Using boundary objects and methodological island (BOMI) modeling in large-scale agile systems development

Citation for the original published paper (version of record):
Holtmann, J., Horkoff, J., Wohlrab, R. et al (2025). Using boundary objects and methodological island (BOMI) modeling in large-scale agile systems development. Software and Systems Modeling, 24(1): 183-207.
http://dx.doi.org/10.1007/s10270-024-01193-x

Software and Systems Modeling (2025) 24:183–207
https://doi.org/10.1007/s10270-024-01193-x

REGULAR PAPER

Using boundary objects and methodological island (BOMI) modeling in large-scale agile systems development

Jörg Holtmann[1] · Jennifer Horkoff[1] · Rebekka Wohlrab[1] · Victoria Vu[2] · Rashidah Kasauli[3] · Salome Maro[4] · Jan-Philipp Steghöfer[5] · Eric Knauss[1]

Received: 5 May 2023 / Revised: 13 November 2023 / Accepted: 21 May 2024 / Published online: 5 August 2024

Abstract

Large-scale systems development commonly faces the challenge of managing relevant knowledge between different organizational groups, particularly in increasingly agile contexts. Here, there is a conflict between coordination and group autonomy, and it is challenging to determine what necessary coordination information must be shared by what teams or groups, and what can be left to local team management. We introduce a way to manage this complexity using a modeling framework based on two core concepts: methodological islands (i.e., groups using different development methods than the surrounding organization) and boundary objects (i.e., artifacts that create a common understanding across team borders). However, we found that companies often lack a systematic way of assessing coordination issues and the use of boundary objects between methodological islands. As part of an iterative design science study, we have addressed this gap by producing a modeling framework (BOMI: Boundary Objects and Methodological Islands) to better capture and analyze coordination and knowledge management in practice. This framework includes a metamodel, as well as a list of bad smells over this metamodel that can be leveraged to detect inter-team coordination issues. The framework also includes a methodology to suggest concrete modeling steps and broader guidelines to help apply the approach successfully in practice. We have developed Eclipse-based tool support for the BOMI method, allowing for both graphical and textual model creation, and including an implementation of views over BOMI instance models in order to manage model complexity. We have evaluated these artifacts iteratively together with five large-scale companies developing complex systems. In this work, we describe the BOMI framework and its iterative evaluation in several real cases, reporting on lessons learned and identifying future work. We have produced a matured and stable modeling framework which facilitates understanding and reflection over complex organizational configurations, communication, governance, and coordination of knowledge artifacts in large-scale agile system development.

Keywords Boundary objects · Agile development · Empirical studies

1 Introduction

Large-scale systems engineering companies commonly face the challenge of coordination between multiple and multi-disciplinary teams that focus on different aspects of a system (e.g., software, systems, hardware). Especially in large-scale agile development, inter-team coordination is a recognized challenge [9]. In practice, ways of working are not universal in large companies. Teams are surrounded by other organizational parts that do not use the same methods—and thus become "*methodological islands*" [25]. For instance, in a large automotive company that we collaborated with, more than 500 teams exist, using diverse practices (agile, waterfall), with complex interdependencies and multiple suppliers. Coordination is supported by various artifacts (e.g., written documents, models, backlogs, or code). Furthermore, phone calls, meetings in communities of practice, and other mechanisms are used to communicate around these artifacts. In such a situation, it can be challenging to coordinate

Communicated by Holger Giese.

✉ Jörg Holtmann
jorg.holtmann@gu.se

1 Chalmers | University of Gothenburg, Gothenburg, Sweden

2 Semcon AB, Gothenburg, Sweden

3 Makerere University, Kampala, Uganda

4 University of Dar es Salaam, Dar es Salaam, Tanzania

5 XITASO GmbH IT & Software Solutions, Augsburg, Germany

knowledge between different organizational groupings. Practitioners need to better understand the factors causing these groups (or islands) to cluster or form and the effectiveness of the current ways of supporting communication. For example, is a particular written document between two islands fit for coordination? Is it too flexible or too rigid? Is it both complex and changing frequently? Is it governed, and do those that govern the document understand its use? Can the current coordination situation be understood, made explicit, and improved?

As part of a long-term line of research, we aim to characterize these coordination needs by focusing on methodological islands (MIs) and boundary objects (BOs) [25]. Boundary objects are artifacts that create a common understanding between groups and can facilitate inter-team coordination and knowledge management [58]. Initially, we investigated the nature and use of these BOs in practice using informal drawings and notes, but found that we need a more systematic approach to guide us and our company partners in capturing BOs and MIs and their interactions. More specifically, we felt we needed an artifact which captured the BOMI landscape graphically, as well as supported discussion, promoted shared understanding, and facilitated improving ways of working. These needs led us to turn to conceptual modeling to capture and work with BOMI concepts. To the best of our knowledge, there is no modeling approach and conceptual model available to specifically address boundary objects and methodological islands.

In a previous paper [54], we have addressed this gap by proposing a high-level modeling method for boundary objects and methodological islands in large-scale systems development, including a metamodel and guiding method. This method was created using a design science approach based on empirical data and accounts from ongoing projects [25, 53, 55]. By using such a method, a complete picture of an organization's coordination needs and boundary objects can be established, analyzed, and used to identify and mitigate current issues in a more visual and structured way.

In this previous work, as part of an early design cycle, we evaluated the method in a short workshop together with four large-scale systems companies and described the corresponding instance models created. We presented initial findings on how the model can be used to identify bad smells and issues.

In this paper, we continue the design science process with additional cycles. In the second iteration, we apply the BOMI framework to three further cases in three companies, one of which did not participate in our earlier iterations, bringing our total number of collaborating companies to five. We further apply BOMI to an internal case. The length of time dedicated to each modeling workshop in some of the additional iterations has increased (from 1.5 to 3 h per case). With each case, we reflect on findings and make any necessary adjustments to the metamodel, method, and list of smells (our artifact). We used our experiences to create guidelines for applying BOMI, and better understand and describe expectations and benefits of applying the model and method.

Our experiences in applying BOMI led to a fourth and final additional iteration, in which we developed dedicated tool support, allowing modelers to create models graphically or via a textual syntax. The tool includes pre-defined views based on potential user needs to manage BOMI model complexity.

To summarize, this work makes the following contributions:

- A metamodel of BOMI concepts (initial version described in [54]).
- A method guiding the creation of BOMI instance models (initial version described in [54]).
- A set of smells over the metamodel which may indicate coordination issues (initial version described in [54]).
- Initial evaluation of the BOMI framework in a short workshop with four companies (described in [54]).
- Experience and empirical feedback on the BOMI framework as part of three, in-depth per-company workshops with three companies.
- An evaluation of BOMI using an internal tool.
- Formalized and stakeholder-driven views over BOMI instance models, including evaluation of proposed views with three companies.
- BOMI framework tool support, including graphical and textual editing, capturing the BOMI metamodel, smells, and automatic view creation.

