discourse cannot be explicated [removed for review] .

Besides the previous elucidations, there is the need for reusing arguments in further discussions in order to promote the reuse of created knowledge and insights. Specifically such a transition of knowledge would enable the creation of mature and advanced process models. Established claims should be used as grounds in prospective arguments to justify new claims. Furthermore the language should be affected by the knowledge and insights proposed by the arguments, thereby the implicit use of that knowledge through the language would avoid the proposal of invalid propositions. Next to the conceptual model, which is gained through the discussion, the used modelling language could be affected by means of certain rules that are derived from the discussion as well. Based on the newly derived insights from a discussion, the affections on the modelling language would reduce flaws in the process of modelling.

4.3 Outline of the ArgML

The proposed approach of the Argumentation Modelling Language (ArgML) is an extension of the TAM with respect to the previous described problems. In particular the use of natural language for the specification of arguments, more specifically of claims, will be excluded by the ArgML. As the specification of conceptual models and especially process models respects well-defined language, the use of natural language for specifying arguments is contrary to such a specification.

The ArgML inherits the various concepts of the TAM. The proposed enhancement extends the TAM with language-based features. The use of natural language within a discussion is excluded by the ArgML. Explicitly defined and well-formed domain-specific languages (DSLs) will be used for the specification of arguments. The use of DSLs contributes to the richness of a discussion as their syntax and semantics are explicitly defined. The syntax and semantics can be used for the actual interpretation of given arguments. Moreover, since DSLs concentrate on a particular domain of interest and have limited expressive power to support just enough expression using concepts from the domain [44].

Next to the well-defined languages, the ArgML is supported by additional mechanisms to express constraints embedded within claims, warrants and qualifiers by the use of the object constraint language (OCL) [45].

Therefore the ArgML extends the set of relevant concepts for discussions appropriately, except for the concept of a backing that is left for future work. The relation between the concepts and in particular the abstract syntax of the ArgML can be retrieved from figure 3. The language was defined according to the language engineering approach described in [removed for review] . Simple UML-style class models define the abstract syntax of the language. A more specific description of the different concepts of the ArgML is given through table 3.

The concept of a claim has two distinct relations towards the concept of a ground. First, grounds support a claim in order to justify them. Second, a ground refers to a claim of a previously settled argument, since any claim of a settled argument can be used to support further claims through a ground. Initially there is no knowledge embodied by the created universe of discourse. The initial used modelling language embodies the only available knowledge. This strictly results in the necessity for stating ungrounded claims in order to start the discussion. So the initial contributions to the universe of discourse are based on tacit knowledge, which is formed by experience and is difficult to justify [46]. The concept of a language, a model and a concept is mainly dependent on the chosen language for the argument. Therefore the illustration of these concepts in figure 3 is rather abstract. For the particular purposes of this paper, the specification of those concepts would strictly rely on the language defined by the SPEM.

5 Discussions about Software & Systems Process Engineering

5.1 Results of an integration between ArgML and SPEM

To overcome the shortcomings of the SPEM, described in section 3, the SPEM should be integrated with the ArgML. Hence the claims given by a discussion that is based on the integration are specified by means of the SPEM. Every design decision is based on an argument, which defines the knowledge it is based on, through the underlying grounds, and the knowledge it seeks to establish, through the claims. Furthermore every argument accounts for the implication from the ground to the claim by means of the qualifiers and warrants. Therefore every model specified by the integration of the ArgML and the SPEM, supports a manner for explanation. In addition, it is possible to rebut already established knowledge of a particular model. Ultimately, the concept of a language in the meta-model

