Mastering Agile Practice Adoption through a Model-Driven Approach for the Combination of Development Methods

Giovanni Giachetti1[1,2*], José Luis de la Vara[2], and Beatriz Marín[3]

[1]Facultad de Ingenieria, Universidad Andres Bello, Antonio Varas 880, Providencia, Santiago, Chile
[2]Universidad de Castilla-La Mancha, Av. de España, s/n, 02001 Albacete, Spain
[3]Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain

[*]To whom correspondence should be addressed; E-mail: giovanni.giachetti@unab.cl

Many software companies are adapting their traditional development processes to incorporate agile practices. In this context, it is necessary to count on expert knowledge to evaluate different agile practices and configure them according to project needs. However, this expert knowledge is scarce, difficult to validate, and time-consuming, since it is applied manually. As a solution, this paper presents a model-driven approach, called SIAM, which automatically generates guidelines for the adoption of agile practices through the combination of different development methods. SIAM is supported by a meta-model architecture to implement a knowledge repository that characterizes method configuration decisions, which can be reused in different development projects. SIAM has been implemented in a tool suite that facilitates the specification of models and the identification of issues during the definition of the development processes. The approach has been successfully applied to reconfigure an industrial development process with agile methods, showing that the effort required for tailoring agile practices according to organizational standards is considerably reduced.

Keywords - *Agile practices, Process configuration, Automatic verification, Model-driven agile, Knowledge management*

Introduction

In recent years, increasing attention has been focused on agile development method (Cockburn (2006), Fowler (2001)), with a growing number of companies adopting such practices to evolve their development processes. This progress is presented in the annual State of Agile report (Digital.ai. (2021)),which also indicates that the main motivations for adopting agile practices are accelerated software delivery, enhanced ability to manage changing priorities, increasing productivity, and improved business/IT alignment (Tripp and Armstrong (2014)). Given these benefits, many companies have started to adapt their traditional development processes to incorporate agile development practices in order to meet their project requirements (Mahanti (2006), Rao, Naidu, and Chakka (2011)).

However, the transition of organizations and their development practices from a traditional process to an agile one is challenging, and requires great effort from organization and teams (Campanelli and Parreiras (2015)). The reconfiguration of traditional development processes demands specific expert knowledge to adapt existing methods to agile practices that are aligned with organizational needs (Lycett, Macredie, Patel, and Paul (2003)). Generally speaking, this configuration activity is performed manually without systematic guidelines, and is thus susceptible to present inconsistencies due to the lack of automatic verification mechanisms or assisted support for the process configuration (García-Borgoñon, Barcelona, García-García, Alba, and Escalona (2014)).

This paper presents a model-driven approach that uses expert knowledge to guide the configuration of development processes in organizations that are moving from traditional to agile development schemas. It reduces the effort and errors involved in the manual configuration of processes while preserving the alignment with organizational development standards and reference methods. The approach has been implemented in a suite of model-based tools called SIAM (Software Improvement Agile Methods), which facilitates the adoption of agile practices in software development processes that must combine traditional and agile methods to ensure alignment with organizational standards. SIAM has been validated by means of industrial projects, which have obtained ISO 9001 and CMMi certifications.

The aim of the SIAM approach is to manage experts' knowledge of agile development processes and to promote the use of this knowledge among practitioners. To this end, SIAM formalizes the configuration decisions of development processes so that third parties can validate these decisions, and have access to evidence about the fulfillment of organizational procedures and protocols that facilitate the external certification of development projects.

The contribution of this paper is threefold: 1) it introduces the conceptual architecture of SIAM, which aligns expert knowledge with developing methods for their tailoring in concrete projects; 2) it presents the implementation of SIAM in a suite of model-driven tools for the adoption of agile practices in the reconfiguration of traditional software development processes; and 3) its demonstrates how SIAM is used to reduce the effort and errors involved in the reconfiguration of an industrial development process with agile practices from different methods.

The rest of the paper is organized as follows. Section 2) presents the related work; Section 3) presents the novelty and main contributions of SIAM beyond the state of the art; Section 4) presents the conceptual foundation of the SIAM architecture; Section 5) presents the implementation of SIAM in a Suite of model-driven tools to guide the configuration of development processes with agile practices; Section 6) presents the analysis of the approach by considering the results obtained from its industrial application; and Section 7) presents conclusions and future works.

Related Work

The main elements involved in understanding the definition and implementation of SIAM are the model-driven principles applied to method engineering and the application of knowledge management to adopt agile practices. The background to these two aspects is detailed below. Subsequently, the main novelties and contributions of the SIAM approach beyond the state of the art are presented.

Development Method Specification and Application

Several authors, such as (Gonzalez-Perez and Henderson-Sellers (2007)), state that a development method must specify the parts of a process and products to be generated during the software development. This approach indicates that it is possible to represent these elements in a metamodel that uses certain *exotic* conceptual constructs to obtain multi-abstraction-level representations. Most of these concepts are involved in the definition of the ISO/IEC 24744:2014 standard, which presents a multi-level modeling approach for situational method engineering (Henderson-Sellers, Ralyté, Ågerfalk, and Rossi (2014)). Multi-level modeling for process development is also explained by (Atkinson and Kühne (2001)) and (Henderson-Sellers (2006)). The latter differentiates between the method/process definition level and the process enactment level, where the process elements are configured according to specific project requirements. Furthermore, as indicated in (Frank (2019)), traditional method engineering approaches, such as that indicated above, are limited with respect to reuse, *i.e.* new methods would need to be created from scratch using the rather generic concepts defined in the metamodel. This particular issue is important for agile methods that are constantly evolving, motivating the definition of novel method engineering approaches to improve agile practice adoption according to specific organizational needs and changing development environments (Heimicke, Dühr, Krüger, Ng, and Albers (2021)).

Drawing on the referenced works about method engineering, it can be seen that the standards and concepts involved are supported by different meta-specification. Thus, the use of Domain-Specific Modeling Languages (DSMLs) (Frank (2011)) seems to be the most suitable alternative for the definition of specific methods and their application to concrete domains. A DSML allows specific development needs to be captured by means of conceptual constructs, which are specified in metamodels that describe the abstract syntax of the related modeling language. For the definition of a DSML, there are two well-known alternatives: defining a metamodel of from scratch or extending/customizing a reference modeling language, such as UML for general-purpose modeling or BPMN for process modeling. However, there are still gaps in the use of DSML implementations to adopt multi-level modeling in practice, such as appropriate tool support for implementing model editors and bridging semantic gaps (Fonseca, Almeida, Guizzardi, and Carvalho (2021)).

The work presented in (Sandkuhl and Seigerroth (2019)) states that a method model provides structured guidance for performing complex modeling tasks including the expert knowledge involved in modeling decisions, tools, and cooperation principles. The management of expert knowledge for method definition seems to be particularly relevant for agile methods that need to be adapted to project requirements and, at the same time, comply with the quality criteria of projects and organizations involved (Kurapati, Manyam, and Petersen (2012); Qumer and Henderson-Sellers (2008)). However, as stated in (Dybå and Dingsøyr (2008)), little is known about how agile methods are carried out in practice, while, due to the massive adoption of agile methods (Digital.ai. (2021)), different approaches have emerged to formalize and support the particularities of the agile development paradigm (Al-Zewairi, Biltawi, Etaiwi, Shaout, et al. (2017)).

Knowledge Management and Configuration of Agile Processes

A number of studies have shown that knowledge management is key in developing more efficient business processes and quality improvement for software development (Maciel, de Souza, de Almeia Falbo, Felizardo, and Vijaykumar (2018)). Moreover, modeling approaches focus special attention on representation and transference of knowledge in smart and digital environments (Manesh, Pellegrini, Marzi, and Dabic (2020)). Recent studies underline the need for knowledge management in technology development companies that adopt agile practices (Khalil and Khalil (2020)).

In general terms, agile development processes comprise a set of practices that have been created from practitioner's knowledge (ÅGERFALK and Fitzgerald (2006)). Thus, agile methods are not used straightforwardly by practitioners; they select the practices that better fit project requirements, organizational standards, and the characteristics of the development team to configure specific development processes (Campanelli and Parreiras (2015), Tripp and Armstrong (2014)). An example would be to define a development process that combines agile practices from Scrum and XP methods. The study presented by (Kurapati et al. (2012)) shows that the rules that govern design decisions of development processes are poorly, or not at all, documented, thus making it difficult to validate and replicate the decisions made for quality verification tasks and future development processes. The verification of development processes obtained from the tailoring of agile practices is an important aspect to ensure consistency with the principles of the methods involved.

The approach presented by (Liu (2010)) shows the application of a formal specification to facilitate the verification and validation of agile processes. In (Qumer and Henderson-Sellers (2008)),the authors present a framework that evaluates the degree of agility for development processes based on agile methods. This approach places special emphasis on the use of expert knowledge for the composition of agile processes, arguing that most software development knowledge is tacit and resides in people's heads. The importance of expert knowledge in agile adoption is also discussed in (Singh, Singh, and Sharma (2012)).

In addition, existing agile tools are mostly oriented towards the management of project teams, and the results obtained during the execution of the development process focus on specific agile methods. However, the survey presented in (Azizyan, Magarian, and Kajko-Matsson (2011)) shows that existing agile tools are insufficiently flexible to accommodate process changes, to introduce new practices, or to combine different methods. This limits creativity and the reuse of expert knowledge in the context of adapting agile practices to project needs.

Other approaches (Fontana, Meyer Jr, Reinehr, and Malucelli (2015), Gren, Torkar, and Feldt (2015)) are oriented towards evaluating the maturity of agile development processes. The systematic reviews presented by Fontana et al. (2018) and by Henriques and Tanner (2017) show that maturity evaluation can be performed by customization of traditional maturity models, such as the Common Maturity Model (CMM) (Łukasiewicz and Miler (2012)), by defining new maturity evaluation approaches for agile methods (Elnagar, Weistroffer, and Thomas (2018), Yin, Figueiredo, and da Silva (2011)), or by means of combination of agile and traditional approaches (Sreenivasan and Kothandaraman (2019)). These maturity evaluation approaches require an appropriate specification of the processes to be evaluated, clearly identifying the activities and outcomes obtained.

There exist different alternatives for the representation of development processes, including general modeling language, such as UML (OMG (2017)), notations for business process modeling, such as BPMN (OMG (2011)), and languages specific to development methods and processes, such as ISO 24744 (Sousa, Vanderdonckt, Henderson-Sellers, and Gonzalez-Perez (2012)) or SPEM (OMG (2008)). Domain-specific languages have also been developed, *e.g.*, for safety-critical systems (de la Vara, Marín, Ayora, and Giachetti (2020), Ratiu, Nordmann, Munk, Carlan, and Voelter (2021)). García-Borgoñon et al. (2014) conducted a systematic literature review on different approaches for process modeling. They reported that the need for tools to guide appropriate software process definition is an open challenge. This is precisely one of the challenges for agile development processes that SIAM aims to tackle.

Novelty and Contributions of SIAM beyond the State of the Art

SIAM follow the principles of situational method engineering, where a development process can be tailored or enacted from method components that are selected according to specific project needs (Henderson-Sellers et al. (2014)). Additionally, methods can evolve according to organizational contexts (Franch et al. (2018)). Broadly speaking, the analyzed approaches and standards for method engineering are based on the configuration of development processes from a unique method reference, which limits the use of different methods for development process configuration (Frank (2019)). Thus, SIAM differs from the referenced method engineering approaches, since it supports the configuration of development processes by means of the combination of practices from different methods, which is common practice in the agile development domain (Campanelli and Parreiras (2015), Tripp and Armstrong (2014), Kiv, Heng, Kolp, and Wautelet (2018)). This is also a requirement from the companies involved in the project, to provide tools that allow practitioners to choose the adequate reference method (or methods) according to the project domain and specific development needs.

Moreover, traditional multi-level approaches for method engineering involve meta-concepts that can be multi-instantiated (Atkinson and Kühne (2001)) and (Henderson-Sellers (2006)). Currently, appropriate tools for implementing model editors under this paradigm cannot be found (Fonseca et al. (2021)). Thus, SIAM also contributes by providing a modeling architecture for method and process configuration that is supported by existing modeling standards and tools, and which has been applied in an industrial context.

Another important requirement from the industrial partners involved in SIAM is how they can adopt agile practices to improve their development methods and maintain compliance with standards for which they are already certified, such as ISO 9001 or CMMi. This presents two challenges for SIAM:

1) to provide process configuration guidelines that allow practices from different agile methods to be used to replace practices from the original methods used in the development companies, while maintaining the same developing standards and quality criteria already defined in the organizations.