Due to the extended evaluation, our results include several new, real instance examples to illustrate and evaluate the BOMI framework. Overall, our findings support the ability of the BOMI framework to systematically model and understand coordination, intended usage, and governance of knowledge artifacts in systems engineering. This enables companies to better manage the knowledge needed and used as part of scaled agile approaches.

This paper is organized as follows: Sect. 2 presents the background. Section 3 describes our artifact, the final metamodel, method, smells description, views and tooling. Our design science research methodology, along with the results for each cycle, is described in Sect. 4. Section 5 reviews and compares our work to related modeling approaches. Section 6 discusses our findings, provides modeling guidelines based on our experiences, and describes threats to validity, while Sect. 7 concludes the paper.

2 Background

In this section, we provide a description of the concepts which underlie this work.

2.1 Boundary objects

Boundary objects are a central mechanism to support inter-team coordination. Previous research has stressed the importance of knowledge management and identified that successful self-organizing teams rely on effective knowledge networks that support knowledge sharing across boundaries [34, 49]. Alternative (tacit) knowledge management strategies are focused on communities of practice or guilds [48]. In this line of research, we are concerned with externalized knowledge in large-scale development [60], where the concept of boundary objects is especially useful.

Boundary objects (BOs) originate from sociology and constitute coordination mechanisms in contexts that involve multiple human groups. They are defined as "*objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites*" [44]. The concept of BOs was initially coined in sociology and has proven to be useful in a variety of domains. Recently, BOs have increasingly been studied in software and systems engineering [43, 58, 61]. For example, architecture descriptions or high-level requirements can serve as boundary objects [58].

Our use of the term "boundary object" should not be confused with the older use of the same term in the object-orientation literature, where it refers to objects that translate user inputs into processable information [39] or provide an API to external systems. Although a code object could be a BO in our framework, as we focus on process-oriented coordination, we have found that it is more useful to model at a higher level of abstraction (e.g., a component, an API). However, we decide to keep the "object" terminology from the sociological origins of the concept, due to the past work of us and others who use this term. In this paper, we use Star and Griesemer's definition of boundary objects [44]. Boundary objects do not have to be artifacts in the classical sense, but can also be physical objects, compilations of artifacts, or classification systems.

Over the several years, we have engaged with four large-scale systems engineering companies to support them in adopting agile methods and managing important knowledge. As part of this work, we made use of the BO concept to capture knowledge coordination.

We analyzed currently used artifacts and created guidelines to manage them in large-scale agile contexts, including concerns related to the level of detail and versioning of these artifacts [58]. We found that BOs can belong to several super types (e.g., Technology, Task, or Planning) [25] and should be managed in groups of representatives of several teams [58]. Moreover, we studied architecture descriptions and interfaces as BOs [53, 57]. We found that important dimensions of interface change are stability, time to perform a change, criticality, level of abstraction, distance to affected parties, number of affected components, position in the interface's lifecycle, and maturity of affected functions. We identified that many companies describe *information models* to capture artifact types and their relations. These information models also serve as BOs, change over time, and can be used to define the required degree of alignment of different teams' practices [55].

BOs are commonly used between individuals from several (sub-)disciplines, who refer to concepts with different terminologies [58]. The groups using BOs need to be properly understood to enable inter-team coordination.

Although our previous work used and evaluated the BO concept for large-scale agile coordination, our approach was informal and did not make use of the power of conceptual modeling, including metamodeling, a modeling method, views, smells and tooling.

2.2 Methodological islands

As part of our past line of work, we found the need to introduce the concept of methodological islands. Although the mix of methods in large-scale organizations is a recognized challenge [58], the term is new as per our past work. In our empirical study on large-scale development, agile teams were described as "*agile islands in a waterfall*" [25]. This phenomenon is not limited to the discrepancy of agile and plan-driven methods, but a general issue. Therefore, we use the term *methodological islands* (MIs) for organizational groups using different development methods than the surrounding organization. Teams in agile development are commonly located at different sites, not only geographically, but also with respect to the temporal, cognitive, or psychological distance [4]. We identified that MIs can be of different types, e.g., individual teams (e.g., component teams), groups of teams (e.g., departments), or entire organizations. MIs arise due to several *drivers* related to *business*, *process*, and *technology*.

Based on our previous studies, we got an understanding of BOs and MIs in large-scale systems engineering. These findings needed to be better instrumentalized to support practitioners, in particular, using a systematic approach to capture BOs and MIs [25]. Such an approach would constitute a formal treatment to describe and evaluate coordination needs.

3 The BOMI framework

In the following, we present the results of our design science cycles, the final artifact. This artifact, updated from [54], is the result of the design science cycles and evaluations as described in Sect. 4; however, we feel that having an understanding of the framework will help the reader to understand the methodology applied. This section includes the BOMI

metamodel, instance examples to illustrate the metamodel, a recommended BOMI modeling method, the analysis capabilities provided by the model, and the BOMI modeling tool.

3.1 BOMI metamodel

In this section, we present the BOMI metamodel. To capture our conceptual model, we use a UML class diagram. Other languages could work just as well, but we choose UML due to its familiarity. The latest version of the BOMI metamodel can be found in Fig. 1. Note that this metamodel has a high abstraction level, and we further detail and formalize it in a tool-processable way as part of our dedicated tool support for the BOMI framework in Sect. 3.6.1.

Based on our past findings, the most critical element of the metamodel is the BoundaryObject itself (in dark gray). We give this class a multi-valued attribute *superTypes*, based on our past classification findings [25]. This attribute is typed by the enumeration *BOSuperType* defining a corresponding list of options.

We use our findings to identify a number of internal BoundaryObject attributes, including the *purpose*, *levelOfDetail*, *frequencyOfChange*, (level of) *modularity* and *maintainability*, whether the BoundaryObject represents *prescriptive* knowledge (as opposed to descriptive), which *lifecycleStages* the BoundaryObject is used in with an enumeration of four options (*Planning*, *Operation*, *Deprecate*, *Retire*), *representationFormat* (e.g., free text, model, table,...), the *tooling* used, what sort of *versioning* information it may have, whether the BO is *upToDate*, and the level of *internalConsistency* as well as of *externalConsistency* with other BoundaryObject instances. These attributes are either free text (String) or are described via an enumeration, e.g., simple qualitative scale of *High*, *Medium*, and *Low* (the HighLow enumeration). We found that although this qualitative scale can be used for a quick summary, often a more complex description is needed. For example, for architecture descriptions, the level of detail of the BoundaryObject changes depending on the *lifecyleStages*. Thus, we find the need to accompany each attribute with a short explanation of the value. We omit this from the current metamodel for simplicity, but note that the instance models can be accompanied by some explanatory text.