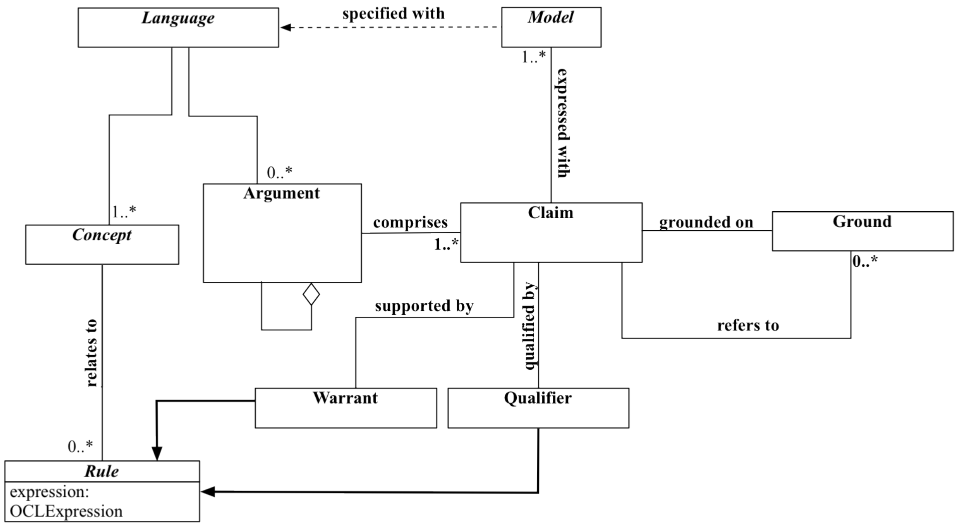

Figure 3 The Argumentation Modelling Language

of the ArgML (cf. figure 3) is exchanged with the SPEM. Therefore every concept of this language can be associated with any qualifier or warrant and every model depicting a claim has to follow the syntactical rules of the SPEM.

The proposed argumentative approach reduces the complexity faced by the respective modellers. It is not required to specify a fully completed process model based on the SPEM language, which would require an understanding about the whole modelling language and respectively the depicted domain. However certain experts can fully rely on their understanding, because their proposals can focus on those aspects of a model that relate to their capabilities regarding their understanding and modelling qualifications. Finally the result of the application incorporates those different perspectives in an integrated manner.

5.2 Derivation of Methods and Processes from Discussions

On the basis of discussions, which were held by the SPEM language and composed regarding the ArgML, actual process models can be derived. Every claim of the given universe of discourse is specified by means of the SPEM. Therefore every claim is a SPEM model or at least an excerpt of a SPEM model. Thereby it is possible to integrate all qualified claims of one discussion in order to create a coherent SPEM model. Furthermore the knowledge of the claims is justifiable with respect to the grounds, which are as well specified according to the SPEM. A common language of the various claim and grounds enables the identification of common concepts. Hence an implication from the grounds to the claims is describable through warrants.

Based on these descriptions, it can be inferred that such a discussion completely disregards the natural language. The claims and grounds of an argument are specified by means of the SPEM. Further the warrants and qualifiers are specified by means of the OCL. Through the OCL it becomes possible to automatically derive whether a claim is sufficiently aligned with its rebuttals with respect to the qualifier or not. Regarding this fact, it becomes further derivable whether an argument, constituted by several claims, can be established or is still in dispute. Hence the outcome of a discussion, respectively a SPEM model, which is constituted by the different qualified claims, can be derived automatically.

A further aspect of the formalisation of the warrants and qualifiers is the possibility to derive rules for the enhancement of the SPEM language, based on the created insights from a discussion. A qualifier specifies the conditions, when a claim can be regarded as valid. Thus the qualifier describes, when the implication given by the warrant is viable. So every qualifier and warrant supporting the identical claim, constitute an invariant that should be comprised by the respective modelling language. For example, an argument could propose that whenever an activity includes the execution of software systems that rely on manual input (Qualifier), a certain role has to be responsible for the execution of this activity (Warrant). If so, then the derived invariant would take the form of an implication, implying the warrant whenever the qualifier is satisfied. The respective language could then embody this invariant in order to promote the newly created knowledge in future projects.

5.3 Enhanced Agility of Method Engineering

Due to the focus on arguments, the participation of multiple stakeholders in the modelling process becomes possible. The formalisation of the arguments by the ArgML shifts the use of a modelling language from a plain and enormous specification of a model to the proposals of arguments. Therefore it can be ensured that an argument is valid by means of the underlying modelling language, which is in this case the SPEM. Claims, which are not expressible by means of the SPEM, cannot be expressed and therefore cannot be claimed [47]. Statements that are not interpretable and consequently do not contribute directly towards the conceptual model are permitted. Through this restriction it can be ensured that only contributions are proposed that fit the universe of discourse.

The continuous dialogue enabled through the discussion, depicts the actual evolution of the conceptual model. On the one hand, this enables the tracing of changes regarding the process model [48]. On the other hand, the process model is directly affected by the continuation of the discussion in two distinguishable ways. First, the propositions, which are given by arguments that are updated as invalid, will be excluded from the particular SPEM model. The evolution of any SPEM model is fostered by the possibility to rebuttal any claim of the discussion. Second, the respective SPEM model will directly include the propositions given by upcoming and valid arguments. Furthermore any valid claim can be used as a foundation for introducing new claims, respectively new knowledge, embodied by the SPEM model.

6 Application of the ArgML with SPEM

6.1 Initial Situation and Problem Description

During the following it will be illustrated, how an adaptation towards new requirements of a process model specified with SPEM can be accomplished with consideration of the ArgML. The respective process model, which shall be adapted, was discussed in section 3 and is depicted by figure 1.

