Experiences from the Design of an Artifact Model for Distributed Agile Project Management

Henning Femmer, Marco Kuhrmann
Technische Universität München
Garching, Germany
{femmer,kuhrmann}@in.tum.de

Jörg Stimmer, Jörg Junge
pliXos GmbH
81247 Munich, Germany
{joerg.stimmer,joerg.junge}@plixos.com

***Abstract*—The organization of projects with distributed teams is a demanding task for every project manager. Requirements need to be collected, documented, and discussed, and the resulting tasks must be distributed to the responsible sites. These activities require an efficient and continuous communication. Furthermore, it is necessary to monitor a project and to track its progress from a management perspective. As a solution, we opt for a monitoring strategy that is based on the project artifacts and corresponding reports. For this, we defined in a previous work a generic artifact model for agile methods to enable seamless communication and data exchange between projects and teams. In this paper, we present a concrete instance aiming at providing the backbone of the information and data exchange subsystem of a SaaS-based collaborative project management and governance software for distributed software development. We present the artifact model, give insights into its development, and discuss its feasibility. Our findings show that while the previously defined reference model adequately reflects basic concepts and thus allows for coupling distributed projects, we need to refine the artifact model to emphasize project management/governance and its implementation in tools.**

***Index Terms*—agile methods; artifact model; governance; distributed projects; project management; experience report**

I. Introduction

Managing a software project is a demanding task, as many activities need to be organized, e.g., requirements elicitation, software development, test, or solution delivery. Distributed software development pushes project management to the next level of difficulty. Project management is performed in a setting in which teams are spread across the globe. Hence, project managers have to deal with time zones, asynchronous communication, language and culture barriers [1], heterogeneous software processes, or delayed/inconsistent reports not adequately reflecting the current project progress. In order to successfully manage distributed settings, project managers require sophisticated tool support that allows for a "near real time" project monitoring and tracking, and that allows the distributed team to share information and artifacts.

A centralized solution in which a shared team server provides the team with the necessary infrastructure seems to be the "promised land", as infrastructure (e.g., repository, test, or build server) as well as collaboration options (e.g., built-in real-time communication and collaboration support) [2]–[4] are provided. Consequently, the question for the data to be exchanged and its structure arises.

Problem Statement & Research Objective. In [5], we proposed a generic artifact model to support data exchange among distributed project sites. The proposed model addressed the need to standardize data structures to abstract from local processes. However, the generic artifact model emerges from a systematic literature review and, thus, only reflects state-of-the-art as reported in literature. In the paper at hand, we discuss the feasibility of the proposed model in practice by investigating the following research questions:

RQ	Question and Rationale
RQ1	*Is the generic artifact model sufficiently complete for practical application?* When applying the generic artifact model in practice, we need to analyze, customize, and/or revise the model in order to address specific (technical) needs. Therefore, we investigate how well the generic model reflects real world requirements, especially, in terms of completeness of the model.
RQ2	*Is the generic artifact model sufficiently precise for practical application?* One purpose of an artifact model is to provide a precise description of a data model. Due to the nature of the generic artifact model, we expect refinements, e.g., of attributes. Thus, we investigate the precision of the generic model compared to the refined model.
RQ3	*Is the refined model still compatible with the generic artifact model?* Based on the generic model, we defined a process interface for data exchange. We investigate whether the refined model is compatible with the generic model to evaluate the feasibility of the generic model for establishing a common project and process interface.

Contribution. In this paper, we contribute an artifact model to support the construction of tools for managing distributed projects. For this, we use a previously defined reference artifact model for agile methods and enhance it for the use as a real-world data exchange model for a SaaS-based collaborative project management tool. We provide an analysis to compare the models and discuss our lessons learnt during the collaboration with practitioners. The rest of the paper is organized as follows: In Sect. II, we discuss related work and previously published material. In Sect. III, we introduce the improved artifact model, analyze it in Sect. IV, present our lessons learnt in Sect. V, before summarizing the paper in Sect. VI.