A MethodologicalIsland (in green) contains an enumeration of types based on our past findings (*Teams*, *Silos (release trains)*, *Departments*, *Organizations*). For MethodologicalIslands, the relations to other elements are crucial. Organizational Roles, with role *name*s are part of the MethodologicalIslands. A Role is responsible for, or has a CRUD relationship with a BoundaryObject. The BORoleInteraction association class between these classes captures how Roles interacts with BoundaryObjects. We can model a BoundaryObject's *accessibility* for a Role, its *stability*, *criticality*, a summary of the *purpose*, and whether it is *fitForPurpose*. Ideally, a Role is part of an MethodologicalIsland, and the Role's interaction with the BoundaryObject is described in the BORoleInteraction class. However, a MethodologicalIsland could coordinate directly with a BoundaryObject without explicit representations of roles. For the specification of such a fact, we provide the association class BOMICoordination between BoundaryObject and MethodologicalIsland. In practice, we find this second BOMICoordination association (between MethodologicalIslands and BoundaryObjects) is often omitted.

Our past work uncovered the concept of MethodologicalIsland drivers, the reason for the MethodologicalIsland divide. We capture that as Driver drives a MethodologicalIsland, and describe possibly interesting attributes of the drivers, including enumerated *driverTypes* (*Technology*, *Process*, *Organization*), the *distanceTypes* culture/geography /organization inspired by [4, 18], and the size of the distance.

Finally, based on past work [53, 58], we find that governance of BoundaryObjects is crucial. Roles can be part of a GovernanceTeam, which governs a BoundaryObject. For instance, a Community of Practice is a potential governance team for architecture descriptions [53]. We collect interesting attributes of this relationship in the BOTeamGovernance association class, including the *coordinationMechanism* (e.g., meetings, processes, standards, tools), the *frequencyOfCoordination*, and the *purpose* of the governance in free text form.

Although other details could be added to this model, we aim for relative simplicity to better enable instantiation with and by our industrial partners.

3.2 Instance examples

To illustrate our model in action, we present three examples derived at different stages of our design cycle. First, we show two different examples from two different companies which happen to focus on the same BO, user stories. Our third example shows a simplified version of a BOMI model focusing System Requirements as envisioned to be used in the requirements management tool *T-Reqs* [27, 28, 30]. Other examples have focused on BOs as features, test cases, and requirements specifications. The first example comes from cycle 2, with Company A, the second example comes from cycle 3 with Company D, and the third example comes from cycle 3, focusing on a research tool. Because they originate in different design science cycles, beyond the different content, there are subtle differences in the attributes and naming, due to the evolution of our metamodel between cycles. We discuss the evolution of the metamodel more in Sect. 4.

Thus far, our instance examples again use UML syntax. In developing a BOMI language, we could create a domain-

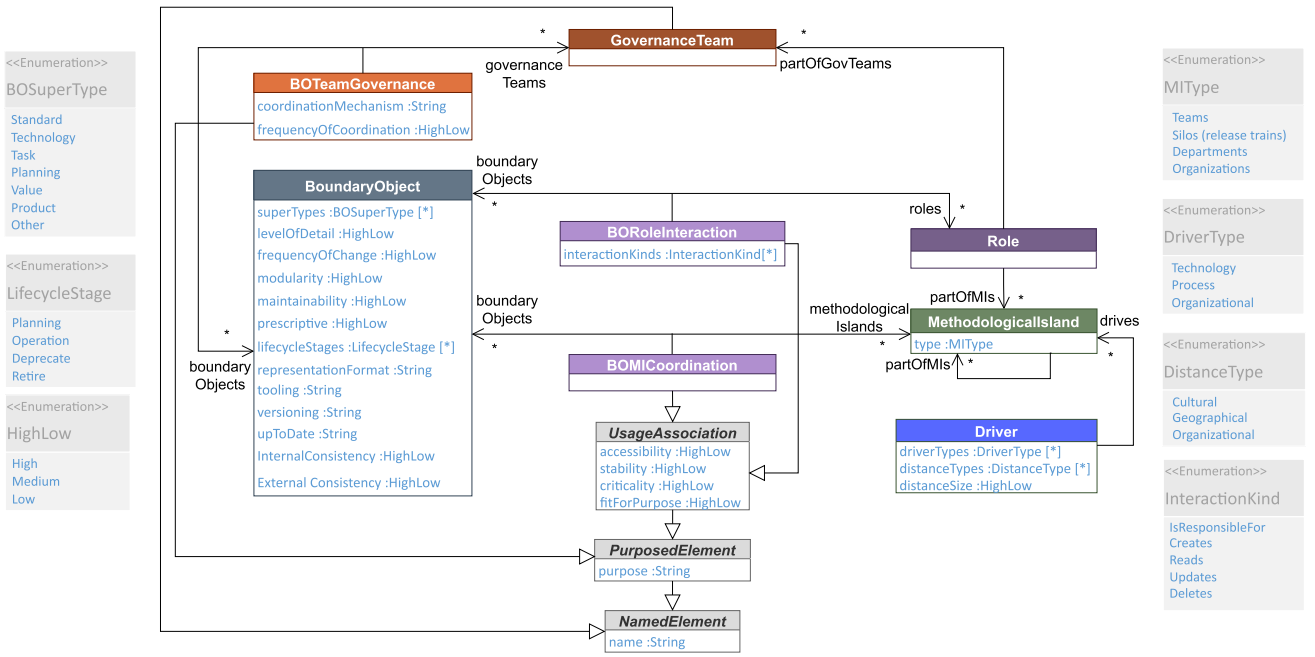


Fig. 1 Metamodel for boundary objects and methodological islands (BOMI)