2) to maintain a task-centered (or process-centered) approach for representing the development processes, since the certification and quality evaluation activities involved in the organizations are based on the analysis of tasks and artifacts involved. This is clearly observable in CMMi, where quality of software depends on the quality of the process (Gonzalez-Perez and Henderson-Sellers (2008)).

Thus, it would be difficult to obtain development processes that only involve agile practices to solve these challenges. This is a well-known issue in companies certified under standards that are not specific to agile, and, hence, a combination of agile and traditional practices (Sreenivasan and Kothandaraman (2019)) to meet the different quality criteria is implemented.

Another contribution of SIAM is thus to provide a process-centered approach to support agile practice adoption when organizations are moving from traditional developments and to maintain consistency with existing quality assurance and certification processes. This is a particularly complex task, which, before SIAM, the companies involved only resolved manually, appealing to the knowledge of their most experienced practitioners.

Furthermore, it is worth noting that the process-centered approach has certain limitations in relation to alternatives such as those related to product-centered specifications (Gonzalez-Perez and Henderson-Sellers (2008)). A product-centered approach provides greater flexibility in task definitions for agile processes configuration, *i.e.*, tasks are not mandatory elements in the method and can be adapted according to the products to be obtained. This can be partially solved by SIAM's capability to move across different methods and combinations of methods, according to project needs.

It is also important to mention that SIAM is not intended to be a complete method engineering approach. It is focused on the process side of method specification, adopting some of the existing definitions and providing the necessary adaptations or extensions at conceptual and implementation levels to fulfill the requirements previously indicated. Moreover, since the SIAM approach centers mainly on task configuration for adapting existing development processes, some concepts have a simpler semantics than that proposed in traditional method engineering approaches. For instance, the artifact concept in SIAM provides a minimum core to validate the approach in the application context involved. However, there are specifications, such as that presented in ISO ISO/IEC 24744 for the *WorkProduct* concept, that provide more fine-grained semantics for this conceptual construct, which has also been ontologically analyzed in (Ruy, de Almeida Falbo, Barcellos, and Guizzardi (2014) and Gonzalez-Perez, Henderson-Sellers, McBride, Low, and Larrucea (2016)) .

From the knowledge management perspective, the review presented in (Manesh et al. (2020)) suggests that knowledge application has been the subject of the least research. Knowledge application is the process by which knowledge, either tacit or explicit, is reused within an organization. In this context, SIAM provides the following two contributions:

1) it characterizes development methods and tacit, or poorly documented, method configuration knowledge in a model-driven method repository

2) it provides model-driven tools that take advantage of the knowledge repository to facilitate the specification and verification of agile processes defined for specific projects.

In summary, SIAM provides a conceptual architecture and implementation based on open-source technologies that can be used as a reference to put existing method engineering concepts and knowledge management about adopting agile methods into practice.

SIAM Conceptual Architecture

As mentioned, SIAM differs from other method engineering approaches that propose the use of multi-level metamodeling (Fonseca et al. (2021)), where a unique metamodel is instantiated twice: it is initially instantiated to obtain a specific method definition that is later instantiated to obtain a process model aligned to the method defined (Atkinson and Kühne (2001), Henderson-Sellers (2006)).

The SIAM conceptual architecture considers the use of one instantiation level metamodels; *i.e.*, a method metamodel is instantiated into method models, and a process metamodel is instantiated into process models (see Fig. 1). Thus, the SIAM conceptual architecture has been defined as a multi-model approach with different instances of method and process models that are interrelated by means of reference models or model weavings (Jossic, Del Fabro, Lerat, Bézivin, and Jouault (2007)). With the definition of these model weavings, it is possible to configure development processes from practices from different methods, which is a challenge in the adoption of agile practices. Moreover, SIAM conceptual architecture permits traceability of equivalencies among different method models to be define, which is then used to interchange method practices when configuring a process model. Furthermore, when implementing SIAM, existing model-driven technologies for the generation of modeling tools and graphical editors (such as Eclipse Modeling Tools - EclipseFoundation (2021)) do not provide proper support for multilevel metamodels.

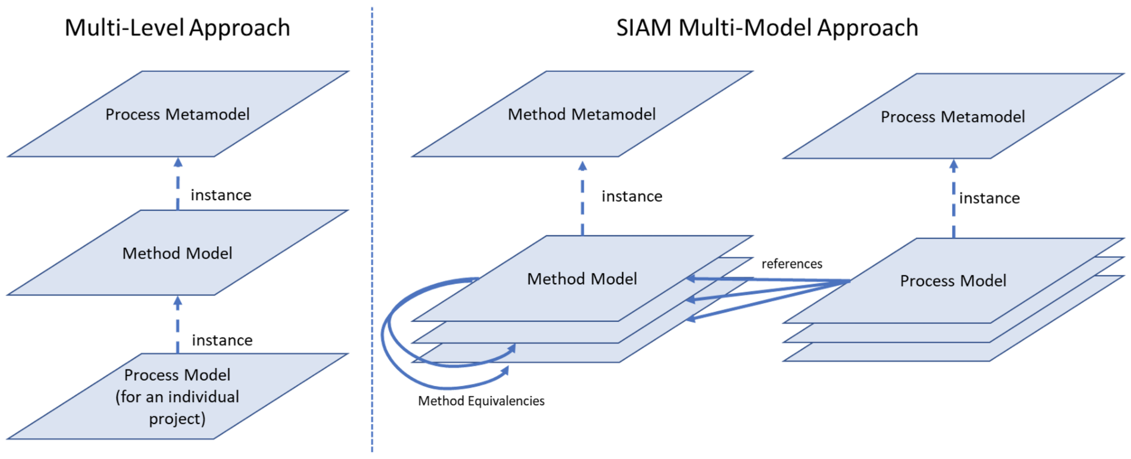


Figure 1: Multi-Level approach vs SIAM Multi-Model approach

For the definition of the models involved, SIAM has the following two specification levels:

- 1. The Knowledge Repository Level, which has the conceptual information related to method activities and artifacts; dependencies among activities; and equivalencies among methods.

- 2. The Process Definition Level, to represent the development activities that must be performed for specific projects. This is the end-user level of the SIAM approach, which uses the knowledge repository to drive the process configuration.

As its name suggests, the elements at Knowledge Repository Level are defined by considering expert knowledge for the specification of development methods.

Fig. 2 shows a general vision of the relationships between the conceptual elements of the two specification levels. In this figure, the activities of process X are referencing specific activities of the methods in the Knowledge Repository Level. These references are represented by black arrows. The main concepts of the SIAM architecture for method definition are explained below.

Method Activities. Method Activities. A method activity indicates a concrete action that is performed in the context of a method. These activities are related by using dependency links, which means

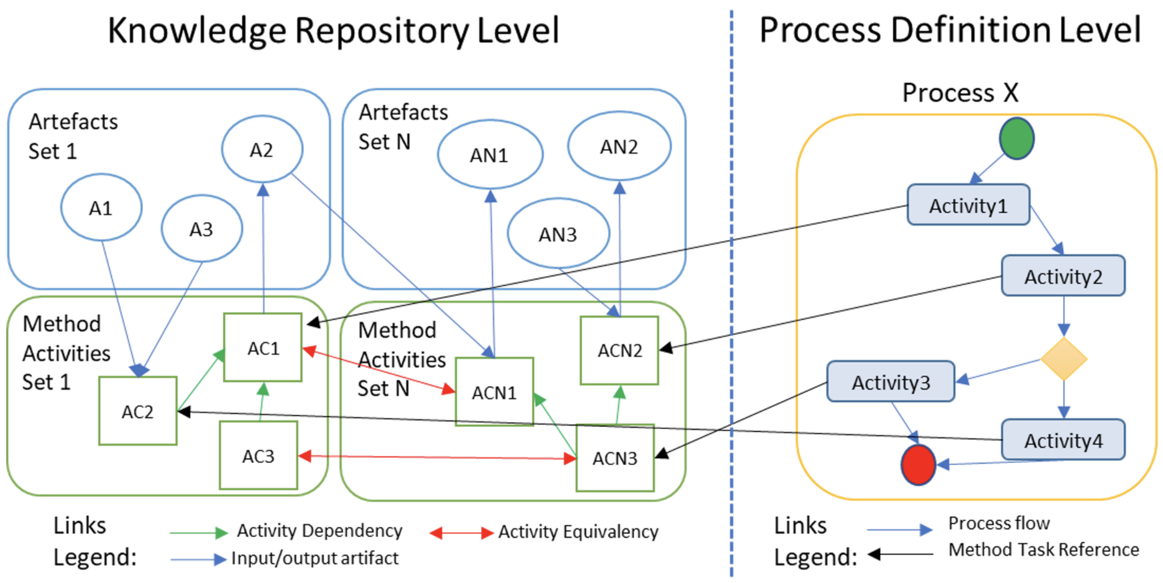


Figure 2: Overview of SIAM Conceptual Architecture Relationships

that certain activities are required to perform the action involved in another activity. Fig. 2 shows the activities *AC2 and AC3* related to activity *AC1* by green arrows. This indicates that *AC2* and *AC3* require *AC1* to be previously performed in a development process.

Activity Dependencies. SIAM considers three types of dependency links between method activities:

- **Backward Dependency:** From an activity A to an activity B, indicating that activity B must be performed before activity A. For instance, the pre-game phase of the Scrum agile method requires the Scrum team be defined before the sprint planning in a development process.
- **Forward Dependency:** From an activity A to an activity B, indicating that activity B must be executed after activity A. Following the example of the Scrum pre-game phase, the product backlog is specified once the product objectives are defined.
- **Parallel Dependency:** From an activity A to an activity B, indicating that activity B must be executed at the same time as A. For instance, the Scrum Master must be defined together with the Product Owner in the configuration of the Scrum team.

The dependencies do not indicate in which specific moment of the process the activities must be performed. For instance, in the example, *AC2* and *AC3* can be performed at different moments in a development process, but must be performed at some instant before activity *AC1*. Thus, activity dependencies are important to guide and validate the configuration of processes, especially when defined as a composition of practices from different methods.

Activity Artifacts. Activity Artifacts. The artifacts represent input resources of an activity, or outputs obtained from an activity execution. For instance, the blue arrows in Figure 1 show, at Knowledge Repository Level, that activity *ACN1* uses input Artifact *A2* and generates Artifact *AN1* as output. In this case, activity *ACN1* is using artifacts that are related to two different methods. As an example, the Backlog definition related to the Scrum method can use, as input, the product requirements or objectives that are defined by different requirement elicitation methods, even traditional ones. The Backlog definition activity generates the Backlog specification as output.

Weaving Metamodels

The SIAM conceptual architecture enables a flexible method configuration, which in turn facilitates the reuse of concepts. For instance, it would be possible for the same artifact to be used by different method configurations, *i.e.*, it is unnecessary to duplicate the artifacts for each method that use the same concept. Furthermore, it would be possible to configure new methods as a composition of different method activities, an example being to combine activities from Scrum and XP to configure a new agile method.

A general overview of the links between the different metamodels of the SIAM Knowledge Repository Level are presented in Fig. 3. The most important metamodels of the SIAM Knowledge Repository used to perform the analysis and configuration of processes are the weaving metamodels for activity equivalencies and method configuration (represented as blue boxes in Fig. 3).

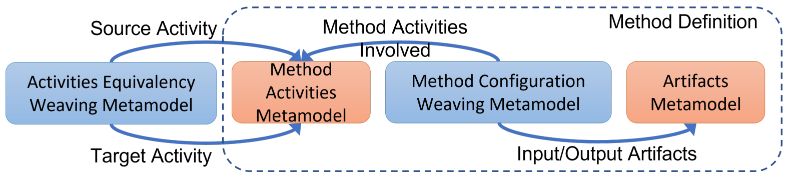

Source Activity
Method Activities Involved
Method Definition
Activities Equivalency Weaving Metamodel
Method Activities Metamodel
Method Configuration Weaving Metamodel
Artifacts Metamodel
Target Activity
Input/Output Artifacts

Figure 3: Relationships between the metamodels of SIAM Knowledge Repository Level

Activity Equivalencies Weaving Metamodel. The activity equivalencies indicate replacing alternatives among the different methods defined, *i.e.*, activities from method *A* that can be replaced by activities from method *B* in a process definition. These equivalences are also used to validate the processes configured as a composition of methods to ensure their consistency. For example, in Fig. 2, the process activity *Activity3* is referencing the method activity *ACN3* that has dependencies on the activities *ACN1* and *ACN2*. These dependencies indicate that activities equivalent to *ACN1* and *ACN2* must be performed before *"Activity3"* in the process. Note that for *"ACN2"*, the dependency condition is met, since it is referenced by *"Activity2"*. Nevertheless, *"ACN1"* is not referenced by any activities in the process defined. Nonetheless, the dependency condition is still met because *"ACN1"* is equivalent to *"AC1"* from the *"Method Activities Set 1"* of the Knowledge Repository Level, and *"AC1"* is referenced by *"Activity1"* in the Process Definition Level. It can also be observed that *Activity1* and *Activity2* are performed at different moments in the processes; *Activity1* is performed first, despite the dependency from *ACN3* pointing to activities *ACN1* and *ACN2* at the same time. In this case the, dependency condition is met, since the only requirement is that the corresponding process activities, *Activity1* and *Activity2*, be performed before *Activity3*.