The case studies is specifically considered with outsourcing the particular introduced methodology given in Section 3. Specifically the adaptation with ArgML is used, because of the needed evaluation of the respective requirements. While some of the considerable requirements are at hand, such as the externalization of the specific roles, other requirements need to be gained from the involved stakeholders and the specific analysis of previous made decisions. Within the given example there will be definitely new requirements proposed by the service contractors that are ought to execute the outsourced method, both for checking the feasibility as well as the implementation of the feature requests. Further requirements might consider the communication between the original enterprise and the service contractor, since the enterprise has to define standards that have to be fulfilled during the execution of the outsourced process. These requirements and respectively the adaptations can be embodied during a discussion about the process model by means of the ArgML.

Specifically within the case study, it will be shown, how the ArgML enhances the respective reuse of developed artifacts as well as the extended reusability enables the prevention of mistakes and errors in future projects. Thereby, the case study not just exemplifies how a specific process may be outsourced, but further it is possible to generate policies for future outsourcing projects. In conclusion, such a corset of entangled artifacts supports the increased and more efficient maintenance of the process models.

6.2 Application of the ArgML

The related universe of discourse is depicted by figure 4. The whole universe of discourse is formed by means of the ArgML and the SPEM. Any claim is specifically expressed through a model or an excerpt of a model created by means of the SPEM. While different stakeholders could contribute these claims, they lastly contribute to the same process model.

The initial arguments A and B of the universe of discourse rely on a ground that was derived from the initial process model, which is depicted by figure 1. Hence, claims were derived from the first specification of the process model, which are used as grounds in upcoming arguments. However, these claims are neither grounded nor are they justified, but for the continuation of the discussion it is a premise that these claims are taken for granted. The claims might be justified with reference to a certain experience, respectively tacit knowledge [46]. Certainly, the claims may still be rebutted as the discussion continues.

The first argument A describes partly the initial process model, which hasn't been outsourced yet. It is grounded on the various aspects, which the development has to consider. This argument defines how the method would be executed within the company and more particular the order of activities of the method. The claim of argument A is rebutted by argument B, because of the need for an externalisation. Argument B claims that *preceding a delivery by a service contractor, the enterprise has to approve the artefact*. Furthermore, it includes a role named "Quality Control Inspector", which is in charge for evaluating and approving the new application created by the external software development team. This procedure is argued through a necessity control of service contractors during their interaction with customers.

Next to the externalisation of the development team, argument C is considered with the externalisation of the feasibility check by the analyst. It claims that *any activity, which is external needs to be provided with guidelines*. On that account, a preceding activity of the feasibility check will be introduced that provides it with directives from the original enterprise.

Argument D describes a required reporting by activities that are executed externally. Such activities need to report their results to the original enterprise. So, argument D states *that an activity of an external role that has no successor needs to provide a report that is owned by an internal role*. Hence, the delivery activity needs to produce a notification note