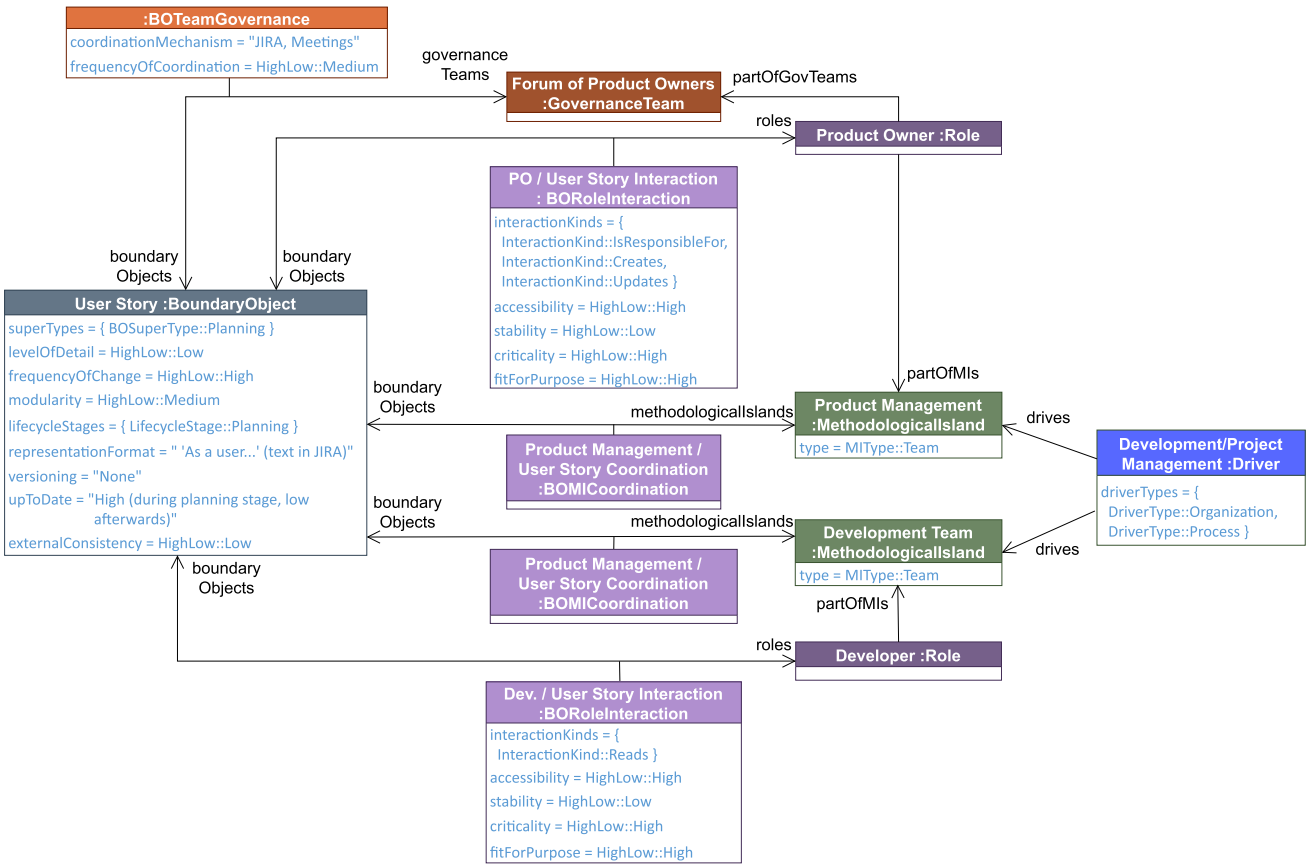


Fig. 2 Instance model of BOMI setup for user stories for company A

specific visual language, using customized icons or different shapes. We discuss this possibility more in Sect. 6.

In the Fig. 2 example, Company A (more detail in Sect. 4) chose to focus on a UserStory which is a BoundaryObject that is used in planning, acting as a backlog item. Other attributes include the *levelOfDetail*, *frequencyOfChanges*, and *representationFormat*. In this example, we include extra explanatory text for the attributes in parentheses. Two MethodologicalIslands, the Development Team and the Product Management, use this BoundaryObject for coordination. Developer and Product Owner roles are part of these MethodologicalIslands, respectively. The Dev. / User Story Interaction for the Developer is captured via a BIRoleInteraction association class, the attributes indicating that the User Story is easily accessible, critical, but with low stability, among other things. A similar BIRoleInteraction class called PO / User Story Interaction captures usage of the BoundaryObject by the Product Owner. The Product Owner is part of a Forum of Product Owners who make up the Governance Team for the User Story. The BOTeamGovernance association class captures attributes of the governance process, e.g., they coordinate using the JIRA tool and meetings, and coordinate at least once per agile sprint.

In the second example in Fig. 3, we see a more complex view of what BOMI instance models can look like when describing real situations. The increased size of the model is in part due to the longer time dedicated to modeling in the cycle 3 workshops, as compared to the short time in cycle 2. Here, we see several more Roles and MethodologicalIslands compared to Fig. 2. That is, there are the roles Scrum Master, Tester, Developer, and Architect as well as the MIs Testing Team, Development Team, Sys. Arch, and an ART Release Train (from the Scaled Agile Framework, SAFe [26, 41]). The model includes the Driver instances Cultural Perspective and SAFe, where Cultural Perspective culturally drives the Testing Team and Development Team MIs apart due to their differing cultures. As a process, SAFe drives apart the Development Team and Sys. Arch MIs due to the use of different development methods. Despite the longer time dedicated to the workshop, the model is incomplete in a few ways due to lack of time for eliciting more information. For example, it mainly focuses on the BORoleInteraction association called PO / User Story Interaction between the role of Product Owner and the BO User Story. Furthermore, the role Product Manager / Business Owner is not connected to the remaining model elements.

Finally, Fig. 4 gives an overview of the envisioned usage of the T-Reqs tool to capture system requirements by exploring a key BO with roles and MIs. Here, we focus on a BoundaryObject called System Requirements, including how it is used by Roles called Member XFT, Customer Facing Roles (Product Manager), Test Architect, and System Manager. These roles are part of the MethodologicalIslands called Development Level, Customer Level, and System Levels. The Member XFT role is part of the GovernanceTeam that governs the System Requirements.

We consider how instance models like these can be created and analyzed in the next sections.

3.3 BOMI method

As part of our modeling workshops, we created a simple list of guiding questions based on our metamodel concepts and attributes, e.g., "Which BO would you like to focus on?", "What roles interact with the BO?", and "Which islands do the roles belong to?". The full list of questions can be found in Table 1. These questions are intended to guide in the creation of a BOMI instance model, either led by a modeling facilitator, or independently in a company.

3.4 BOMI analysis

Although the process of creating a BOMI instance model is useful to understand BOs and MIs, one can go a step further and use the instance model created to detect potential issues or "smells" in the BOMI configuration, similar to the idea of smells in models or source code [3, 47]. The idea is that these smells can be detected and discussed, determining if there is an underlying problem. This analysis and discussion would be conducted by those having a higher-level view of an organization, e.g., team leaders or project managers. The overall aim is to promote potential beneficial changes in the BOs, MIs, and ways of working.

We can detect these smells within a BO, or across relationships in the model. For example, we can detect smells within individual attributes: low modularity, high maintainability, not up to date, not internally consistent, or not externally consistent. We can also detect possible smells between attributes, including: having a high level of detail but a high frequency of change, meaning that frequent changes may be difficult and involve changing many elements; and being in an early lifecycle stage (planning) yet being very infrequently changed, or being in a later lifecycle change (deprecate, retire) yet having a high frequency of change.

Similarly, with the two sub-classes BORoleInteraction and BOMICoordination of the UsageAssociation class, smells include not being fit for purpose, or high criticality with low stability or low accessibility. For instance, in Fig. 2, the usage of the BO by both the developer and product owner is critical—but the stability is low. Is it acceptable for something so critical to change so frequently? Looking into the BO, we see the lifecycle stage is planning, so the organization may argue that high criticality and low stability is unavoidable for key artifacts like user stories in this early stage. If the artifact was instead in an operational stage, this situation may pose more of a problem.