The method activity equivalencies are also used to indicate missing activities to obtain a sound process configuration or to recommend alternatives in the reconfiguration of an existing development process. The Activity Equivalency Weaving Metamodel is presented in Fig. 4, which shows the two types of equivalencies that can be defined between method activities: alternative activities and mirror activities, which are represented by the *ActivityAlternative* and *MirrorActivity* metaclasses, respectively.

- **Alternative Activities:** When A source activity *A* from a Method *X* can be replaced by one or more activities *B* to *Bn* from a method *Y*. these kinds of equivalencies are unidirectional, *i.e.*, activities B to *Bn* can replace activity *A* but not vice-versa. All activities indicated by the equivalency must be included, otherwise, the process will be incomplete in relation to the referenced methods. For instance, Project Planning from a traditional (waterfall-like) method can be replaced by the Release Plan Definition and the Configuration of Ceremonies from the Scrum method. This equivalency can be represented by means of weaving links from the traditional activities model to the Scrum activities model. Thus, the two Scrum activities must be used to replace the project planning activity; if only one Scrum activity is included in the process instead of the original (traditional) activity, the process configuration will be incomplete and will not be consistent with the original specification. However, this equivalency is not bidirectional, and, in a Scrum-based process, it would thus not be possible to replace the activities Release Plan Definition and Configuration of Ceremonies by a traditional Requirement Elicitation, as there could be other aspects of these Scrum activities that are not covered by the traditional activity. In the case of the bidirectional equivalency of activities being possible, a new equivalency weaving must be defined from the Scrum activities model to the traditional (or waterfall) activities model to represent the opposite equivalency.

- **Mirror Activities:** In this case, a source activity *A* from a Method *X* can be replaced by only one activity *B* from a Method *Y* that represents the same concept. These kinds of equivalencies are bidirectional because they refer to the same activity that is specified in different methods, although the name can differ from one method to another. This type of equivalency reduces the computational analysis effort when configuring a process, since it is unnecessary to review the dependencies of the related methods for the equivalent activity. An example of mirror activities can be a requirement elicitation task defined in different methods.

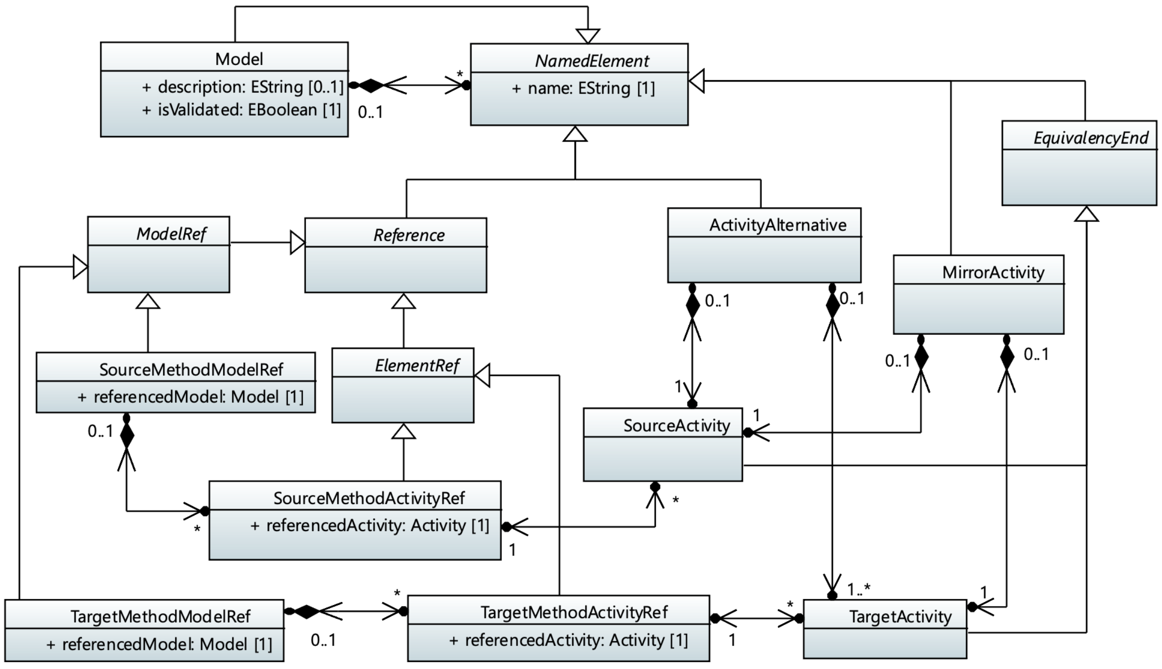


Figure 4: Activity Equivalency Weaving Metamodel

In the Activity Equivalency Metamodel, it can also be observed that the equivalency models are defined in two methods: one source Method (Metaclass *SourceMethodModelRef*) and one target method (Metaclass *TargetMethodModelRef*). Each equivalency model must be validated by experts to ensure that the definitions are correct in terms of the methods involved. The property *isValidated* iis included in the metaclass *Model* to indicate the models that have been validated.

Method Configuration Weaving Metamodel. The method practices are configured in terms of activities and the different related input/output artifacts. A method activity can have different combinations of related artifacts, and, hence, the process designer's experience is fundamental to determine the valid combinations of activities and artifacts involved to ensure consistency with the reference method. This development process configuration is error-prone and highly time-consuming when performed manually. The weaving metamodel for method configuration is presented in Fig. 5.

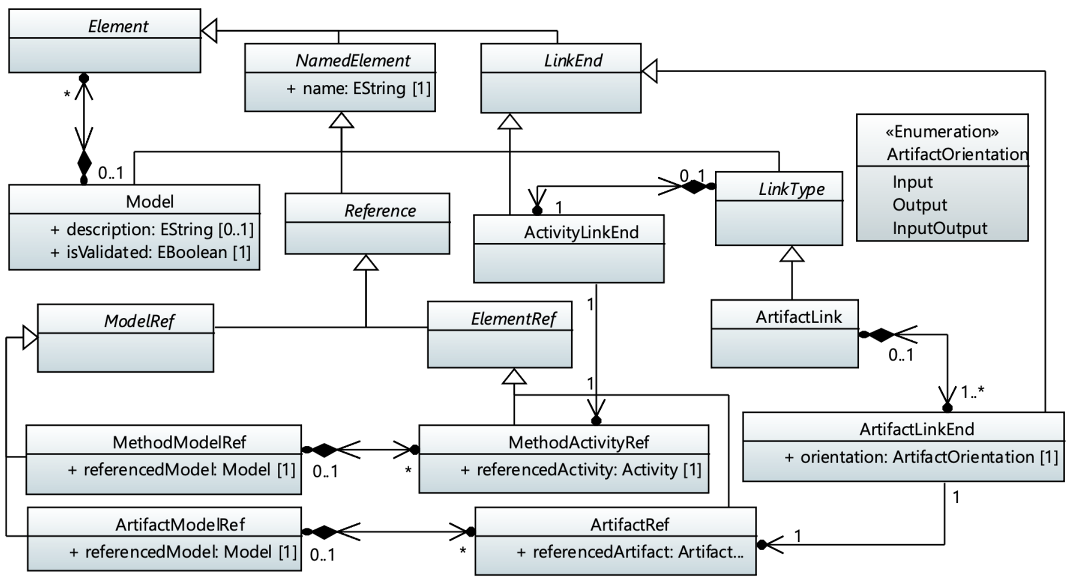

Element
NamedElement
+ name: EString [1]
LinkEnd
«Enumeration» ArtifactOrientation
Input
Output
InputOutput
Model
+ description: EString [0..1]
+ isValidated: EBoolean [1]
Reference
ActivityLinkEnd
LinkType
ModelRef
ElementRef
ArtifactLink
MethodModelRef
+ referencedModel: Model [1]
MethodActivityRef
+ referencedActivity: Activity [1]
ArtifactLinkEnd
+ orientation: ArtifactOrientation [1]
ArtifactModelRef
+ referencedModel: Model [1]
ArtifactRef
+ referencedArtifact: Artifact...
*
0..1
1
1..*

Figure 5: Method Configuration Weaving Metamodel

The metamodel presented in Fig. 5 shows that a method activity and the artifacts related are configured the with metaclass *ArtifactLink*. Moreover, the artifacts related to a method activity can be input artifacts, output artifacts, or both (input/output). This artifact directionality is indicated in the enumeration *ArtifactOrientation*, which evaluates the attribute orientation of the metaclass *ArtifactLinkEnd*. As in the Method Equivalences Weaving Metamodel, this metamodel must be validated by using expert knowledge; the attribute *isValidated* in the metaclass *Model* indicates this situation.

The next section exemplifies how the models and weavings from the Knowledge Repository Level are used to generate guidelines for the configuration of development processes and automate their verification.

Using the Knowledge Repository for Process Configuration

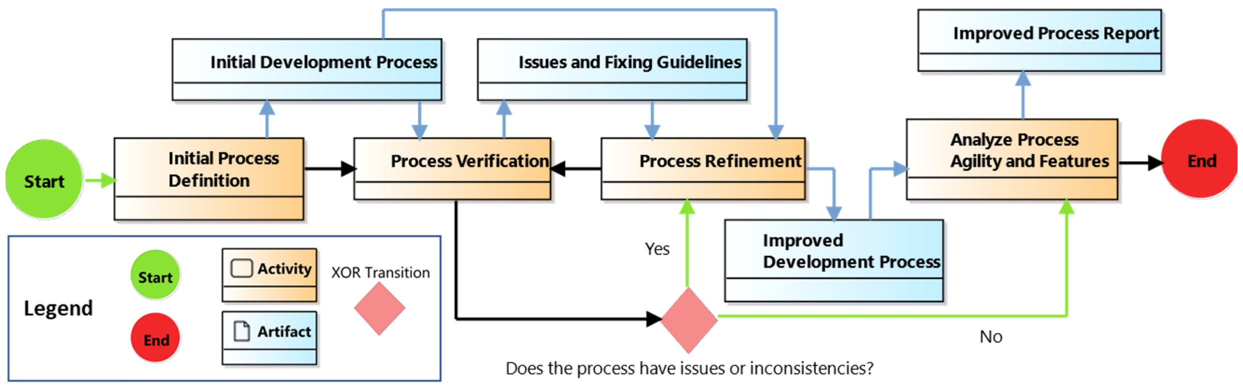
The metamodels of the SIAM conceptual architecture are instantiated in different models according to the methods configured at the Knowledge Repository Level. The information on these models is used to generate guidelines for practitioners to configure development process models at the Process Definition Level, and to verify the consistency of the resultant processes in relation to the reference methods.

The core idea behind the use of expert knowledge is quite simple. A process designer defines an initial process, which is automatically analyzed with the information from the knowledge repository by using references from the process activities to the method activities. This analysis indicates the missing elements or dependency conflicts, accompanied by the alternatives to solve them. The designer chooses one of the alternatives indicated to refine the process. Thus, SIAM reduces the definition effort for the process designer through the automatic generation of guidelines that indicate the appropriate combination of tasks and related artifacts that must be defined according to the methods involved.

Each time the process is modified, the verification is automatically performed by the SIAM process editor to identify any pending issue with regard to the methods involved. This verification considers the soundness and consistency of the process with regards to method activities, artifacts, and dependencies defined by experts at the Knowledge Repository Level.

The refinement process ends once no conflicts or missing elements remain. Finally, a report is generated to indicate the ratio of agile activities defined, the methods referenced, dependencies, method activities referenced with their alternative activities, and method artifacts involved from the process defined.