II. Background & Related Work

Agile methods gained much attention over the years and, beyond small-scale and co-located development, are also subject for research in distributed software development, e.g., [6]–[8]. Hersleb and Mockus [9] investigated collaboration in

distributed projects. As part of collaboration, direct communication is a basic pillar of agile methods. However, direct communication is hard to realize in distributed settings and, thus, often relies on tools [**?**], [10]. In [11], we found artifacts as means to direct communication between projects beneficial. However, regarding agile processes, we also found significant gaps of artifact descriptions. In response, we conducted a systematic literature review to improve the understanding of artifacts in agile methods, and proposed an artifact-based process interface to couple distributed projects [5].

A Generic Artifact Model for Agile Methods. The motivation behind the general artifact model is that in the context of distributed development the data (structures) and information of the local processes usually differ. However, although processes may differ, all information that is exchanged needs some degree of standardization. The basis for defining an artifact model to support distributed project management is a generic artifact model that provides a general notion of agile methods, and how agile methods can be systematized [5]. The artifact model emerges from a comprehensive literature study in which the reported state of the art on the use of artifacts in agile methods was investigated. The complete model ist shown in [5]; in the following, we describe the most important parts.

The heart of the artifact model is the class *Artifact*, representing any piece of information created in a project (realized using the composite pattern, cf. [**?**]). Artifacts have dependencies among each other to model relationships to enable tailoring for creating project-specific artifact models. Also, artifacts have roles assigned that are either responsible for the artifacts' creation, or that contribute to an artifact's creation or modification. All elements of the model are subclasses of *Artifact*, e.g., *BacklogItem*, *Code*, or *Deliverable*. In order to directly support the description of a project-specific approach, the artifact model comprises several refined artifact types addressing typical project disciplines, e.g., project-, task- and time management: *TeamMember*, *ProjectBacklogItem*, or *Task*; development: *SourceCode*, *Ressource*, or *Release*; quality assurance: *TestCase* or *UnitTest*; or requirements engineering: *Feature*, *UserStory*, or *UseCase*. However, we designed the artifact model in a generic fashion that requires a refinement (extension, deletion, modification of classes and attributes) respecting the actual organization and project context.

III. An Artifact Model for Agile PM

In the following sections, we present a context-specific refinement of the previously defined artifact model [5] to support the development of a tool for distributed agile project management and governance. In order to transfer the ideas and concepts into practice, we refined the artifact model and adopted it for being used as a backbone for a real-world application. In this section, we describe the refinement approach, before presenting the refined artifact model.

A. Refinement Approach

The refinement was conducted in a cooperation project in which *pliXos GmbH* and TU München investigated solutions to support distributed projects and their management. The goal was to develop an actionable artifact model for a tool under development at pliXos, called the *Outsourcing Director*. For this, we refined the existing model through the following steps: 1) Suitability analysis of the general (*reference*) artifact model. 2) Analysis of the *Outsourcing Director* use cases: specifically, we looked at (implicit/explicit) artifact models of possible collaboration systems, e.g., Microsoft's Team Foundation Server (TFS) [12]. 3) Analysis and design workshops to enhance the reference model and to develop the target model. 4) Feedback (industry and potential customers for the tool) and refinement cycles.

In summary, the refined artifact model was inductively developed. We refined the model in several workshops taking into account the roadmap of the tool development, respective use cases, and especially comprehensive feedback from clients.

B. Refined Artifact Model

The refined artifact model serves the development of a data format to allow for collaboration and data exchange between project sites from the perspective of project management. We describe the model by its packages, and go into details for selected refinements. However, the focus of this paper is not to explain the entire model but to discuss its refinement and applicability and explain our lessons learnt.

1) Resulting Model: Figure 1[1] shows the refined artifact model resulting from the aforementioned refinement steps. Around the key class *Artifact*, we refined elements from the generic artifact model and introduced new elements, which we organized in packages. The package *Artifacts* comprises the artifacts reflecting the project disciplines *Planning*, *Requirements & Specification*, *Change Management*, *Development*, and *Testing*. In these packages, we locate all artifact types that are subject to controlling and reporting, e.g., *Task*, *User Story*, or *Issue*.