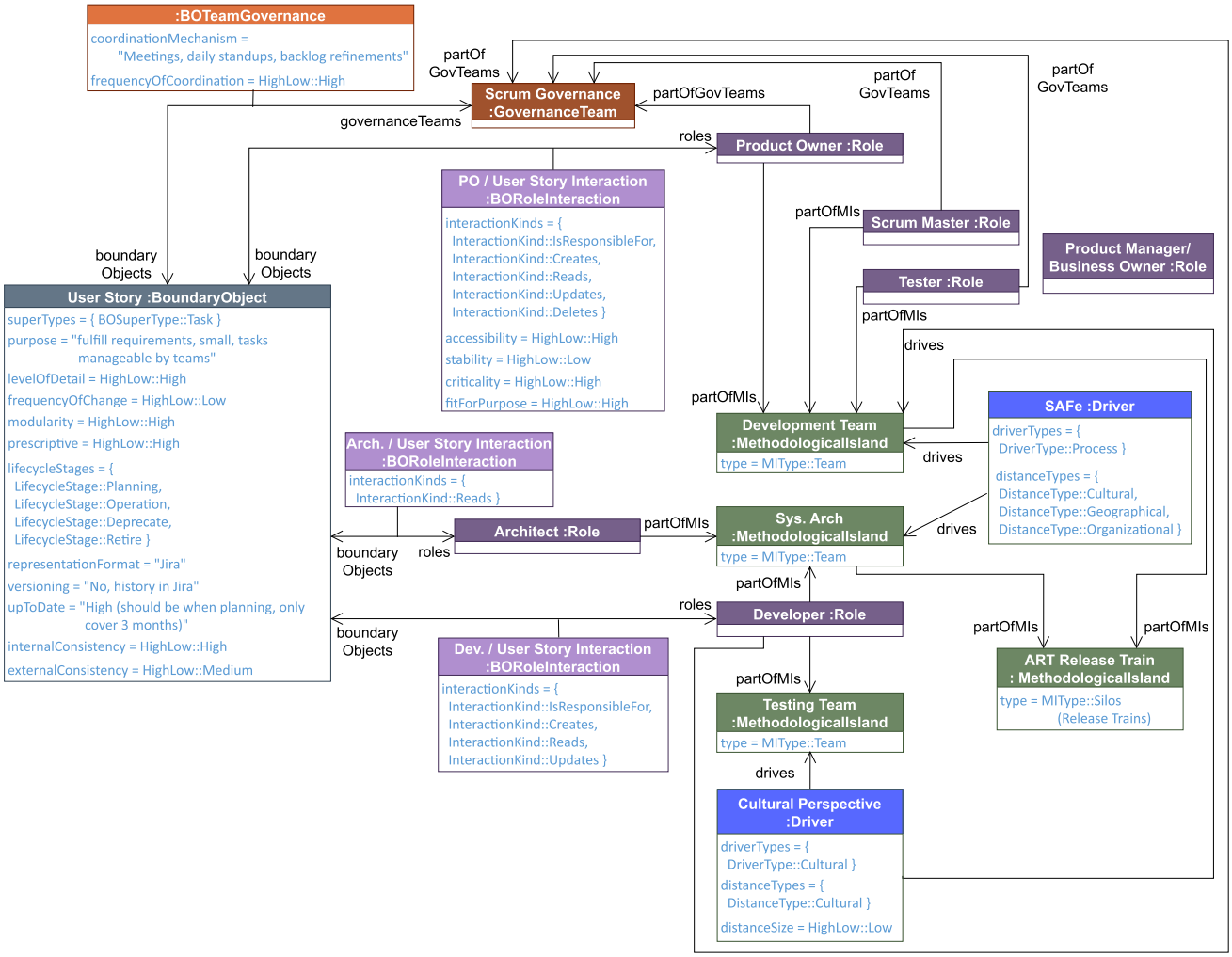


Fig. 3 Instance model of BOMI setup for user stories for company D

Smells can be detected across attributes in different classes. For example, a BO has a high frequency of change, but there is a low frequency of coordination in the BOTeamGovernance class, possibly indicating that the governance may not keep up with the rate of change. Similarly, there can be a high frequency of change in the BO, but a high stability in the BORoleInteraction / BOMICoordination of the BO, or a low frequency of change in the BO and a low stability in the BO's BORoleInteraction / BOMICoordination. This may indicate a conflict between the design or nature of the BO and how it is used by a particular role or MI.

We can also detect smells at a broader level, for example, the BO has no governance team or no one responsible for it. Our company partners suggest that those governing a BO should also use it, to ensure that they are aware of how the BO is used. It can also be checked whether there exists someone who can update and delete the BO. And, if the BORoleInteraction or BOMICoordination is critical, or if the frequency of change is high, the BOTeamGovernance should likely have a high frequency of coordination.

Based on the high-level metamodel depicted in Fig. 1 in Sect. 3.1, we specify the smells by means of the Object Constraint Language (OCL) [36] to illustrate how automatic checking can look like. We summarize these OCL constraints in Table 2 and formalize them in an automatically executable manner as part of the BOMI tool support in Sect. 3.6.3.

3.5 BOMI views

Our experiences in iteratively evaluating the BOMI framework (Sect. 4) led us to the need for views for BOMI models, in order to manage the complexity of such models. In this context, we refer to the view-based modeling taxonomy of Goldschmidt et al. [15], who refine the notion of viewpoints as defined in the ISO/IEC/IEEE 42010 [22]. According to this taxonomy, stakeholders have certain concerns that are addressed by different viewpoints of a modeling lan-