Fig. 6 shows the steps for process configuration within the SIAM approach.



Start
Initial Process Definition
Initial Development Process
Process Verification
Issues and Fixing Guidelines
Process Refinement
Improved Development Process
Yes
No
Does the process have issues or inconsistencies?
Analyze Process Agility and Features
Improved Process Report
End
Legend
Start
End
Activity
Artifact
XOR Transition

Figure 6: SIAM Process configuration

The SIAM process editor was implemented by using a subset of the BPMN abstract syntax (OMG (2011)) to reduce the modeling complexity and learning curve by means of a well-known modeling approach at academic and industrial levels. The main BPMN conceptual constructs considered in the SIAM editor are Activities, Artifacts, Transitions, and Subprocesses.

Activities. The activities are the basic concept in a process definition and represent a specific task to be executed. The process activities can refer to activities from the different methods defined in the knowledge repository. In this way, a process can be defined as a composition of different methods.

Transitions. The transitions describe different paths the activities of a process can follow. These are intermediate elements connected to the activities by means of links. Input links go from one or more activities to the transition, whereas output links from the transition to other activities might be executed depending on the type of transition represented.

- **AND transition**, which indicates that the next step of the process involves the execution in parallel of two or more paths of activities.
- **OR transition**, which indicates that, in the next step of the process, one or more paths of activities are executed depending on a specific condition.
- **XOR transition**, which refines the OR transitions by allowing the execution of only one path of activities after evaluating the condition.

Subprocesses. A subprocess is used to group a set of activities (small processes) inside the main process. They typically represent the workflow related to specific development stages that comprise the main development process.

Artifacts. The artifacts describe the input or output resources of the activities defined. Although, at the Knowledge Repository Level, a method activity can have different configurations of input/output artifacts, an activity can only have one configuration of input/output artifacts in a process definition; *i.e.*, a process activity references one configuration of artifacts from the Knowledge Repository Level according to the requirements of the development project.

Waterfall and Scrum Methods Configuration

The core of method definition considers a set of method activities and dependencies between them. When certain method practices are used in a concrete process, the dependencies indicate there are activities that must be executed in a previous stage of the process (backward dependency), in a later stage of the process (forward dependency), or activities that must be executed together (parallel dependency).

The example presented in Fig. 7 shows an initial development process that has been defined with the SIAM method editor tool, following a traditional (waterfall-like) development method. This initial process will be analyzed to automatically identify agile practices that can be used to reconfigure the original process. Thus, an equivalent agile (or more-agile) process will be obtained as result.

Following the example, an agile method is defined to provide a set of agile practices that can be used to re-configure the initial development process. For this purpose, a reduced Scrum method was defined,

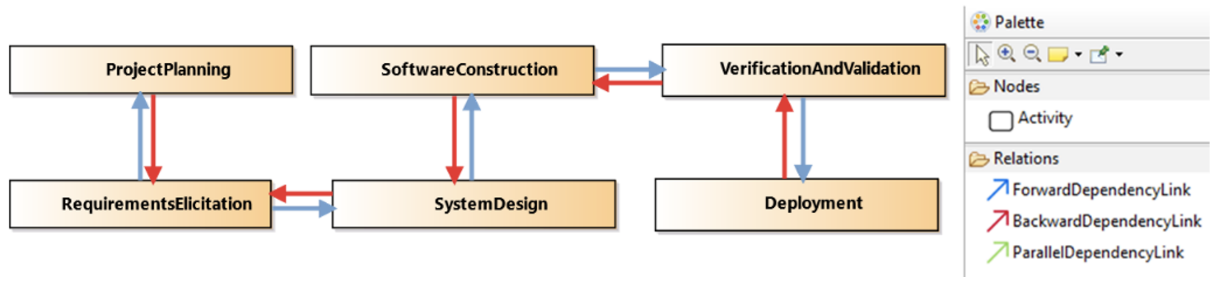
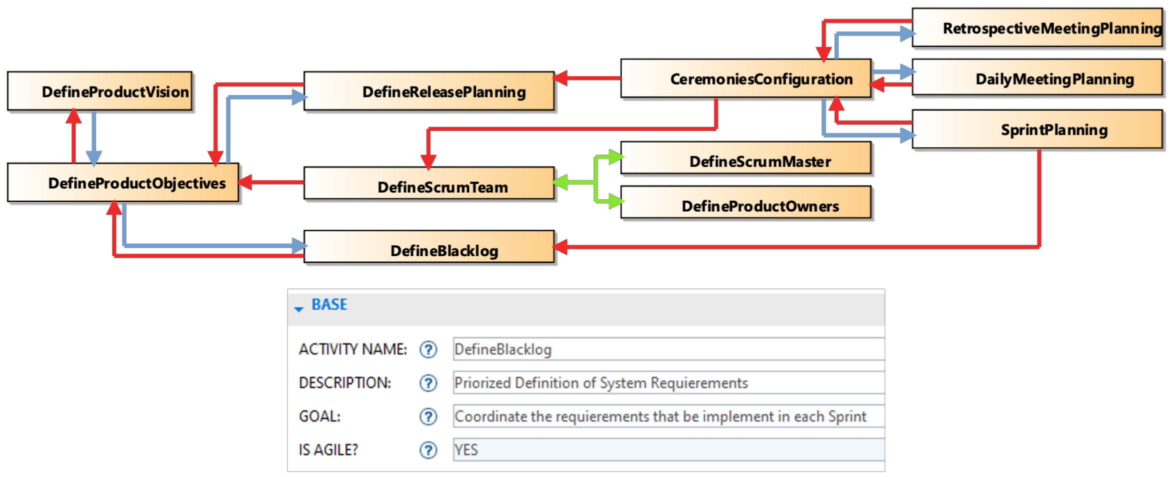


Figure 7: Example of method definition for the Waterfall method

which considers elements related to project planning and project configuration from the pre-game phase (Schwaber and Beedle (2002)).

Real development methods are typically defined as a composition of agile and non-agile activities. In the method model, the agile activities can be indicated by means of a specific property. This information is particularly valuable to identify the agile practices involved in the configuration of a development process. Fig. 8 shows the Scrum method defined and the properties related to the activity *DefineBacklog*, where it is possible to observe the *isAgile* boolean property set as *YES*.



ACTIVITY NAME:	DefineBlacklog
DESCRIPTION:	Priorized Definition of System Requierements
GOAL:	Coordinate the requierements that be implement in each Sprint
IS AGILE?	YES

Figure 8: Example model related to the pre-game phase of the Scrum method

To simplify the example and facilitate its explanation, all the method activities from the waterfall method are defined as *non-agile*, and all the method activities for the Scrum method are defined as *agile*. The mapping of equivalencies is always defined between the activities of two methods, a source method, and a target method. The methods referenced are instances of the method configuration weaving metamodel with their corresponding activities and artifacts. Thus, in the repository there are as many equivalency models as pairs of methods to be analyzed. Moreover, the equivalencies are analyzed in a unidirectional manner, from the source method to the target method. For a bidirectional equivalency analysis, two pairs of equivalency models are required. In the example, the waterfall method is used as source and the Scrum method as target. Fig. 9 shows the equivalency model defined.

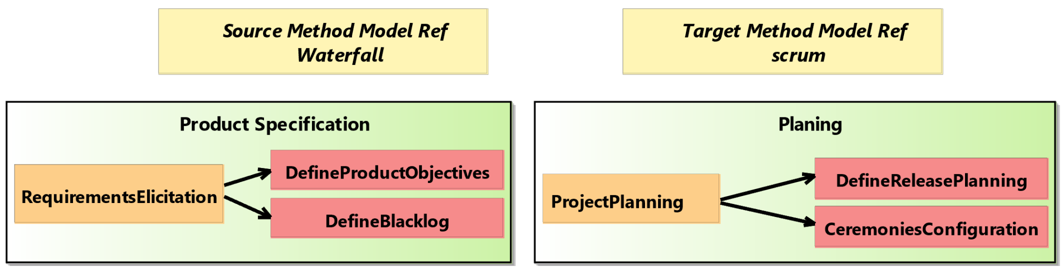
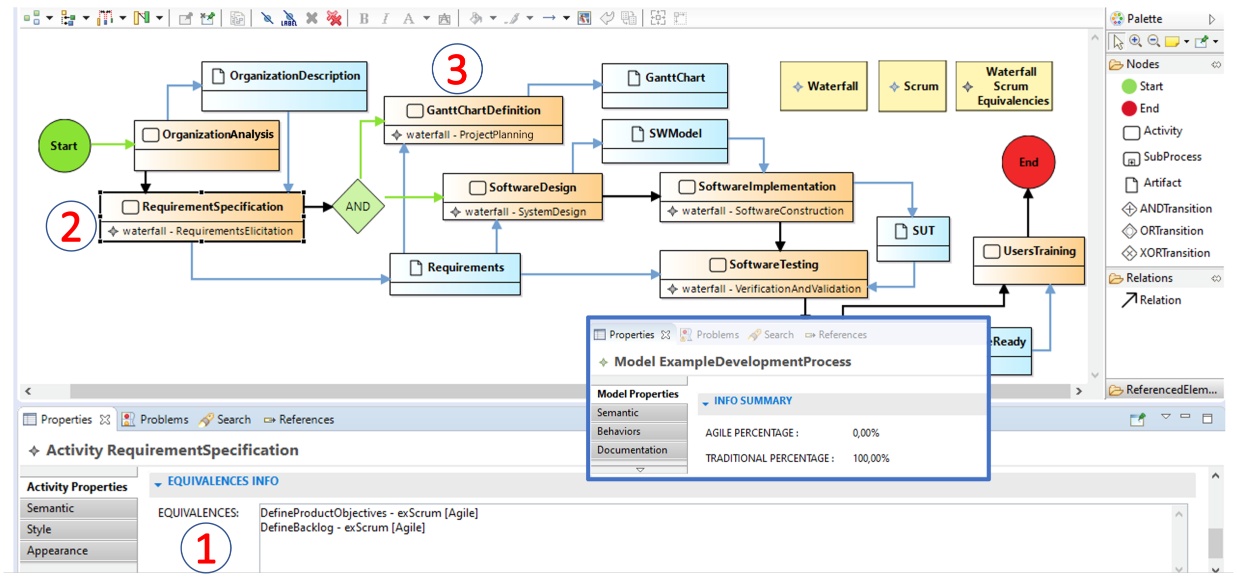


Figure 9: Activities equivalency model defined between the Waterfall and Scrum methods.

With the equivalency models information, it is possible to provide configuration alternatives for the specification of a development process. For instance, Fig. 10 shows the equivalencies information for the activity *RequierementSpecification* of the process model (see Fig. 10 - number 1). The activity *RequierementSpecification* is referencing the waterfall activity *RequierementElicitation* (see Fig. 10 - number 2). In this case, and according to the equivalency model defined, *RequierementElicitation* can be replaced by *DefineProductObjectives* and *DefineBacklog* from the Scrum Method.



Model ExampleDevelopmentProcess

INFO SUMMARY

AGILE PERCENTAGE : 0,00%

TRADITIONAL PERCENTAGE : 100,00%

Activity RequirementSpecification

EQUIVALENCES INFO

EQUIVALENCES: DefineProductObjectives - exScrum [Agile]
DefineBacklog - exScrum [Agile]

Figure 10: Equivalency identification for the example process.

In Fig. 10, it can also be observed that the equivalency model Waterfall Scrum Equivalencies has been referenced by the process model (represented as a yellow box) to indicate that this model will be used to analyze the configuration alternatives. With the SIAM approach, it is possible to reference as many equivalency models as methods wanted to be analyzed to configure the development process.

Furthermore, the SIAM tool calculates the ratio of agile activities from the total activities defined in the process by analyzing the *agile* and *non-agile* (considered traditional) activities defined. The example process is initially referencing activities of the waterfall method, which are all defined as *non-agile* (or traditional) activities. For this reason, the process has an agility ratio of 0% and 100% of traditional activities defined.