To allow for creating a distributed project structure, we added the packages *GloBuS Bus* and *Service Provider Project Configuration*. In the first package, we locate those artifact types necessary to build the common structure between parties. In the GloBuS context, a project has a *Client*, and a project is carried out by at least one *ServiceProvider*. Furthermore, in a project, a number of *GloBuSReleases* (deliverables) are shipped to the client. While the *GloBuS Bus* package represents the overall project, the provider configuration package represents the structure of a sub-project conducted by a service provider. The root element is the class *Project*, which relates the project team (class *TeamMember*) with artifacts being produced by the team. Furthermore, a project produces several deliverables, which are (collections of) artifacts themselves, and that become part of an integrated *GloBuSRelease*. That is, according to the artifact-based design approach, the provider configuration package serves as a proxy for the fine-grained artifact model. All considered artifact types are children of the

[1]This is a simplified version without attributes and relations. The full version can be found here: http://www4.in.tum.de/~kuhrmann/sonst/globus.zip

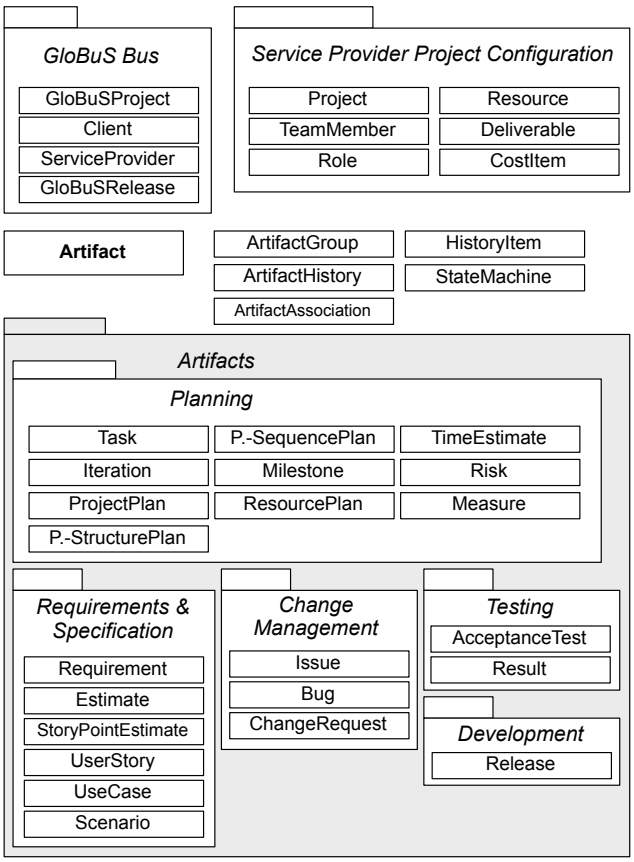

Fig. 1. The refined artifact model for the GloBuS project (simplified).

class *Artifact* and thus allow for polymorphic data structures to be exchanged among (sub-)projects. To support the management of a distributed project, the refined model comprises all artifact types considered necessary to organize and distribute work, and to create reporting lines to track project progress.

2) Examples for Refinements: To use the reference artifact model [5] in practice, we had to refine it. The following two examples provide some insights into the construction procedure.

A major refinement considered essential was the notion of finances. To enable project managers to control a project, effort needs to be estimated, and effort needs to be converted into costs to allow, e.g., for creating offers, or bidding on a publicly announced project. For this, the attribute *estimate* (original model *BacklogItem*) was extracted and explicitly modeled as a class *Estimate* for which the refinements *TimeEstimate* and *StoryPointEstimate* were introduced. Furthermore, a class *CostItem* was defined and linked to resources (internal costs) and roles (external costs). Every Task has resources and several estimates assigned, and, thus, allows for estimating effort, time, or costs. Including internal and external costs allows for internally calculating costs for personnel or consuming resources, and, at the same time, for billing the clients.

As second example for a substantial refinement, we discuss the enriched attribute lists, e.g., for the artifact type *Issue*. The initial study, on which the reference model is based, revealed no fine-grained attribute structure. To make *Issue* artifacts tangible, we relied on previous research in the context of process enactment and adopted work item structures from Microsoft's TFS process templates for the intended artifact model. This approach was then also applied to the remaining artifact types to provide an initial but proven set of attributes.

IV. Analyzing the Artifact Model

In this section, we analyze the refined artifact model to answer the research questions. For this, we briefly show the study design in Sect. IV-A and present the results in Sect. IV-B.