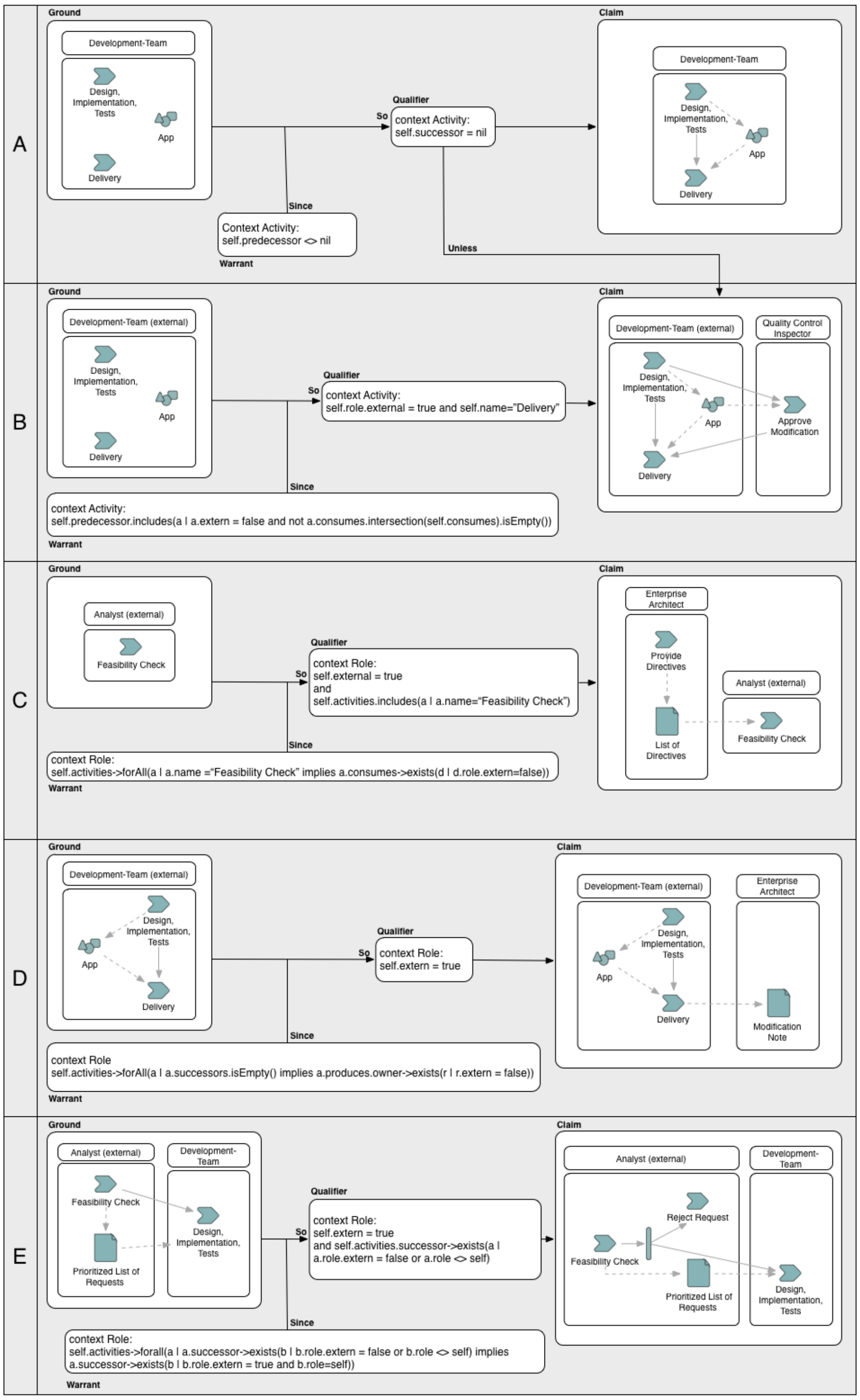
A
Ground
Development-Team
Design, Implementation, Tests
App
Delivery
Qualifier
So
context Activity: self.successor = nil
Claim
Development-Team
Design, Implementation, Tests
App
Delivery
Since
Context Activity: self.predecessor <> nil
Warrant
Unless
B
Ground
Development-Team (external)
Design, Implementation, Tests
App
Delivery
Qualifier
So
context Activity: self.role.external = true and self.name="Delivery"
Claim
Development-Team (external)
Quality Control Inspector
Design, Implementation, Tests
App
Approve Modification
Delivery
Since
context Activity: self.predecessor.includes(a | a.extern = false and not a.consumes.intersection(self.consumes).isEmpty())
Warrant
C
Ground
Analyst (external)
Feasibility Check
Qualifier
So
context Role: self.external = true and self.activities.includes(a | a.name="Feasibility Check")
Claim
Enterprise Architect
Provide Directives
List of Directives
Analyst (external)
Feasibility Check
Since
context Role: self.activities->forAll(a | a.name ="Feasibility Check" implies a.consumes->exists(d | d.role.extern=false))
Warrant
D
Ground
Development-Team (external)
Design, Implementation, Tests
App
Delivery
Qualifier
So
context Role: self.extern = true
Claim
Development-Team (external)
Enterprise Architect
Design, Implementation, Tests
App
Delivery
Modification Note
Since
context Role self.activities->forAll(a | a.successors.isEmpty() implies a.produces.owner->exists(r | r.extern = false))
Warrant
E
Ground
Analyst (external)
Development-Team
Feasibility Check
Design, Implementation, Tests
Prioritized List of Requests
Qualifier
So
context Role: self.extern = true and self.activities.successor->exists(a | a.role.extern = false or a.role <> self)
Claim
Analyst (external)
Development-Team
Reject Request
Feasibility Check
Prioritized List of Requests
Design, Implementation, Tests
Since
context Role: self.activities->forall(a | a.successor->exists(b | b.role.extern = false or b.role <> self) implies a.successor->exists(b | b.role.extern = true and b.role=self))
Warrant