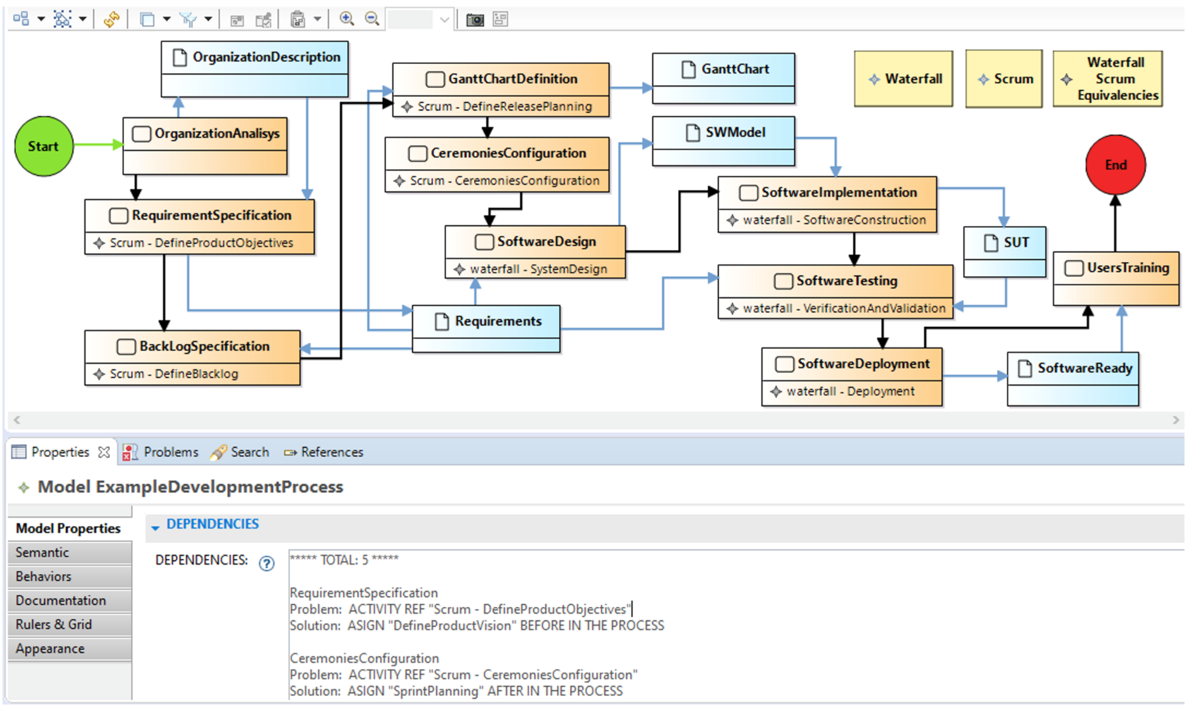
Considering the equivalencies between the different activities defined in the Knowledge Repository, Scrum practices can be used to guide the reconfiguration of the process and increase the percentage of agility. This reconfiguration involves replacing some original (traditional) activities with agile ones, and defining new activities to be consistent with the different methods referenced. At this point, the analysis of dependency models and method configuration models is particularly key to verify the completeness and correctness of the process defined.

Fig. 10 shows the equivalencies for the process activity *RequirementSpecification* (see Fig. 10 - number 1), which indicates that the referenced waterfall activity *RequierementsElicitation* can be replaced by the Scrum activities *DefineProductObjectives* and *DefineBacklog*. To perform the process reconfiguration, the original reference to the waterfall activity *RequirementsElicitation* is replaced by the Scrum activity *DefineProductObjectives*. This change indicates the process activity will now be performed as indicated in the Scrum method, thus becoming an agile activity. Moreover, since *DefineBacklog* is also necessary according to the equivalency information (see Fig. 10 - number 1), a new activity named *DefineBacklog* will be defined in the process, which references to the Scrum activity with the same name. Similar action is performed for the activity *GanttChartDefinition* (see Fig. 10 - number 3), which references the waterfall task *ProjectPlanification*. According to the equivalency model defined (see Fig. 9), *ProjectPlanning* can be replaced by *DefineReleasePlanning* and *CeremoniesConfiguration*. The rest of the activities have no Scrum equivalencies, and so the tool is unable to provide more replacement alternatives.

Fig. 11 shows the result of modifying the process model with the suggested alternatives. Based on these changes, the SIAM tool indicates other Scrum activities that are required according to the method dependencies. In this case, five dependency issues were derived from the Scrum method:

- One (1) equivalency issue was found for the process activity *RequirementSpecification*, which references the Scrum method activity *DefineProductObjectives*. The activity *DefineProductVision* must be defined earlier in the process.
- Four (4) equivalency issues were found for the process activity *CeremoniesConfiguration*, which

references the Scrum activity with the same name (*CeremoniesConfiguration*). In this case, the activities *SprintPlanning* (issue 1), *DailyMeetingPlanning* (issue 2), and *RetrospectiveMeeting-Planning* (issue 3) must be defined after the activity *CeremoniesConfiguration* in the process, and the activity *DefineScrumTeam* (issue 4) must be defined earlier in the process.



```
***** TOTAL: 5 *****

RequirementSpecification
Problem:  ACTIVITY REF "Scrum - DefineProductObjectives"
Solution:  ASIGN "DefineProductVision" BEFORE IN THE PROCESS

CeremoniesConfiguration
Problem:  ACTIVITY REF "Scrum - CeremoniesConfiguration"
Solution:  ASIGN "SprintPlanning" AFTER IN THE PROCESS
```

Figure 11: Dependencies evaluation for the example process.

It is interesting to observe that the dependencies for the waterfall method are still met with the changes performed in the process. For instance, the activity *SoftwareDesign* references the waterfall activity *SystemDesign*, which has a backward dependency on the waterfall activity *RequirementsElicitation*. This last activity (*RequirementsElicitation*) is not referenced in the process model due to the changes in the process activity *RequirementSpecification*. Despite this missing reference, the process model is still correct, since the dependency is met because *RequirementElicitation* is equivalent to *DefineProductObjectives* and *DefineBacklog*, which are defined before *SoftwareDesign* in the process model. The equivalencies between the methods involved are also considered by SIAM to verify the process model defined. In this simple case, manual analysis of all the dependencies and equivalencies involved can clearly be a complex and error-prone task, with these problems being considerably increased in larger process models that involve multiple development methods. Another aspect to consider is that each time a new method activity is referenced in the process model, new dependency issues may appear. Thus, the process configuration requires all the dependency issues be solved to obtain the final process (see Fig.

12).

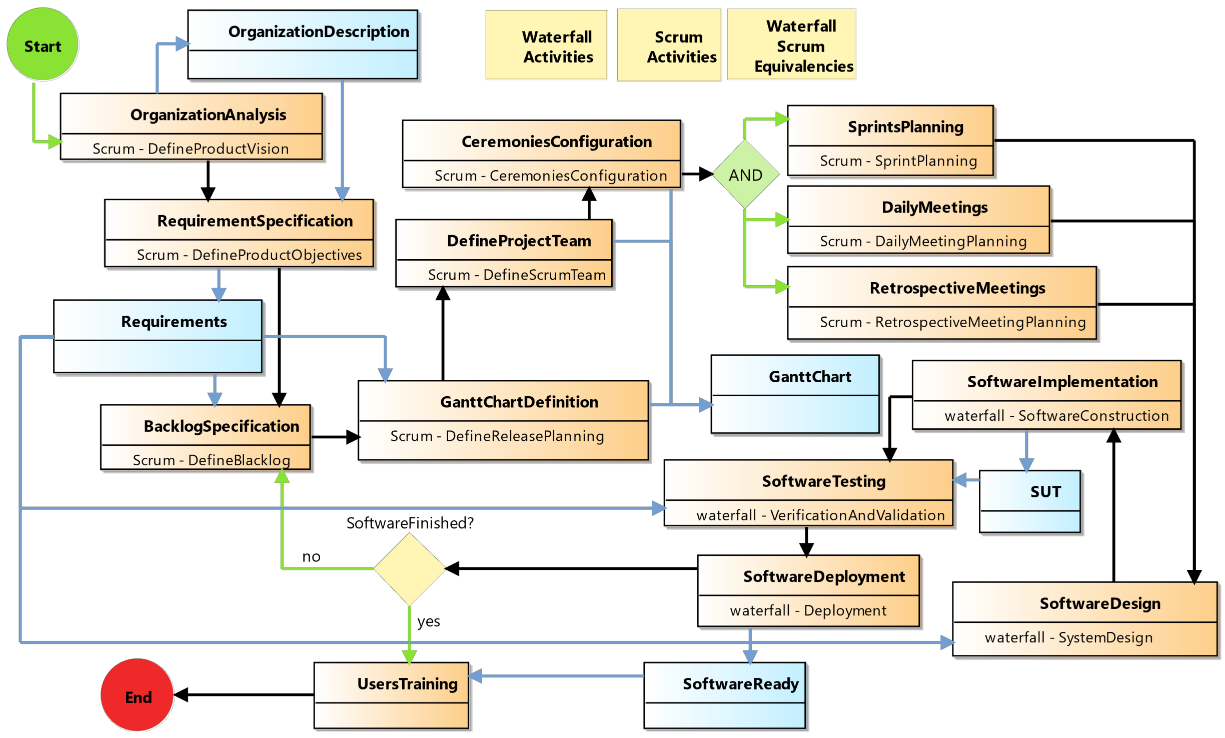


Figure 12: Development process obtained after solving dependency issues.

The SIAM tool also analyzes the soundness of the process model in relation to the method configuration models to determine missing activities or artifacts. If an issue is identified, specific guidelines are presented to solve that issue. The method configuration models are represented by yellow boxes in the process model. In this case, the *Scrum Method Configuration* model has been included, the specification of which is presented in Fig. 13.

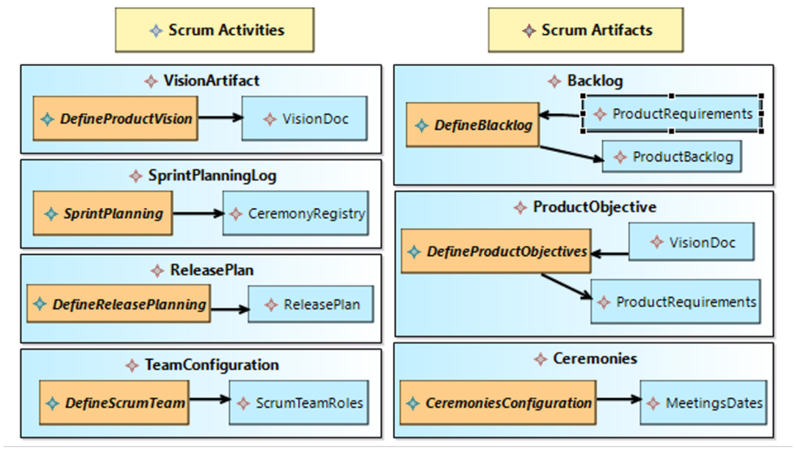


Figure 13: Scrum method configuration model.

The method configuration indicates the set of artifacts that can be involved when a method activity is executed in a concrete process. Each set of artifacts for a certain activity is specified with the construct

ArtifactLink (see the metamodel in Fig. 5). The configuration model defined for the example Scrum method (Fig. 13, shows the construct *ArtifactLink*represented by means of a light blue box that contains the set of artifacts used as input, output, or input/output by an activity. For instance, the artifact link *Blacklog* indicates the activity *DefineBacklog* of the Scrum method uses *ProductRequirements* as input artifact and generates *ProductBacklog* as output artifact.

The results obtained from the process analysis performed by considering the method configuration models is presented in an automatically generated report. This report provides relevant information on the process to facilitate its validation from third parties, and to determine the design decisions made, as well as other configuration alternatives that can be considered for future refinements. The report generated includes a table with the analysis obtained for each activity of the process defined. Fig. 14 shows the analysis automatically obtained from the example process model with the following structure:

- **Referenced Activity** indicates the activity name in the process and the method activity referenced with the following format: **ProcessActivityName** – MethodActivityName (MethodName).
- **Process Artifacts** indicate the artifacts used by the process activity and its input/output properties with the following format: **ArtifactsName** (input/output property)
- **Method Artifacts** indicates the Artifacts recommended by the method for the referenced activity with their input/output property. It also indicates the different set of artifacts for the activity involved in case that be more than one. Finally, Method Artifacts indicates whether an artifact recommended by the method is related (or not) to the process activity analyzed with the following format: MethodArtifactName (SetMethodArtifactName) (**Input/Output** property/**Included** or **Not Included**).

Finally, the process presented in Fig. 15 is obtained. The analysis report is especially important in generating a correct process model. Otherwise, it would be difficult to find the appropriate configuration for the different activities in the process from the knowledge repository information, which may contain many different sets of artifacts and configuration alternatives for each method activity.