A. Study Design

The overall goal is to provide a refined artifact model aiding the development of project management tools. For this, the general artifact model for agile methods was taken as reference with the purpose to analyze and compare the refined model in a practical context. To investigate the objective, we defined three research questions (see Table I): *RQ1* focuses on the amount of refinement necessary to get from the reference model to the applicable refined model on a coarse-grained level. *RQ2* analyzes the details that we needed to specify to achieve applicability, and *RQ3* analyzes whether the reference model still sufficiently serves as the least common denominator for different artifact models.

To answer the research questions, we conducted a comparative analysis of the reference model and the refined artifact model. In the following, we describe in more detail how we addressed the research questions: For *RQ1*, we walked through each class in both the reference as well as the refined model, and decided which element exists in either both or only a single artifact model. For the discussion, we afterwards qualitatively analyzed intersections as well as gaps between the models, by walking through the elements and reflecting on the rationales behind their presence or absence. For *RQ2*, we looked at the attributes and references of the model and marked those classes that were refined during the process. Again, the results were qualitatively interpreted for the discussion. For *RQ3*, we specifically selected the classes of the reference model that were considered part of the proposed interface [5], which shall define the least common denominator regarding data structures to be exchanged among project sites, e.g., artifacts or planning information. For each class defined in the proposed interface, we decided whether the class was present in the refined model, whether a subclass or renamed class of the original class was present in the refined model, or whether the class was not part of the refined model at all. Afterwards, reasons were again qualitatively analyzed.

Author 1 and Author 2 performed the data collection for these tasks in critical discussion by element-wise comparing artifact model printouts and storing the results in a spreadsheet[2]. The results were validated together with Author 3 and Author 4.

[2]Full dataset: http://www4.in.tum.de/~kuhrmann/sonst/globus.zip

B. Results

The presentation of results is structured according to the research questions.

RQ1: Figure 2 shows the relation between the two models. Apparently, there is a gap between the reference model and the refined model: 28 of the 41 classes (ca. 68%) are not included in the refined model and 25 of the 38 classes (ca. 66%) of the refined model are not present in the reference model. This leads to 13 classes that are present in both artifact models.

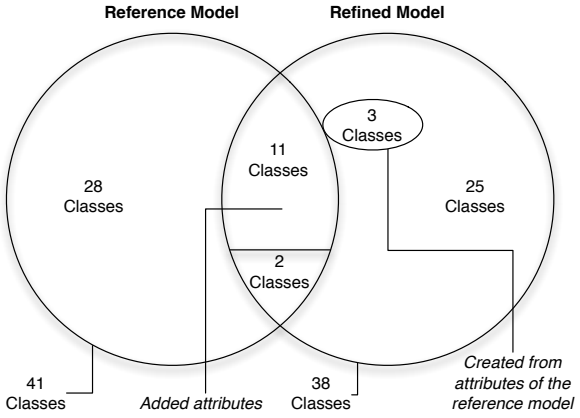

Fig. 2. Comparison of the reference and refined artifact model

Interpretation—When looking into the details, large parts of the original model were removed, whereas other parts, e.g., the *GloBuS Bus*, are completely new (Sect. III-B). We were expecting that some parts must be added because of the context. However, we did not expect the large number of classes that did not appear in the final model. For instance, the packages *GloBuS Bus* and *Service Provider Project Configuration* were completely new. The reason: the generic artifact model does not provide any means to model project configurations, which is, however, necessary when developing project management support. On the other hand, the code-related artifacts from the reference model were removed, as particular code artifacts, e.g., classes or resource file, were considered irrelevant in the actual context. From this, we argue that the reference model, which reflects the reported state of the art of the use of artifacts in agile methods, is yet incomplete from the perspective of project management, as it does not address the overall project organization.

RQ2: As Figure 2 shows, 13 classes are present in both the reference as well as the refined artifact model. Of these 13 classes, 11 classes needed further refinement of the attributes (in summary: 45 attributes were refined and/or added to these classes). Extracting an attribute of a class and creating a separate class with further attributes lead to 3 more refinements. For instance, in the generic artifact model, in the class *Artifact*, information regarding artifact status and history were comprised as attributes. In dialog with industry, it was suggested to make this (essential) information more explicit, e.g., to better support history analyses, or flexible state machine configuration.