Figure 4 Exemplary Discussion based on the ArgML combined with the SPEM

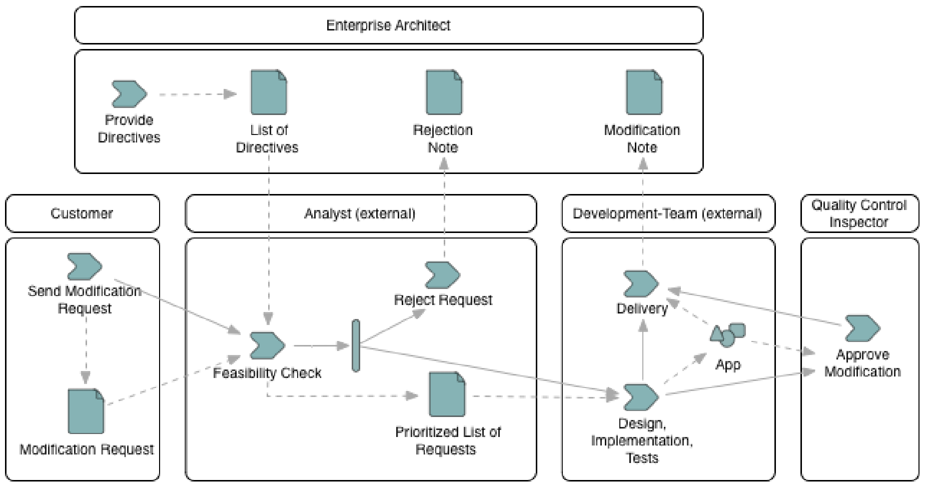

Figure 5 Derived result from the discussion

that provides the enterprise architect with informations regarding the deliverance, e.g. date of deliverance.

Ultimately, argument E focusses on the required communication between the analyst and the development team, which are both related to different service contractors. It states that *every external activity, which is succeeded by another external activity that is executed by a different role, there must be a selection which of the information will be communicated.* Hence, within argument E another succeeding activity of the feasibility check is introduced, which rejects infeasible request. With reference to argument D it can already be inferred that the new introduced activity has to report to the original enterprise. So by means of the ArgML a gap within the argumentation of the new method has been already identified, which will be considered by the final result.

A process model is derivable from the given discussion that is given in figure 5. Furthermore the claims from the discussion can be used to ground additional claims that consider new requirements of the process model. So as soon as new claims become valid, the process model includes the propositions given by the claim. Additionally if a claim becomes invalid, because of a rebuttal, the process model excludes all propositions given by the claim as well as from all claims that are grounded on that particular claim. However, next to the actual process model, the different stakeholders can consider the discussion to get an understanding about the respective domain and certain related design decisions, due to the capture of the discussion by means of the ArgML.

Additionally, the given qualifiers and warrants of the discussion contribute to rules that can be embodied by the respective modelling language. Therefore an implicit use of the gained knowledge can be achieved. For example, upcoming models that follow the SPEM syntax and additionally the rules derived from the discussion have to consider instruction guidelines for every externally located activity they define. As described in the given example, the need for reporting the rejected request by the analysts occurs from the argument D, in which it is generally stated that an external and ultimate activity of a method needs to report to an internal role. In conclusion, every knowledge that was established in the held discussion and is depict-able with the ArgML might be reused in further modelling activities, respectively discussions, regarding SPEM models. Either explicitly through grounding new claims or implicitly through the prevention of invalid proposition based on derived invariants, which are incorporated by the respective modelling language.

6.3 Evaluation

For purposes of evaluation, the implications of using the ArgML for an argumentative-based modelling were quantitatively evaluated. Specifically, for the assessment key indicators were identified for evaluating the reuse of the developed policies and artefacts. Thereby, a quantitative assessment can be given regarding the satisfaction of the initial stated requirements. These indicators are the separation of concerns, average need for consistency check of associated concepts and the number of possible design decisions. For purposes of the evaluation, the definition of the indices follows a definition of a model as a simple graph.

$$G = (E, V), \text{whereby } V = (E \times E) \tag{1}$$

The definition of these indicators is given in the following:

separation of concerns Reflects the number of considerable artefacts that can be perceived as complete. With respect to the ArgML each subgroup corresponds to a specific claim. If the ArgML is not applied then the amount of subgroups equal to one.

$$SG_i = (E_i, V_i), \text{whereby } V_i = (E_i \times E_i) \text{ and } E_i \subset E \wedge V_i \subset V \tag{2}$$

average need for consistency check of associated concepts Represents the average effort for consistency checks in the number of concepts that needs to be checked after adaptations or modifications of a specific concept.

$$traceable(e_1, e_2) = \begin{cases} 0 & \text{If no finite path exists from } e_1 \text{ to } e_2 \\ 1 & \text{If a finite path exists from } e_1 \text{ to } e_2 \end{cases} \tag{3}$$

$$\frac{\sum_{x<i} \sum_{(e_1,e_2) \in E_x \times E_x} traceable(e_1, e_2)}{i} \tag{4}$$

number of possible design decisions Indicates the possible design decisions that can be used to adapt the model, which are then in a semantic conflict to the intention of the model. Specifically, these are reduced by the invariants gathered by the ArgML during modelling.