It is worth noting that the process obtained fulfills the different validation criteria in terms of dependencies, equivalencies and artifacts defined in the knowledge repository to combine Scrum and Waterfall methods. This brief example is useful to understand how the SIAM approach works. To validate the ap-

Referenced Activity	Process Artifacts	Method Artifacts
OrganizationAnalysis - DefineProductVision (Scrum Activities)	**OrganizationDescription** (Output)	VisionDoc(ScrumArtifacts) (**Output** / **Not included**)
RequirementSpecification - DefineProductObjectives (Scrum Activities)	**OrganizationDescription** (Input) **Requirements** (Output)	ProductRequirements (ScrumArtifacts) (**Output** / **Not included**) VisionDoc (ScrumArtifacts) (**Input** / **Not included**)
GanttChartDefinition - DefineReleasePlanning (Scrum Activities)	**Requirements** (Input) **GanttChart** (Output)	ReleasePlan (ScrumArtifacts) (**Output** / **Not included**)
CeremoniesConfiguration - CeremoniesConfiguration (Scrum Activities)	**GanttChart** (Output)	MeetingsDates (ScrumArtifacts) (**Output** / **Not included**)

BacklogSpecification
- DefineBlacklog
(Scrum Activities)
	Requirements (Input)	ProductBacklog(ScrumArtifacts)
		(**Output** / **Not included**)
		ProductRequirements(ScrumArtifacts)
		(**Input** / **Not included**)

SprintsPlanning - SprintPlanning (Scrum Activities)	**None**	CeremonyRegistry(ScrumArtifacts) (**Output** / **Not included**)
DefineProjectTeam - DefineScrumTeam (Scrum Activities)	**GanttChart** (Output)	ScrumTeamRoles (ScrumArtifacts) (**Output** / **Not included**)
DailyMeetings - DailyMeetingPlanning (Scrum Activities)	**None**	"DailyMeetingPlanning" does not have Artifacts related in the reference method.
RetrospectiveMeetings - RetrospectiveMeetingPlanning (Scrum Activities)	**None**	"DailyMeetingPlanning" does not have Artifacts related in the reference method.

Figure 14: Analysis report generated for the example process.

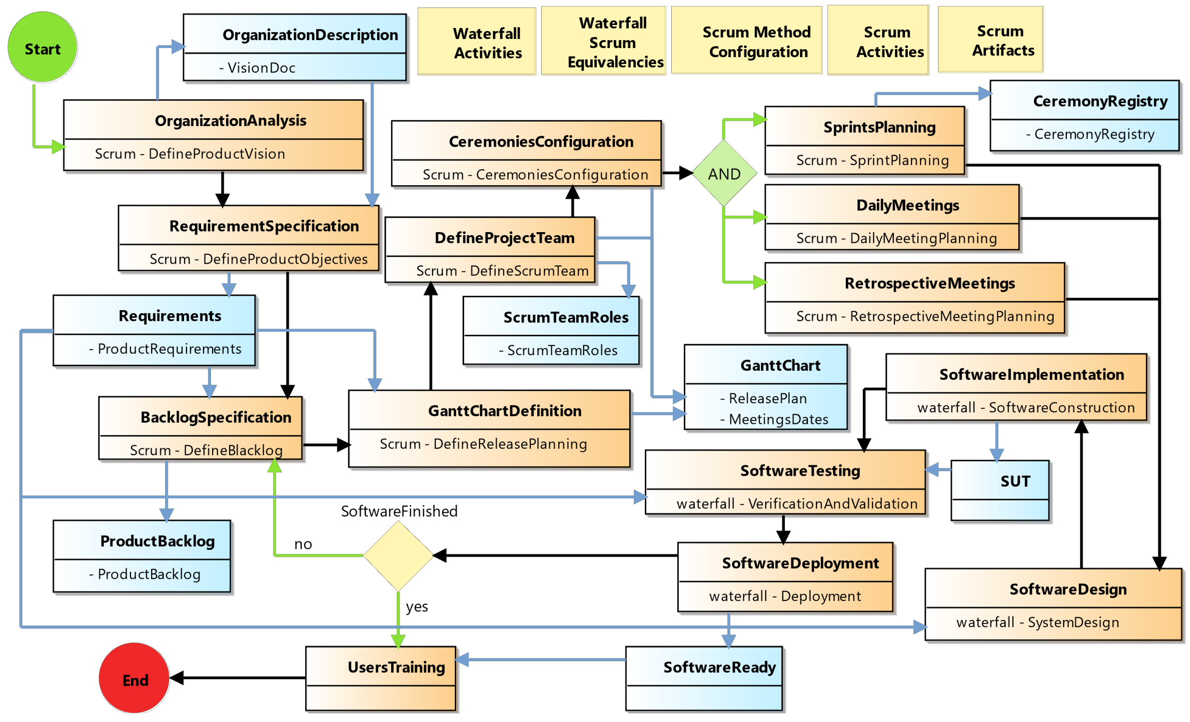


Figure 15: Process obtained after the Analysis of method configurations.

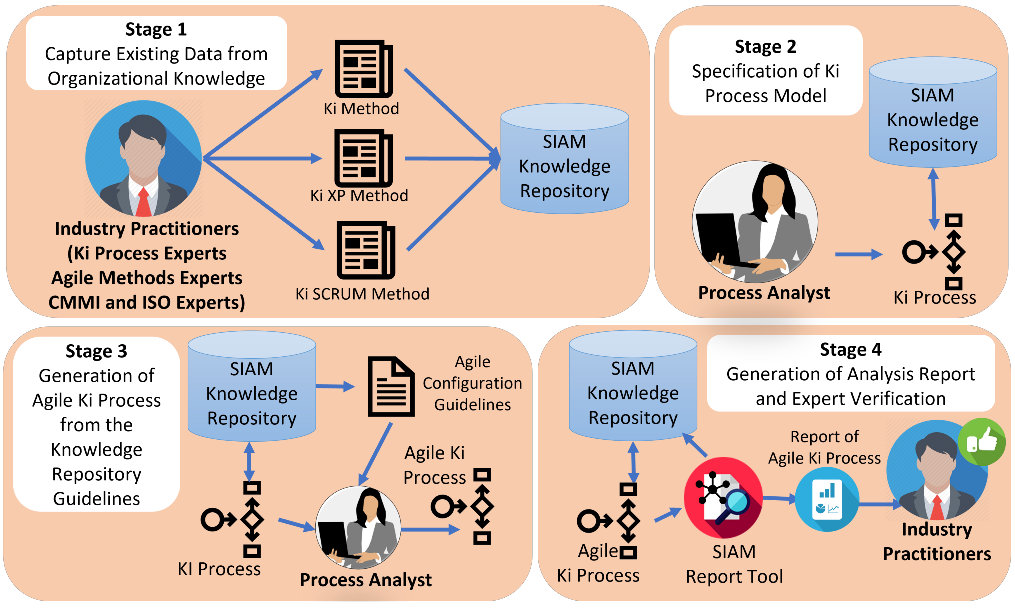
plicability of the approach in larger development processes, the next section shows the results obtained from using SIAM in an industrial context.

Analysis of SIAM Applied to an Industrial Development Process

The SIAM approach and its implementation were supported by industrial partners. This section summarizes the results obtained from the reconfiguration of an industrial development process to analyze the contributions and open issues of SIAM. The process involved is called Ki Process, which was mainly defined by using traditional development practices. We applied SIAM to obtain a new version of Ki Process that integrates more agile practices while continuing to be consistent with the original definitions and quality criteria of the company. To do this, we followed well-known guidelines for case studies, such as those proposed by Runeson, Host, Rainer, and Regnell (2012) and Wohlin et al. (2012).

The original Ki Process is based on the Tutelkan Reference Process (TRP) (Valdés, Visconti, and Astudillo (2011)), which complies with two widely-know quality standards: CMMi-DEV (v1.2) (SEI (2006)) and ISO 9001 (International Organization for Standardization (ISO) (2000)) (Mutafelija and Stromberg (2003)).

The Ki Process was certified against CMMi-DEV and ISO 9001 in 2015. Its specification corresponds to a large document (169 pages) that describes the activities, artifacts, and practices that must be configured according to different development needs. Fig. 16 shows the four stages of the application of SIAM to the Ki Process to obtain a new *Agile Ki Process*.



Stage 1
Capture Existing Data from Organizational Knowledge
Industry Practitioners (Ki Process Experts Agile Methods Experts CMMI and ISO Experts)
Ki Method
Ki XP Method
Ki SCRUM Method
SIAM Knowledge Repository
Stage 2
Specification of Ki Process Model
SIAM Knowledge Repository
Process Analyst
Ki Process
Stage 3
Generation of Agile Ki Process from the Knowledge Repository Guidelines
SIAM Knowledge Repository
Agile Configuration Guidelines
Agile Ki Process
KI Process
Process Analyst
Stage 4
Generation of Analysis Report and Expert Verification
SIAM Knowledge Repository
Report of Agile Ki Process
Agile Ki Process
SIAM Report Tool
Industry Practitioners

Figure 16: Stages of the SIAM Application for the Reconfiguration of Ki Process.

Stage 1: Capturing Existing Data from Organizational Knowledge.

As its starting point, the SIAM approach requires the Knowledge Repository being filled with the information of the three methods involved in the process definition and process reconfiguration.

The first method corresponds to the *Ki Method*, which has all the activities, artifacts, and configurations related to the original specification off *Ki Process*. The other two methods are those used to incorporate agile practices into the *Ki process*: *Ki SCRUM*, and *Ki XP*. These methods are interpretations of the agile methods SCRUM (Cervone (2011)) and Extreme Programming (XP) (Paulk (2001)), respectively. The interpretations were defined in line with the knowledge of a group of nine industry experts with more than 5 years of experience in project management and application of development methods to industry projects. Moreover, they also have experience in consultancy related to guiding companies in the adoption of agile practices. The industry domains in which they have worked include banking, retail, telco, and civil aviation. Tab. 1 indicates the specific knowledge related to development method application and quality certification that each expert provides. For confidentiality, we call the experts Expert1, Expert2, etc.

Table 1: Knowledge of the group of experts

Expert	Ki Method	SCRUM	XP	ISO 9001	CMMi-Dev
Expert 1	x				
Expert 2	x				
Expert 3	x				
Expert 4			x		
Expert 5	x	x		x	
Expert 6	x	x	x		
Expert 7	x				x
Expert 8	x			x	
Expert 9	x				x

Tab. 1 shows there are at least two experts for each method or quality model. The *Ki Method* has the highest number of experts, which is logical considering that most of the experts come from the enterprise that defined *Ki Process*.

The definition of the different SIAM models for the 3 methods took 6 months, involving defining activity models, artifact models, method configuration models, and activity equivalency models. This definition was based on an iterative and incremental process. Weekly one-hour meetings were held to analyze the method elements and achieve consensus with the different experts, and a method engineer then used the meeting information to define the different models. The models defined were refined with the information from the following weekly meetings until the group of experts agreed with the results obtained. Tab. 2 summarizes the number of elements that comprise the different methods defined.

Table 2: Number of elements defined for the different methods involved

Method model	Activities	Dependencies	Artifacts
Ki Method	168	333	116
Ki SCRUM	24	58	21
Ki XP	30	57	26

Contribution 1: At this point, it is important to underline the first contribution of SIAM: the provision of a reference model-based support for the different method definitions and agile practices. Previously, they were in plain text only for the *Ki Method* or in the heads of experts for the *Agile Methods*.

Open Issue 1: An open issue related to the definition of the different models of the knowledge repository is that the experts reported they understood the models defined, but were insufficiently confident to use the modeling tools alone. A process analyst supported the experts in using the SIAM modeling tools. Additional effort to provide more usable and intuitive modeling tools for the knowledge repository specification might be necessary.

Tab. 3 shows the number of equivalencies defined. This table indicates the source and target methods for each equivalency model specified. The equivalencies also considered the quality criteria necessary to maintain the certifications already held by *Ki Process* already has. At this point, the knowledge provided by the ISO and CMMi experts was of paramount importance.

Table 3: Equivalencies Between Methods

Target↓/Source→	Ki Method	Ki SCRUM	Ki XP
Ki Method		18	3
Ki SCRUM	51		15
Ki XP	47	12	

Contribution 2: The activity equivalency models provide common reference artifacts to configure development processes from the combination of the methods defined. Before SIAM, it had not been possible to define these common reference artifacts with the expert knowledge that can be automatically analyzed to configure appropriate development processes. In this way, any change in the process can be automatically analyzed to determine its alignment with the organizational standards and development practices, providing the necessary information to fix the possible gaps identified.