Interpretation—This analysis mirrored our perception of the reference model. Even though on a coarse-grained level we felt that the general information is present, we expected that all reused classes must be refined to fit to our concrete purpose and setting. From this, we argue that the general direction of the generic reference artifact model is correct, however, that it needs refinement to be used in practice.

RQ3: The artifact-based process interface as part of the reference model [5] consists of 10 classes. In this research question, we analyzed in how far the refined model can serve to the interface. Figure 3 shows the three different levels of support: directly supported classes are marked with green color, indirectly supported classes were branded with a yellow color (their proxy classes in light-green hatched), and non-supported classes were marked red.

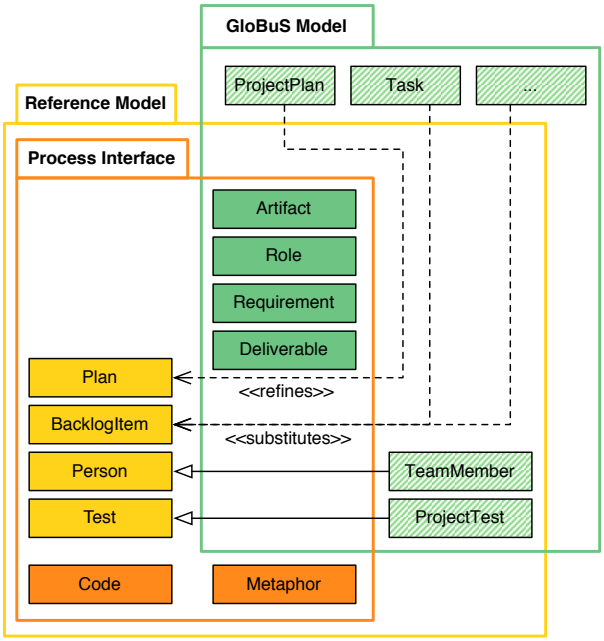

Fig. 3. Supported classes of the interface of the reference model

The figure shows that four classes of the interface (*Artifact*, *Role*, *Requirement* and *Deliverable*) are also present in the refined model and are thus directly supported. Four further classes (*Plan*, *BacklogItem*, *Person*, and *Test*) are not directly part of the refined model; however, classes that are defined as subclasses in the reference model or contain at least the information of the interface (*refines, substitutes* in the figure) were present in the refined model (indirectly supported). Thus, we can say that the information required for the interface is also present in the refined model. For example, even though we did not model a *Person* as class in the refined model, we find the subtype *TeamMember* that contains at least all the information of the class *Person*. Two classes (*Code*, *Metaphor*) are present in the interface, but not in the refined model.

Interpretation—The directly supported artifacts are central to any project management approach, and also, in contrary to the indirectly supported classes, have a very well-defined meaning and purpose. The indirectly supported classes are

mostly very abstract (e.g., *Person*, *Plan*, or *Test*) and thus need more specific details in order to be of value. Lastly, we have to investigate the non-supported classes: First, code was not in the focus of the GloBuS endeavor and, thus, it did not surprise us that there was a gap between the reference and the refined artifact model. For the *Metaphor*, experience has always told us that this is a concept that has been discussed widely in the theory of the XP community, but did not make it into practice. Or, as [13] puts it: "The 'metaphor' is the practice of agile processes most ignored by practitioners". This study once again supports that claim.

V. Lessons Learnt

In our study, we discussed the feasibility of a previous reference artifact model for practical use. In detail, we analyzed how we need to transform the reference model to satisfy the requirements of its deployment in practice. Comparing both models—the reference model and the refined model—we saw that the reference model needed extensive refinement. We found that especially the details regarding the attributes needed substantial revision. However, although several adjustments were necessary to enable the model for practical use, the resulting refined model is still "compatible" with the reference model, and, thus, the intention to define a generic interface to exchange project data is still met with the refined model.