$$removable(n) = \begin{cases} 0 & \text{Remove results in conflict with invariants.} \\ 1 & \text{Remove does not result in conflict with invariants.} \end{cases} \tag{5}$$

$$\sum_{e \in E} removable(e) + \sum_{v \in V} removable(v) \tag{6}$$

Table 4 Comparative assessment of the ArgML

No.	Indicator	Without the ArgML	With the ArgML
1	separation of concerns	1	5
2	average need for consistency check of associated concepts	19	5,6
3	number of possible wrong design decisions	37	23

In general all of the specific indicators are the result of the logical distinction of modelling decisions based on the ArgML's structure. The following table 6.2 provides a comprehensive overview for the indices applied to the plain model (cf. 5) and the discussion based on the ArgML (cf. 4).

Especially, the number of possible wrong design decisions and its reduction by means of the ArgML, as the requirement for having additional documents for outsourced activities, reflects two aspects. First, one can speak of a successful prevention of malicious design decisions within the respective model. Second, such knowledge, which is inscribed in the respective invariant, can enrich further created models that cope with outsourced activities. Therewith, it was possible to reduce the degree of possible decision by approximately 38 per cent.

However, although the evaluation is limited due to the considered dataset, more general conclusion can be gained from the respective case study. Accordingly, it can be concluded that the approach is especially suited for long-term decisions, such as project and strategic settings or for the design of domain-specific modelling languages. Whereby, the definition of rapidly changing and single-concern models may suffer from a required overhead as required by the ArgML. Also increasingly large models, such as holistic enterprise models can benefit from the approach, as the ArgML enables the prevention of wrong decisions. Knowledge for identifying these decisions is already considered by the warrants and qualifiers, which form the invariants and therefore certain experience can be inscribed accordingly and reuse in further projects. However, to completely benefit from such logically formed separations of arguments, the specific segregation of duties is required. Although, this comes with further maintenance effort, it can be assumed as the case study has shown that the maintenance effort will be compromised by the reduced effort of evaluating design decisions.

7 Conclusion

Within this paper it was shown how a formalised argumentative approach for method engineering and in particular for process modelling could foster the participation of various stakeholders in the specification of process models. The approach of the ArgML was integrated with the SPEM, which enables the discussion about process models.

The approach offers several advantages compared to traditional modelling approaches. *First*, it decreases the complexity for single individuals tremendously. Discussions offer the opportunity to simply state facts about a model, which are formed of concepts the participant is familiar with. A concentration on those concepts that are necessary, regarding the requirements of the participant, is sufficient. *Second*, the explanation of process models becomes possible by means of the respective design decisions. Design decisions are traceable and explainable by their justification. These explanations are formalised and therefore unambiguously accessible even if the initial modeller is not available anymore. Thereby a discussion is not just held during a certain period of time, but may be the result of a continuous dialogue over various periods of time. *Third*, the presented approach offers a multi-perspective creation of process models. Not just one particular modeller contributes arguments, but every stakeholder is encouraged to contribute arguments. *Fourth*, the agility of a process model is substantially increased, due to the possibility of ever evolving discussions about process models. Necessity adjustments can be proposed by means of arguments. The newly proposed arguments are then aligned with the existing body of knowledge, which is constituted by the universe of discourse. Due to the formalisation the adjustments given by arguments, which had become valid, can be included automatically.

As it was outlined within the case study, the application of the ArgML leads to a better management and maintenance of modelled artefacts, due to a purposeful separation of modelled contents with respect to the discussion. Additionally, it enables a better traceability of models, because of the clarification of the impetus for creating a concept by means of the implications given by the argument. Ultimately, the introduced concept of reuse, which not necessarily considers the reuse of available model or parts of a model, but rather on the rationale behind the design decisions. Therewith, the reuse of artefacts is promoted without jeopardising the contribution by overexerting the modeller.

Future research will concentrate on the expressivity of arguments with the ArgML. Furthermore, the expressivity of the arguments might be increased with an enhancement of the SPEM. The possible expressions mostly rely on the concepts of the ArgML, the OCL and the SPEM. Possible enhancements are anticipated in all three of these aspects. The growth of the expressive power of the illustrated approach would supplementary affect the modelling quality, due to more meaningful arguments. Additionally an integration of the SPEM with further languages might be useful, if arguments have to rely on grounds that are expressed through additional languages.

References

[1] T. H. Davenport and J. E. Short. The New Industrial Engineering : Information Technology and Business Process Redesign. *Sloan Management Review*, 31(4):11–27, 1990.