Open Issue 2: Further support for the identification of equivalencies could be provided. The definition of equivalencies is a complex task that demands great effort from experts. We observed that it is possible to infer equivalency candidates from the equivalency models defined, thus reducing the effort required in defining activity equivalency models at the Knowledge Repository Level.

Stage 2: Specification of Ki Process Model.

The second stage is aimed at obtaining a model representation of *Ki Process* that is complete in relation to the original textual definition. This task was performed by a process analyst that did not participate in the Knowledge Repository Definition. This decision was intended to determine whether the information provided in the Knowledge repository was sufficiently complete to guide the proper definition of a development process based on the *Ki Method*.

Since the *Ki Process* is a large process, only a subset of its activities is normally used in a development project. The time required to manually configure the subset of activities involved into a specific development project by using the textual definition is typically around 2-3 weeks. This configuration requires the validation of an expert, in addition to the person responsible for the corresponding process configuration solving any issues in the configuration. With the application of the SIAM approach, the complete *Ki Process* (not a subset) was configured by the process analyst in five days only. Thus, obtaining a process that was fully aligned with the Ki Method specification, without the need for an expert to solve configuration issues. This was possible because all the configuration issues are identified by the SIAM tools, using the activity dependency models and method configuration models from the Knowledge Repository. A summary of the process obtained[1] is presented in Tab. 4, which is automatically generated by the SIAM tools. The initial Ki Process model has an agility ratio of 19%, which comes from 33 of the 167 activities defined, which the experts consider to be agile practices.

Contribution 3: Reduced effort in the Process Configuration. The initial evidence shows that using SIAM requires much less time than the traditional process configuration, reducing this from 2-3 weeks for configuring a subset of the process to only five days for the complete process configuration. This is a promising preliminary result that requires further evaluation to exactly estimate the effort necessary to configure a development process using the SIAM approach.

Analyzing the information in Tab. 4, it can be seen that the row *Nº Activities* shows that 167 of 168 the process activities are referencing the Ki Method. Hence, there is an activity used in development projects that had not been specified in the reference Ki Process document, which is an issue in the original

[1][Blind Review Warning. It contains information about authors] The complete report generated by the SIAM Tool can be downloaded from https://doi.org/10.5281/zenodo.5718107. The original report is in Spanish.

Table 4: Summary of Ki Process Model

Name	Number
Nº Activities:	168 Activities (167 Activities referenced to “Ki Method”)
Nº Artifacts:	369 Artifacts (356 Artifacts referenced to “Ki Method Artifacts”)
Nº Subprocesses:	62 Subprocesses
Nº Transitions:	62 Transitions (21 AND, 41 OR, and 0 XOR)
Nº Method Models:	1 Method Model (1 Referenced Method Model) (Ki Method)
Nº Artifact Models:	1 Artifact Model (Ki Method Artifacts)
Nº Method Configuration Models:	1 Method Configuration Model (Ki Method Configuration)
Nº Equivalency Models:	2 Equivalency Models (KiMethod to KiScrum Equivalency, KiMethod to KiXP Equivalency)
% Agility:	19% of Agility (33 of 167 Referenced Activities are agile)

process specification. Something similar happened for a set of artifacts that are necessary to obtain a sound process definition, but these artifacts are not part of the original specification. These issues were not identified during the 7 years of application of the Ki Process and its different updates during this time; arguably, because it is unusual to find these methodological bugs in a large textual specification. Despite these issues, the original Ki Process obtained the CCMi-DEV (Level 2) and ISO 9001 certification, which means that external auditors also found no inconsistencies in the textual specification.

Contribution 4: Automatic Verification of the Process Definition and its Consistency with the Reference Methods. The results obtained from the Ki Process specification demonstrate the value of SIAM tools to automate the verification of process definition and to check their consistency with the reference methods. In this case, the SIAM tools identified issues that were even imperceptible for method and quality model experts, as well as for process auditors.

Stage 3: Specification of Agile Ki Process from Knowledge Repository Guidelines.

The automatic analysis of the Ki Process model also provides a set of guidelines to increase the ratio of agile practices defined in the process. In the application of the SIAM approach, 56 reconfiguration guidelines were automatically generated for the original Ki Process, *i.e.*, 56 non-agile practices can be replaced by one or more agile practices from the Ki SCRUM Method or the Ki XP Method.

Each time an activity is replaced following the reconfiguration guidelines, the SIAM process modeler tool automatically performs the verification of the dependencies according to the methods involved. This validation implies that it might be necessary to include additional activities to those suggested by the reconfiguration guidelines. Finally, the *Agile Ki Process* is obtained once all the reconfigurations are performed and there are no pending verification issues. The complete reconfiguration of the original *Ki* *process* to obtain the *Agile Ki Process* took three weeks.

Contribution 5: Knowledge Repository continuous improvement. It should be noted that the issues identified in the original Ki Process will not be present for future process configurations, since the solutions to these issues update the Knowledge Repository. In other words, the Knowledge Repository is continuously improved based on the experience obtained from the configuration of processes.

Open Issue 3: The selection of the reconfiguration alternatives is dependent on the process analyst's experience. For the reconfiguration guidelines that provide two or more alternatives, it is the process analyst who decides which alternative to choose according to the project needs. Although the possibilities of process reconfiguration are considerably reduced with the information provided by the guidelines generated, it would be helpful to have additional recommendation mechanisms to evaluate the project requirements and indicate the most suitable alternative among the different possibilities.

Stage 4: Generation of Analysis Report and Expert Verification.

The final stage consisted of automatically generating a complete analysis report (comprising 48 pages) of the KI process[2]. The process summary provided in the report is presented in Tab. 5. This report is valuable for validating the quality of the process, since it provides information about the methods and design decisions involved, the agile practices used, and the inputs and outcomes of the process activities. The *Agile Ki Process* configured by using the SIAM approach obtained the ISO 9001 certification in 2020[3]. Thus, The SIAM approach and tools developed are currently at technology readiness level 7 (TRL-7).

Finally, Tab. 5 shows the ratio of agile activities for the Agile Ki Process was doubled, from 19% to 39%, by using the Ki XP and Ki SCRUM methods. The agility ratio can increase further if new agile methods are referenced in the process reconfiguration or by improving the already defined methods with new agile practices.

Contribution 6: Automatic Analysis of Development Processes. This contribution is related to the enormous amount of information that can be automatically analyzed by the SIAM approach. In the case of the Agile Ki Process, the verification was performed automatically, and the validation by the group of experts took three weeks using the process model and the generated analysis report as inputs.

[2][Blind Review Warning. It contains information about authors] The complete report generated by the SIAM report tool can be downloaded from https://doi.org/10.5281/zenodo.5718107. The original report is in Spanish.

[3][Blind Review Warning. It contains information about authors] The complete model for the Agile Ki Process can be downloaded from https://doi.org/10.5281/zenodo.5267178. The original model is in Spanish

Table 5: Summary of Agile Ki Process Model

Name	Number
Nº Activities:	171 Activities (171 Activities referenced) (135 Referenced to “Ki Method”) (14 Referenced to “Ki XP Method”) (22 Referenced to “Ki Scrum Method”)
Nº Artifacts:	357 Artifacts (336 referenced) (277 Referenced to “Ki Method Artifacts”) (24 Referenced to “Ki XP Artifacts”) (38 Referenced to “Ki Scrum Artifacts”)
Nº Subprocesses:	62 Subprocesses
Nº Transitions:	57 Transitions (18 AND, 39 OR and 0 XOR)
Nº Method Models:	3 Method Models Referenced (Ki Method, Ki XP Method, Ki Scrum Method)
Nº Artifact Models:	3 Artifact Models Referenced (Ki Method Artifacts, Ki XP Artifacts, Ki Scrum Artifacts)
Nº Method Configuration Models:	3 Method Configuration Models (Ki Method Configuraiton, Ki XP Configuration, Ki Scrum Configuration)
Nº Equivalency Models:	6 Equivalency Models Referenced (Ki Process to Ki Scrum Equivalency, Ki Method to Ki XP Equivalency, Ki XP to Ki Method Equivalency, Ki XP to Ki Scrum Equivalency, Ki Scrum to Ki Method Equivalency, Ki Scrum to Ki XP Equivalency)
% Agility:	39% of Agility (68 of 171 Referenced Activities are agile)

Considering that the initial definition of the original Ki Process took two years and that seven years later it still presented a number issues (not identified before SIAM), the effort required is considerably reduced. Even more importantly, the completeness and alignment of the resultant process in relation to the referenced methods and quality criteria of the organization are guaranteed.

Quality Insights

From the industrial application presented, interesting quality indicators were obtained:

1) Reduction of effort in configuring development processes for specific projects. The average time for the manual process configuration was 96 man-hours per project. This involves the project manager responsible for process configuration and two experts that validate the development method application and compliance with standards and quality criteria. Using the SIAM configuration, tool this time has been reduced to less than 24 man-hours per project. Together with the reduction in the time required for the configuring development process, the time required by experts is also greatly reduced. The average time required by experts involved in manually defining and verifying the processes was 40 hours. Using SIAM, this time was reduced to only 8 hours. Since SIAM already validates the consistency with reference methods and development practices, the effort made by the experts is mainly centered on validating the alignment of the process with the project’s requirements and expected results.

2) Improvement of the quality assurance of the development process configuration. SIAM is able to automatically identify process issues and indicate fixing guidelines according to the development practices configured in the knowledge repository. The issues identified and fixing guidelines proposed were validated by the 9 industry experts involved in the industrial evaluation of SIAM. Moreover, the tools was able to identify issues that were detected neither by the different experts nor by external process consultants in the previous certification process. Not only does this indicate that the expert knowledge is properly represented in the SIAM repository and guidelines provided, but also that automating the knowledge application to prevent human-errors, especially in the evaluation of large specifications, is of considerable importance.

3) Facilitating the reconfiguration of traditional development process for the adoption of agile practices. The application to a large industrial development standard shows that the ratio of agile practices involved in the process was doubled, while maintaining compliance with organizational development standards. The Agile Ki Process obtained using the SIAM tools was internally validated by 9 industry experts and externally validated to obtain the ISO re-certification in 2020, thus demonstrating the applicability of the SIAM approach in an industrial context.

4) The configuration guidelines provided by SIAM facilitates the selection and configuration of agile practices for a specific project, which, considering the amount of possible method configurations, can be difficult to remember and evaluate by the project manager manually. Moreover, practitioners declare that guidelines provided are valuable to identify new agile practices and learn and how they must be tailored in a process configuration. Thus, SIAM contributes to the transference and application of expert knowledge for mastering agile practices, specially for novel practitioners.

5) In terms of usability, the project managers reported that, being based on the BPMN standard, the process definition tool is extremely simple to use. Moreover, this facilitated the communication with methodological experts and quality assurance teams in verifying not only the process definition, but also in validating the achievement of the different project results with the development team. However, there are certain usability issues to be considered, for example, the mapping from the development process to the practices defined in the repository is a functionality that could be improved. Recommendations proposed include suggesting mapping alternatives for the process tasks defined or being able to use a pre-defined process as a starting point. Improvements in the usability of SIAM's tools are part of the future work discussed in the next section.

6) Additional studies are necessary to properly measure the improvement in the degree of agility. Despite SIAM not being focused on improving the degree of agility of the process, practitioners perceived that the development process obtained from the SIAM configuration guidelines is more agile. There was consensus on the notion that increasing the ratio of agile practices in a development process involves a higher level of agility, which corresponds to the ratio of agile practices provided by SIAM. However, it was also indicated that this ratio is not sufficiently precise to properly evaluate and compare the agility gained from the process reconfiguration. Thus, further research is necessary to obtain adequate measures for evaluating the degree of agility.

Conclusions

As we have seen throughout this paper, the amount of work involved in the tailoring of development processes to properly apply agile practices is significant, and more so if we consider the organizations must comply with specific methods and quality standards. Using the SIAM approach, it is possible to overcome the complexity involved, thus reducing the effort and time involved in the configuration of development processes for introducing agile practices. In summary, the main contributions obtained from SIAM are:

- Model-driven support to properly characterize development methods and the tacit or poorly documented knowledge related to the application of these methods to specific projects with a special focus on the application of agile practices.
- The implementation of a knowledge repository to store and reuse the agile expert knowledge in different projects, thus helping novel practitioners master agile practices.
- A modeling approach that supports the tailoring of the development process by combining different methods at the same time as their agility level is improved.
- A set of modeling tools to guide the specification of the development process and automate their verification to ensure the alignment with method definitions and organizational quality criteria.