Moreover, our study revealed further interesting insights: the study indicates—again—that the concept of a *metaphor* is widely discussed in theory but apparently not relevant in practice (the respective artifact was removed from the model). Another striking and interesting insight of the results are the predominant concepts of traditional project management, which shape the once very agile artifact model. After extensive discussion with practitioners, we conclude that this is due to the fact that if an agile process is transferred into a global scale, more and more information for the business context is needed—here the practice often relies on the long experience of traditional project management techniques. The experts using these techniques told us that that traditional key-performance indicators are still necessary for controlling regardless of the process used. Moreover, during the design workshops we realized that the more "formalization" (to support tool development) is necessary, the less agility remains in the model. From this, we argue that we still miss adequate techniques to precisely model agile projects today.

We deliberately started to create an artifact model for a particular context in order to extend this model iteratively driven by demand. However, we consider the presented artifact model a first step towards a deeper understanding of structures in agile projects, and for exchanging project information. We are very interested in the feedback of future users of the artifact model, and we cordially invite researchers and practitioners to share their experiences.

VI. Conclusion

In this paper, we described the refinement of an existing reference artifact model for its application in distributed agile project management. We furthermore comparatively analyzed the difference between the generic reference model and the concrete, practically usable refinement for the GloBuS project.

Our study shows that large parts of the artifact model must be refined, either by creating new classes or by adding attributes to existing classes. However, the study also shows that even though refinements are necessary, the original model provides enough material to start with. The original intention (provide common ground) is still valid, and the study shows a compatibility of both models. Our study also raises questions on the dissemination of the concept of a *metaphor* in practice.

Particularly interesting is the result that the artifact model that we created and that started out as an *agile* artifact model had to be extended more and more with *traditional* project management artifacts. This raises a question that must be analyzed in detail in future investigations: *Is it true that even for agile project management, the more we think on a global scale, the more the artifact model resembles models of old-school project management?*

Acknowledgments

The project *GloBuS* was supported by the Bavarian State Ministry for Economic Affairs, Infrastructure, Transport and Technology (StMWIVT) under IUK-1110-0002//IUK391/001.

References

[1] D. Šmite, N. B. Moe, and R. Torkar, "Pitfalls in Remote Team Coordination: Lessons Learned from a Case Study," in *Product-Focused Software Proces (PROFES)*. Springer, 2008.

[2] P. Tell and M. Babar, "Activity Theory applied to Global Software Engineering: Theoretical Foundations and Implications for Tool Builders," in *International Conference on Global Software Engineering (ICGSE)*, 2012, pp. 21–30.

[3] F. Lanubile, T. Mallardo, and F. Calefato, "Tool support for geographically dispersed inspection teams," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 217–231, 2003.

[4] F. Calefato and F. Lanubile, "Socialcde: A social awareness tool for global software teams," in *Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2013, pp. 587–590.

[5] M. Kuhrmann, D. M. Fernández, and M. Gröber, "Towards artifact models as process interfaces in distributed software projects," in *International Conference on Global Software Engineering (ICGSE)*, 2013.

[6] H. Holz and F. Maurer, "Knowledge Management Support for Distributed Agile Software Processes," in *Advances in Learning Software Organizations*. Springer-Verlag, 2003.

[7] M. Paasivaara and C. Lassenius, "Collaboration practices in global inter-organizational software development projects," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 183–199, 2004.

[8] H.-C. Estler, M. Nordio, C. Furia, B. Meyer, and J. Schneider, "Agile vs. structured distributed software development: A case study," in *ICGSE*, 2012.

[9] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, 2003.

[10] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin, "Communication tools for distributed software development teams," in *the 2007 ACM SIGMIS CPR conference*. ACM Press, 2007, pp. 28–35.

[11] M. Kuhrmann, C. Lange, and A. Schnackenburg, "A survey on the application of the v-modell xt in german government agencies," in *Proceedings of the 18th Conference on European System & Software Process Improvement and Innovation (EuroSPI)*, 2011.

[12] M. Kuhrmann and G. Kalus, "Werkzeugspezifisches Tailoring für das V-Modell XT," Technische Universität München, Research Report (in German) TUM-I0804, 2008.

[13] J. Tomayko and J. Herbsleb, "How Useful Is the Metaphor Component of Agile Methods? A Preliminary Study," CMU, Tech. Rep., 2003.