[2] H. R. Jorysz and F. B. Vernadat. CIM-OSA Part 1: total enterprise modelling and function view. *International Journal of Computer Integrated Manufacturing*, 3(3-4):144–156, May 1990.

[3] W.M.P. van Der Aalst. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[4] O.I. Lindland, G. Sindre, and A. Solvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, March 1994.

[5] Sjba Hoppenbrouwers, H A Proper, and T P Van Der Weide. A Fundamental View on the Process of Conceptual Modeling. In L Delcambre, C Kop, H C Mayr, J Mylopoulos, and O Pastor, editors, *Proceedings of the 24th International Conference on Conceptual Modeling*, volume 3716 of *Lecture Notes in Computer Science*, pages 128–143. Springer, Berlin, Heidelberg, 2005.

[6] Graeme Shanks, Elizabeth Tansley, and Ron Weber. Using ontology to validate conceptual models. *Communications of the ACM*, 46(10):85–89, October 2003.

[7] P. van Bommel, S. J. B. A. Hoppenbrouwers, H. A. Proper, and T. P. van der Weide. Exploring modelling strategies in a meta-modelling context. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *Lecture Notes in Computer Science*, pages 1128–1137. Springer, Berlin, Heidelberg, October 2006.

[8] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[9] C. Rolland and C. Proix. A natural language approach for Requirements Engineering. In Pericles Loucopoulos, editor, *Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 257–277. Springer, Berlin, Heidelberg, 1992.

[10] Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Adaptive socio-technical systems: a requirements-based approach. *Requirements engineering*, 18(1):1–24, 2013.

[11] Varsha Veerappa and Rachel Harrison. Assessing the maturity of requirements through argumentation: A good enough approach. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 670–675. IEEE, November 2013.

[12] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, 2001.

[13] Colette Rolland and Naveen Prakash. From conceptual modelling to requirements engineering. *Annals of Software Engineering*, 10(1-4):151–176, January 2000.

[14] A. Egyed and P. Grunbacher. Automating requirements traceability: Beyond the record & replay paradigm. In *Proceedings 17th IEEE International Conference on Automated Software Engineering,*, pages 163–171. IEEE Comput. Soc, 2002.

[15] W. W. Royce. Managing the development of large software systems : Concepts and techniques. In *Technical Papers of Western Electronic Show and Convention (WesCon) (Los Angeles, August 25-28, 1970)*, pages 328–338, Los Alamitos, CA, USA, 1970. IEEE Computer Society Press.

[16] Christian Dawson and Ray Dawson. Software Development Process Models: A Technique for Evaluation and Decision-Making. *Knowledge and Process Management*, pages n/a–n/a, November 2013.

[17] Hercules Dalianis and Paul Johannesson. Explaining Conceptual Models - Using Toulmin's argumentation model and RST. In *Third International workshop on the Language Action Perspective on Communication Modelling*, pages 131–140, 1998.

[18] Daniela V. Carbogim, David Robertson, and John Lee. Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*, 15(2):119–149, June 2000.

[19] Henry Prakken. From logic to dialectics in legal argument. In *Proceedings of the fifth international conference on Artificial intelligence and law - ICAIL '95*, pages 165–174, New York, USA, May 1995. ACM Press.

[20] Ronald P. Loui, Jeff Norman, Jon Olson, and Andrew Merrill. A design for reasoning with policies, precedents, and rationales. In *Proceedings of the fourth international conference on Artificial intelligence and law - ICAIL '93*, pages 202–211, New York, USA, August 1993. ACM Press.

[21] Robert A. Kowalski and Francesca Toni. Abstract argumentation. *Artificial Intelligence and Law*, 4(3-4):275–296, 1996.

[22] Kathleen Freeman and Arthur M. Farley. A model of argumentation and its application to legal reasoning. *Artificial Intelligence and Law*, 4(3-4):163–197, 1996.

[23] Minhong Wang, Huaiqing Wang, Doug Vogel, Kuldeep Kumar, and Dickson K.W. Chiu. Agent-based negotiation and decision making for dynamic supply chain formation. *Engineering Applications of Artificial Intelligence*, 22(7):1046–1055, October 2009.

[24] Thomas F. Gordon and Nikos Karacapilidis. The Zeno argumentation framework. In *Proceedings of the sixth international conference on Artificial intelligence and law - ICAIL '97*, pages 10–18, New York, USA, June 1997. ACM Press.

[25] A Bondarenko. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101, June 1997.