The SIAM approach aims to preserve the expert knowledge related to agile practice application by means of model-driven support. This permits the implementation of specific tools oriented towards practitioners being able to tap this expert knowledge. Thus, the examples presented and the results obtained from the industrial application of SIAM provide initial evidence of the main goal behind this approach being achieved, namely, to facilitate the adoption of agile practices in organizations. Furthermore, although the SIAM approach has an agile vocation, the main concepts involved can be generalized to both the adoption of traditional and agile development methods.

The industrial application of SIAM also suggests different contributions and potential improvements; of these, facilitating the definition of equivalencies between method activities at the Knowledge Repository Level is particularly important. We are working on adding more intelligence to this task, taking advantage of the models already defined to specify new interrelated methods, and thus improving the scalability of the Knowledge Repository. Additionally, we are working on intelligent wizards that suggest relevant practices according to specific project requirements, as well as having pre-defined processes according to specific application domains. Thus, the usability of the tools for configuring suitable development processes will be improved.

Moreover, we are working on specific agility measures with the companies involved to determine the degree of agility achieved when transitioning from traditional processes. These measures can also be used to generate better recommendations for the set of guidelines that improve the overall agility of the processes tailored.

The end-user modeling tool for SIAM is based on the BPMN standard. However, further research is necessary to improve the notation related to the knowledge repository models. In this context, extensions to existing notations for method engineering process are considered as future work for supporting the SIAM modeling needs, such as the notation proposed for the ISO/IEC 24744:2014 standard.

We are preparing different case studies to provide more details of the features of SIAM, the lessons learned, and limitations that may appear in the industrial application of the approach. As a future contribution, we plan to present the application of SIAM to the alignment of agile methods with quality models such as ISO 9001 or CMMi. In particular, our intention is to provide automated evaluation of processes and the generation of recommendations to solve the gaps for the achievement of specific quality standards.

Moreover, since SIAM allows for the alignment of development processes to specific models that represent methods and agile practices, we are working on new application domains for this model-driven approach. In this regard, we are currently working on adapting the SIAM concepts and tools to the configuration of processes aligned with models of standards for the safety certification of critical systems.

Acknowledgements: The work leading to this paper has received funding from the VALU3S (H2020-ECSEL ref. 876852; MCIN/AEI/10.13039/501100011033 ref. PCI2020-112001; European Union NextGenerationEU/PRTR), iRel40 (H2020-ECSEL ref. 876659; MCIN/AEI/ 10.13039/501100011033 ref. PCI2020-112240; European Union NextGenerationEU/PRTR), SIAM (Chile's CORFO-INNOVA ref. 16COTE-60218), and Treasure (JCCM ref. SBPLY/19/180501/000270; ERDF A way of making Europe) projects, from the Ramon y Cajal Program (MCIN/AEI/10.13039/501100011033 ref. RYC-2017-22836; ESF Investing in your future), and from NISUM Technologies Chile.

References

ÅGERFALK, P., & Fitzgerald, B. (2006). Old petunias in new bowls? *Communications of the ACM*, *49*(10), 27.

Al-Zewairi, M., Biltawi, M., Etaiwi, W., Shaout, A., et al. (2017). Agile software development methodologies: survey of surveys. *Journal of Computer and Communications*, *5*(05), 74.

Atkinson, C., & Kühne, T. (2001). Processes and products in a multi-level metamodeling architecture. *International Journal of Software Engineering and Knowledge Engineering*, *11*(06), 761–783.

Azizyan, G., Magarian, M. K., & Kajko-Matsson, M. (2011). Survey of agile tool usage and needs. In *2011 agile conference* (pp. 29–38).

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring–a systematic literature review. *Journal of Systems and Software*, *110*, 85–100.

Cervone, H. F. (2011). Understanding agile project management methods using scrum. *OCLC Systems & Services: International digital library perspectives*.

Cockburn, A. (2006). *Agile software development: the cooperative game*. Pearson Education.

de la Vara, J. L., Marín, B., Ayora, C., & Giachetti, G. (2020). An empirical evaluation of the use of models to improve the understanding of safety compliance needs. *Information and Software Technology*, *126*, 106351.

Digital.ai. (2021). *15th annual state of agile report.* `https://stateofagile.com/`. (Online; accessed 23 Dec 2021)

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review.

Information and software technology, *50*(9-10), 833–859.

EclipseFoundation. (2021). *Eclipse modeling tools.* `https://www.eclipse.org/downloads/packages/release/20`
(Online; accessed 23 Dec 2021)

Elnagar, S., Weistroffer, H., & Thomas, M. (2018). Agile requirement engineering maturity framework for industry 4.0. In *European, mediterranean, and middle eastern conference on information systems* (pp. 405–418).

Fonseca, C. M., Almeida, J. P. A., Guizzardi, G., & Carvalho, V. A. (2021). Multi-level conceptual modeling: Theory, language and application. *Data & Knowledge Engineering*, *134*, 101894.

Fontana, R. M., Albuquerque, R., Luz, R., Moises, A. C., Malucelli, A., & Reinehr, S. (2018). Maturity models for agile software development: what are they? In *European conference on software process improvement* (pp. 3–14).

Fontana, R. M., Meyer Jr, V., Reinehr, S., & Malucelli, A. (2015). Progressive outcomes: A framework for maturing in agile software development. *Journal of Systems and Software*, *102*, 88–108.

Fowler, M. (2001). The new methodology. *Wuhan University Journal of Natural Sciences*, *6*(1), 12–24.

Franch, X., Ralyté, J., Perini, A., Abelló, A., Ameller, D., Gorroñogoitia, J., ... others (2018). A situational approach for the definition and tailoring of a data-driven software evolution method. In *International conference on advanced information systems engineering* (pp. 603–618).

Frank, U. (2011). Some guidelines for the conception of domain-specific modelling languages. *Enterprise modelling and information systems architectures (EMISA 2011).*

Frank, U. (2019). Specification and management of methods-a case for multi-level modelling. In *Enterprise, business-process and information systems modeling* (pp. 311–325). Springer.

García-Borgoñon, L., Barcelona, M. A., García-García, J. A., Alba, M., & Escalona, M. J. (2014). Software process modeling languages: A systematic literature review. *Information and Software Technology*, *56*(2), 103–116.

Gonzalez-Perez, C., & Henderson-Sellers, B. (2007). Modelling software development methodologies: A conceptual foundation. *Journal of Systems and Software*, *80*(11), 1778–1796.

Gonzalez-Perez, C., & Henderson-Sellers, B. (2008). A work product pool approach to methodology specification and enactment. *Journal of Systems and Software*, *81*(8), 1288-1305. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0164121207002439` doi: https://doi.org/10.1016/j.jss.2007.10.001

Gonzalez-Perez, C., Henderson-Sellers, B., McBride, T., Low, G. C., & Larrucea, X. (2016). An ontology for iso software engineering standards: 2) proof of concept and application. *Computer Standards & Interfaces*, *48*, 112–123.

Gren, L., Torkar, R., & Feldt, R. (2015). The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. *Journal of Systems and Software*, *107*, 38–49.

Heimicke, J., Dühr, K., Krüger, M., Ng, G.-L., & Albers, A. (2021). A framework for generating agile methods for product development. *Procedia CIRP*, *100*, 786–791.

Henderson-Sellers, B. (2006). Method engineering: Theory and practice. In *Information systems technology and its applications, 5th international conference ista 2006.*

Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. J., & Rossi, M. (2014). *Situational method engineering*. Springer.

Henriques, V., & Tanner, M. (2017). A systematic literature review of agile and maturity model research. *Interdisciplinary Journal of Information, Knowledge, and Management*, *12*, 53–73.

International Organization for Standardization (ISO). (2000). *Iso 9001:2000 quality management systems-requirements.*

Jossic, A., Del Fabro, M. D., Lerat, J.-P., Bézivin, J., & Jouault, F. (2007). Model integration with model weaving: a case study in system architecture. In *2007 international conference on systems engineering and modeling* (pp. 79–84).

Khalil, C., & Khalil, S. (2020). Exploring knowledge management in agile software development organizations. *International Entrepreneurship and Management Journal*, *16*(2), 555–569.

Kiv, S., Heng, S., Kolp, M., & Wautelet, Y. (2018). Agile manifesto and practices selection for tailoring software development: A systematic literature review. In *International conference on product-*

focused software process improvement (pp. 12–30).

Kurapati, N., Manyam, V. S. C., & Petersen, K. (2012). Agile software development practice adoption survey. In *International conference on agile software development* (pp. 16–30).

Liu, S. (2010). An approach to applying sofl for agile process and its application in developing a test support tool. *Innovations in Systems and Software Engineering*, *6*(1), 137–143.

Łukasiewicz, K., & Miler, J. (2012). Improving agility and discipline of software development with the scrum and cmmi. *IET software*, *6*(5), 416–422.

Lycett, M., Macredie, R. D., Patel, C., & Paul, R. J. (2003). Migrating agile methods to standardized development practice. *Computer*, *36*(6), 79–85.

Maciel, C. P., de Souza, É. F., de Almeia Falbo, R., Felizardo, K. R., & Vijaykumar, N. L. (2018). Knowledge management diagnostics in software development organizations: a systematic literature review. In *Proceedings of the 17th brazilian symposium on software quality* (pp. 141–150).

Mahanti, A. (2006). Challenges in enterprise adoption of agile methods-a survey. *Journal of Computing and Information technology*, *14*(3), 197–206.

Manesh, M. F., Pellegrini, M. M., Marzi, G., & Dabic, M. (2020). Knowledge management in the fourth industrial revolution: Mapping the literature and scoping future avenues. *IEEE Transactions on Engineering Management*, *68*(1), 289–300.

Mutafelija, B., & Stromberg, H. (2003). *Systematic process improvement using iso 9001: 2000 and cmmi*. Artech House.

OMG. (2008). *Software & systems process engineering meta-model (spem) specification.* (Version 2.0.)

OMG. (2011). *Business process model and notation (bpmn) specification.* (Version 2.0.)

OMG. (2017). *Unified modeling language (uml) specification.* (Version 2.5.1)

Paulk, M. C. (2001). Extreme programming from a cmm perspective. *IEEE software*, *18*(6), 19–26.

Qumer, A., & Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of systems and software*, *81*(11), 1899–1919.

Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A study of the agile software development methods, applicability and implications in industry. *International Journal of Software Engineering and its applications*, *5*(2), 35–45.

Ratiu, D., Nordmann, A., Munk, P., Carlan, C., & Voelter, M. (2021). Fasten: An extensible platform to experiment with rigorous modeling of safety-critical systems. In *Domain-specific languages in practice* (pp. 131–164). Springer.

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

Ruy, F. B., de Almeida Falbo, R., Barcellos, M. P., & Guizzardi, G. (2014). An ontological analysis of the iso/iec 24744 metamodel. In *Fois* (pp. 330–343).

Sandkuhl, K., & Seigerroth, U. (2019). Method engineering in information systems analysis and design: a balanced scorecard approach for method improvement. *Software & Systems Modeling*, *18*(3), 1833–1857.

Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum* (Vol. 1). Prentice Hall Upper Saddle River.

SEI. (2006). *Cmmi for development (cmmi-dev).* (version 1.2)

Singh, A., Singh, K., & Sharma, N. (2012). Managing knowledge in agile software development. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *2*(4).

Sousa, K., Vanderdonckt, J., Henderson-Sellers, B., & Gonzalez-Perez, C. (2012). Evaluating a graphical notation for modelling software development methodologies. *Journal of Visual Languages & Computing*, *23*(4), 195–212.

Sreenivasan, S., & Kothandaraman, K. (2019). Improving processes by aligning capability maturity model integration and the scaled agile framework®. *Global Business and Organizational Excellence*, *38*(6), 42–51.

Tripp, J. F., & Armstrong, D. J. (2014). Exploring the relationship between organizational adoption motives and the tailoring of agile methods. In *2014 47th hawaii international conference on system sciences* (pp. 4799–4806).

Valdés, G., Visconti, M., & Astudillo, H. (2011). The tutelkan reference process: A reusable process model for enabling spi in small settings. In *European conference on software process improvement* (pp. 179–190).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Yin, A., Figueiredo, S., & da Silva, M. M. (2011). Scrum maturity model. *Proceedings of the ICSEA*, 20–29.