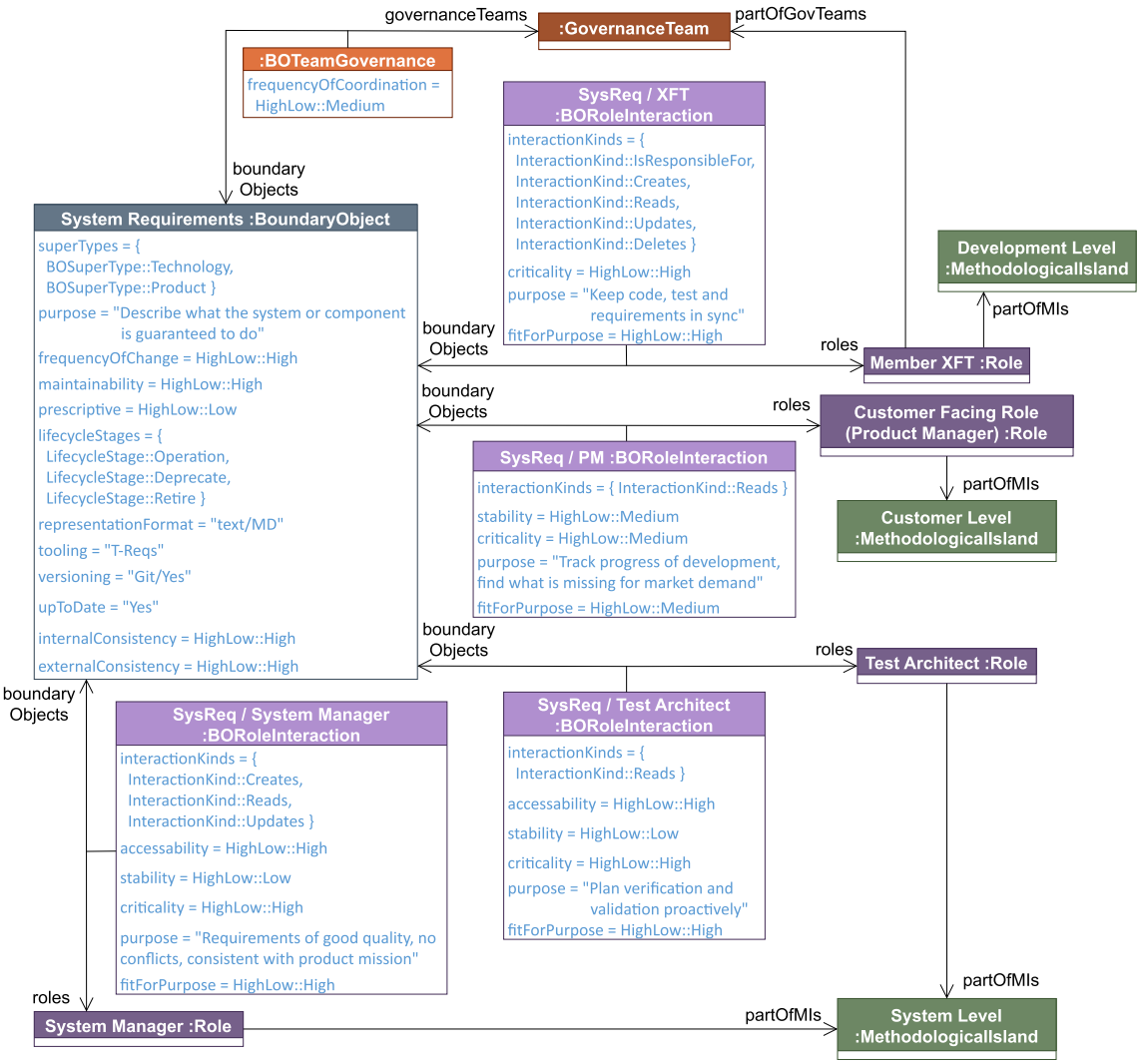


Fig. 4 Instance model of BOMI setup for system requirements for the envisioned use cases of the T-Reqs tool

guage. Particularly, a viewpoint collects different *view types* that define which modeling elements shall be represented in which concrete syntax (i.e., a view type defines a diagram kind). A concrete view then adheres to its corresponding view type and displays one concrete set of model instances (i.e., a view is one particular diagram).

The identified view types are as follows: Beyond the *BOMI details view type* that displays all BOMI model information, the identified view types encompass the *overview view type*, the *methodological island (MI) view type*, the *boundary object (BO) view type*, and the *governance view type*.

The overview view type simply displays all model concepts but does not display any of their attributes, thereby abstracting from the detailed information and providing an initial big picture understanding.

The remaining view types stem from the identified stakeholder concerns of managing MIs and understanding the coordination between each BOs, roles, and governance teams. Thus, these view types focus on the particular BOMI aspects by displaying only each a corresponding subset of selected model concepts including their relationships and attributes.

More specifically, the MI view type focuses on managing and understanding MIs by displaying from the overall BOMI metamodel (cf. Fig. 1) only the subset of metaclasses Methodologicalllsland, Role, and Driver including their relationships and attributes. For example, Fig. 5 depicts an MI view on the instance example from Fig. 2.

Similarly, the BO view type focuses on the understanding of the coordination between BOs and the roles, hence displaying from the overall BOMI metamodel only the subset of metaclasses BoundaryObject, Role, and the intermediate association classes BORoleInteraction and BOMICoordination including their attributes.

Table 1 Guiding method for creating BOMI instance models

Overview
– Which BO would you like to focus on? Pick one that's problematic or interesting in some way
– What roles interact with the BO?
– Which islands do the roles belong in?
– Is the BO governed, by whom?
About the BO
– What type is it?
– What is the: level of detail, frequency of change, modularity, etc.. (for all attributes)
About the Roles
– Do the roles CRUD with the BO?
– What about Accessibility, stability, purpose, criticality, etc. (for all attributes) for that role?
About the Islands
– What type are they?
– What is the distance, driver between each island
About Governance
– Which roles are part of the governance team?
– What frequency of governance, formality of governance

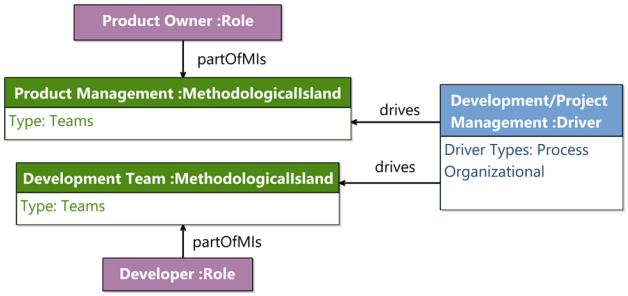


Fig. 5 Example of an MI view (cf. Fig. 2)

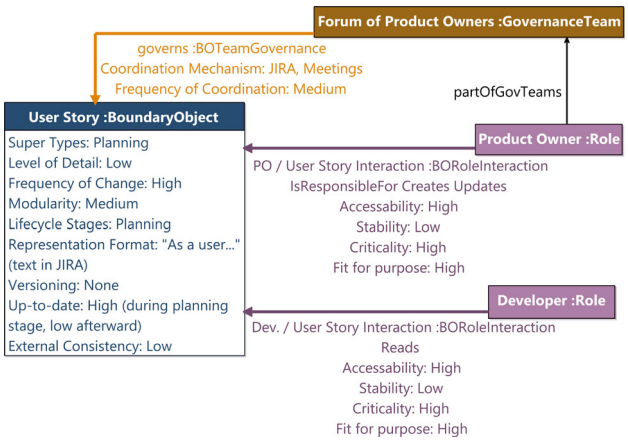


Fig. 6 Example of a governance view (cf. Fig. 2)

Likewise, the government view type focuses on understanding the governance aspects and consequently displays from the overall BOMI metamodel only the subset of metaclasses BoundaryObject, GovernanceTeam, the intermediate association class BOTeamGovernance, Role, and the intermediate association classes BORoleInteraction and BOMICoordination. From these metaclasses, the governance view type displays all attributes except for the intermediate association classes BORoleInteraction and BOMICoordination. For example, Fig. 6 depicts a governance view on the instance example from Fig. 2.