[26] Ali Koudri and Joel Champeau. MODAL: A SPEM Extension to Improve Co-design Process Models. In Jürgen Münch, Ye Yang, and Wilhelm Schäfer, editors, *New Modeling Concepts for TodayâŁ™s Software Processes*, volume 6195 of *Lecture Notes in Computer Science*, pages 248–259. Springer, Berlin, Heidelberg, 2010.

[27] Jesús Gallardo, Crescencio Bravo, and Miguel A. Redondo. A model-driven development method for collaborative modeling tools. *Journal of Network and Computer Applications*, 35(3):1086–1105, May 2012.

[28] Luca Iandoli, Ivana Quinto, Anna De Liddo, and Simon Buckingham Shum. Socially augmented argumentation tools: Rationale, design and evaluation of a debate dashboard. *International Journal of Human-Computer Studies*, 72(3):298–319, 2014.

[29] RosalieJ. Ocker. Promoting Group Creativity in Upstream Requirements Engineering. In John M Carroll, editor, *Creativity and Rationale SE - 11*, volume 20 of *HumanâŁ"Computer Interaction Series*, pages 223–236. Springer London, 2013.

[30] S. Simon, S Johnson, S. Cavell, and T. Parsons. Promoting argumentation in primary science contexts: an analysis of students' interactions in formal and informal learning environments. *Journal of Computer Assisted Learning*, 28(5):440–453, October 2012.

[31] Oliver Scheuer, Bruce M McLaren, Armin Weinberger, and Sabine Niebuhr. Promoting critical, elaborative discussions through a collaboration script and argument diagrams. *Instructional Science*, pages 1–31, 2013.

[32] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, December 1995.

[33] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.

[34] A Nkwocha, J G Hall, and Lucia Rapanotti. Design rationale capture for process improvement in the globalised enterprise: an industrial study. *Software & Systems Modeling*, pages 1–21, 2010.

[35] Xiaoqing (Frank) Liu, Eric Christopher Barnes, and Juha Erik Savolainen. Conflict detection and resolution for product line design in a collaborative decision making environment. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*, page 1327, New York, New York, USA, February 2012. ACM Press.

[36] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0, http://www.omg.org/spec/SPEM/2.0/, 2008.

[37] Elena Nardini, Ambra Molesini, Andrea Omicini, and Enrico Denti. SPEM on test: the SODA case study. In *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, pages 700–706, New York, New York, USA, March 2008. ACM Press.

[38] B. Henderson-Sellers and I. Hawryszkiewycz. Comparing Collaborative and Process Semantics for Cooperative Information Systems. *International Journal of Cooperative Information Systems*, 17(02):155–176, January 2008.

[39] E. Breton and J. Bezivin. Process-centered model engineering. In *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pages 179–182. IEEE Comput. Soc, 2001.

[40] A.-W. Scheer, O. Thomas, and O. Adam. Process Modelling using Event-Driven Process Chains. In Marlon Dumas, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede, editors, *Process-Aware Information Systems*, pages 119–146. John Wiley & Sons, Inc., Hoboken, NJ, USA, September 2005.

[41] P Wohed, W M P Aalst, M Dumas, A H M Hofstede, and N Russell. On the Suitability of BPMN for Business Process Modelling. In Schahram Dustdar, JoséLuiz Fiadeiro, and AmitP. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 161–176. Springer Berlin Heidelberg, 2006.

[42] Soung-Hie Kim and Ki-Jin Jang. Designing performance analysis and IDEF0 for enterprise modelling in BPR. *International Journal of Production Economics*, 76(2):121–133, 2002.

[43] Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, updated edition, 2003.

[44] A. van Deursen, P. Klint, and J. Visser. Domain-specific languages : An annotated bibliography. *ACM SIGPLAN Notices*, 35(6):26–36, 2000.

[45] Jordi Cabot and Martin Gogolla. Object Constraint Language (OCL): A Definitive Guide. In Marco Bernardo, Vittorio Cortellessa, and Alfonso Pierantonio, editors, *Formal Methods for Model-Driven Engineering*, volume 7320 of *Lecture Notes in Computer Science*, pages 58–90. Springer, Berlin, Heidelberg, June 2012.

[46] Kaj U. Koskinen, Pekka Pihlanto, and Hannu Vanharanta. Tacit knowledge acquisition and sharing in a project work context. *International Journal of Project Management*, 21(4):281–290, May 2003.

[47] Ludwig Josef Johann Wittgenstein and Charles Kay Ogden. *Tractatus Logico-Philosophicus: German and English*. Routledge, 1990.

[48] M. Rossi, J.-P. Tolvanen, B. Ramesh, K. Lyytinen, and J. Kaipala. Method rationale in method engineering. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, volume vol.1, page 10. IEEE Comput. Soc, 2000.