3.6 The BOMI modeling tool

Initially, we applied the general-purpose drawing tools diagrams.net / draw.io [23] and initially Microsoft Visio to specify the BOMI models. This led to some issues in specifying particularly more complex models. That is, a BOMI-specific tool support would give more guidance during the model specification and also ensure a uniform concrete syntax. Furthermore, we discovered the need for the automatic execution of checks on our smells. Finally, as part of our design science evaluation (Sect. 4), we came across the idea of establishing dedicated views on complex BOMI models.

All these aspects can be enabled by a tool based on a domain-specific modeling language. This is especially true as we already conceived a BOMI metamodel as well as constraints specifying the smells (cf. Sect. 3.4). In this section, we hence present our *BOMI modeling tool*, which we publish as open-source software [19]. For convenience reasons, we provide both a visual and a textual syntax including the corresponding editors to specify BOMI models in the tool. In the following, we describe the further formalization of the high-level metamodel to an automatically processable one, the different concrete syntaxes, and the further formalization of the BOMI smells to automatically executable checks.

Table 2 Smells in BOMI model instances with associated OCL expressions

Type	Description	OCL Expression (updated from [54] w.r.t. updated metamodel)
Within BO	Low modularity	context `BoundaryObject` inv LowModularity: self.*modularity* = *HighLow::Low*
	Not internally consistent	context `BoundaryObject` inv InternalInconsistency: self.*internalConsistency* = *HighLow::Low*
	High level of detail and frequent change	context `BoundaryObject` inv DetailedHighChange: self.*levelofDetail* = *HighLow::High* and self.*frequencyofChange* = *HighLow::High*
	Later lifecycle and frequent change	context `BoundaryObject` inv LateHighChanges: (self.*lifecycleStages*→includes(*LifecycleStage::Deprecate*) or self.*lifecycleStages*→includes(*LifecycleStage::Retire*)) and self.*frequencyofChange* = *HighLow::High*
Within Usage	Not fit for purpose	context `UsageAssociation` inv NotFit: self.*fitForPurpose* = *HighLow::Low*
	High criticality and low stability	context `UsageAssociation` inv CriticalUnstable: self.*criticality* = *HighLow::High* and self.*stability* = *HighLow::Low*
Missing elements/ relation-ships	No governance team	context `BoundaryObject` inv Governed: self.*governanceTeams*→isEmpty()
	No one responsible for BO	context `BoundaryObject` inv Responsible: BORoleInteraction→allInstances()→exists(int | int.*interactionsKinds*→includes(InteractionKind::IsResponsibleFor) and int.*boundaryObjects*→includes(self))
	No one can update BO	context `BoundaryObject` inv Updated: BORoleInteraction→allInstances()→exists(int | int.*interactionsKinds*→includes(InteractionKind::Updates) and int.*boundaryObjects*→includes(self))
Across elements	Governing roles should use BO	context `BoundaryObject` inv GovernsUses: BORoleInteraction→allInstances()→exists(int | int.*boundaryObjects*→includes(self) and int.*roles*→exists(r | r.*partOfGovTeams.boundaryObjects*→includes(self))
	High frequency of change but low frequency of coordination	context `BoundaryObject` inv HighChangeLowCoord: self.*frequencyofChange* = *HighLow::High* and self.*governanceTeams*→exists(g | g.*frequencyofCoordination* = *HighLow::Low*)
	Low frequency of change but high frequency of coordination	context `BoundaryObject` inv LowChangeHighCoord: self.*frequencyofChange* = *HighLow::Low* and self.*governanceTeams*→exists(g | g.*frequencyofCoordination* = *HighLow::High*)
	High frequency of change but high stability	context `BoundaryObject` inv HighFreqHighStab: self.*frequencyofChange* = *HighLow::High* and UsageAssociation→allInstances()→exists(int | int.*stability* = *HighLow::High* and int.*boundaryObjects*→includes(self))
	Low frequency of change but low stability	context `BoundaryObject` inv LowFreqLowStab: self.*frequencyofChange* = *HighLow::Low* and UsageAssociation→allInstances()→exists(int | int.*stability* = *HighLow::Low* and int.*boundaryObjects*→includes(self))

3.6.1 Abstract syntax

As base technology, we apply the Eclipse Modeling Framework (EMF) [11], which enables creating modeling tools based on the open-source plugin platform Eclipse. EMF uses the Ecore file format for the formal specification of metamodels, that is, to define the abstract syntax of our BOMI modeling language. We formalized the high-level BOMI metamodel from Fig. 1 in Sect. 3.1 by means of an Ecore metamodel, which is depicted in Fig. 7.

The BOMI metamodel formalized in Ecore has slight differences to the high-level one due to the EMF requirements on the specification of automatically processable metamodels and certain design decisions. This includes that EMF requires a container object for persisting the corresponding models (cf. BOMIModel in Fig. 7) and that association classes are not directly supported but are realized through classes with each two unidirectional links to the associated classes (e.g., BORoleInteraction and BOMICoordination with their superclasses in Fig. 7).

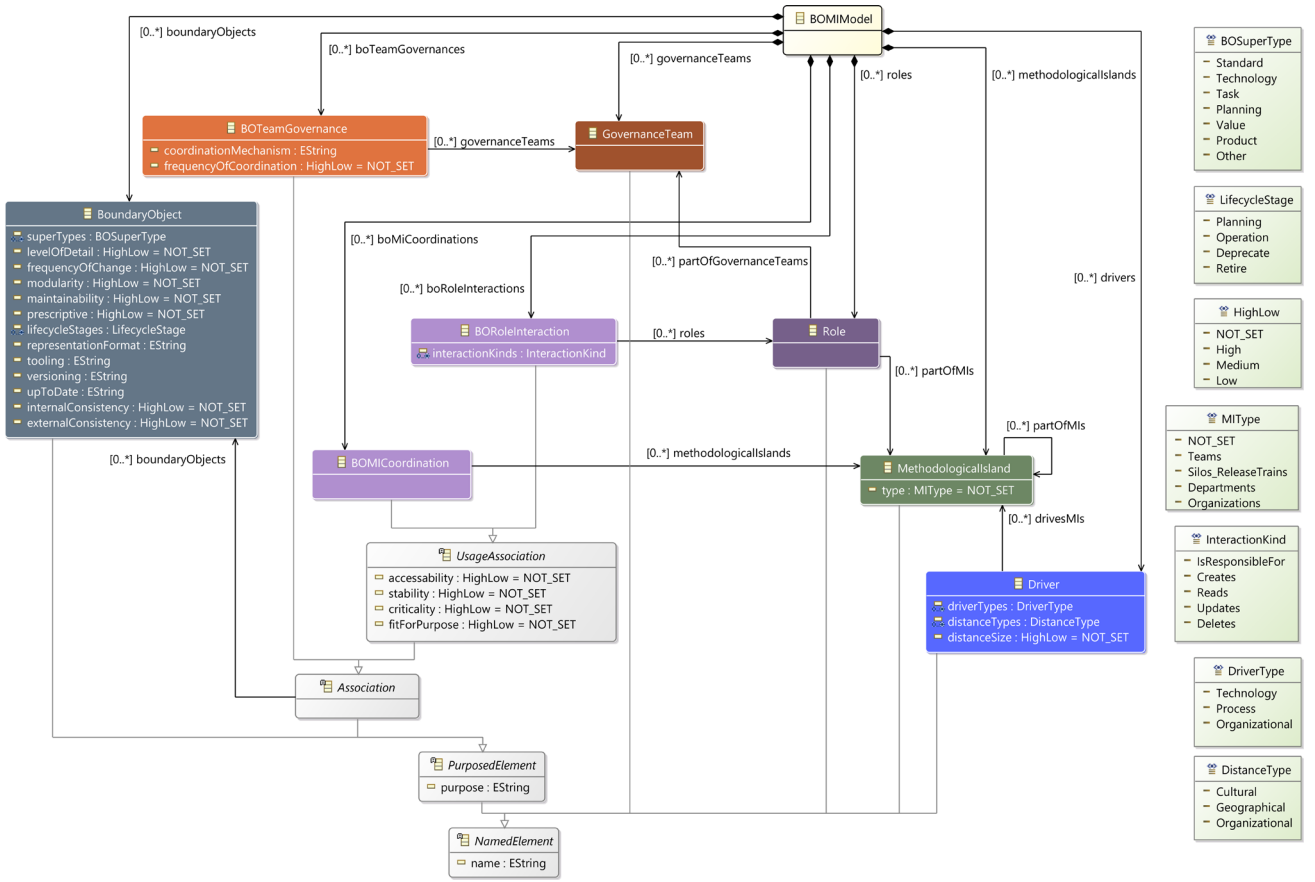


Fig. 7 The BOMI metamodel formalized in Ecore

3.6.2 Concrete syntaxes

Initially, to represent BOMI models, we drew them in the general-purpose drawing tools using a visual and informal boxes-and-lines syntax similar to UML class diagrams. This visual representation helped in the workshops to present and discuss the BOMI models, and it was well received by the companies. Thus, we kept this basic syntax in our tool development. In addition to the visual syntax, one company asked us to provide a textual boilerplate language to identify more easily a certain BOMI concept like a Role.

In order to consider such multiple and heterogeneous concrete syntaxes, we follow the paradigm of *blended modeling*, which "is the activity of interacting seamlessly with a single model (i.e., abstract syntax) through multiple notations (i.e., concrete syntaxes)" [6]. The Eclipse plugin platform supports the blended modeling paradigm to a certain extent by using EMF as the basis for the abstract syntax specification and by the freedom to choose from multiple EMF-based concrete syntax frameworks. Thus, we chose one EMF-based framework for a graphical and one for a textual concrete syntax.

For providing a boxes-and-lines UML-like concrete syntax as we applied in the general-purpose drawing tools and the informal BOMI models used in the preceding figures, we apply the EMF-based graphical modeling framework Sirius [12]. Sirius natively supports the view-based modeling taxonomy [15] and the ISO / IEC / IEEE 42010 viewpoint paradigm [22] as introduced in Sect. 3.5, so that we can directly transfer the identified view types (identification described in Sect. 4.4) to a graphical concrete syntax specification. For example, the left-hand side of Fig. 8 depicts a screenshot of the BOMI details view representing the BOMI instance model from Fig. 2 in Sect. 3.4. We provide the example and different views as part of our BOMI modeling tool repository [19].

For providing a textual boilerplate language, we apply the EMF-based textual language engineering framework Xtext [13]. For this purpose, we used Xtext's built-in functionality of deriving a grammar out of an existing Ecore metamodel, in which we fed in the BOMI metamodel (cf. Sect. 3.6.1). In this context, we did not further refine the grammar. Based on the grammar, we use the further Xtext feature to generate the complete textual editor infrastructure, yielding a textual BOMI model editor that provides features such as syntax highlighting and content assist for keywords and cross-references. For example, the right-hand side of Fig. 8

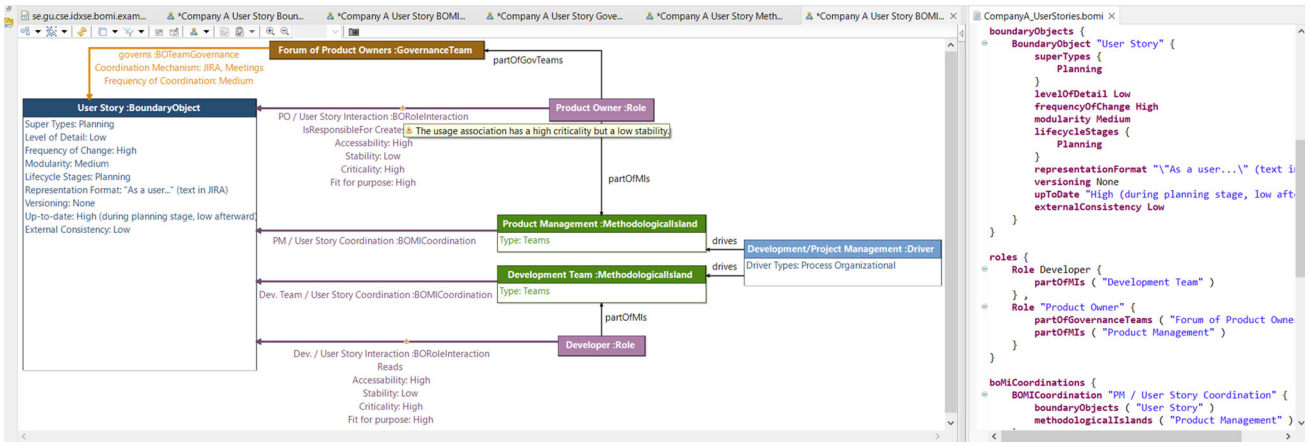


Fig. 8 A screenshot of the BOMI modeling tool

depicts a screenshot of the BOMI textual editor, which represents the same BOMI model contents depicted on the left-hand side. As the textual format is applied as the storage format for BOMI models, EMF synchronizes the different graphical views and the textual representation on every file-saving operation. For those who may prefer to create or edit models via text, the full Xtext grammar is available as part of the BOMI modeling tool repository [19]. The textual editor supports this process with content assist, making it easier to remember the supported language syntax and naming.

3.6.3 Automatic checks on BOMI smells

In Sect. 3.4, we introduced OCL constraints that specify the BOMI smells based on the high-level BOMI metamodel. To transfer the smells into tool support based on the formal BOMI Ecore metamodel (cf. Sect. 3.6.1), we formalized the OCL constraints by means of the Acceleo Query Language (AQL) [10]. AQL is integrated into Sirius (cf. Sect. 3.6.2) and similar to OCL, but it is more permissive regarding missing or empty intermediate results in complex expressions [14]. Thereby, it enables less cumbersome specifications due to a strong reduction of null-checks and castings.

For example, the BOMI modeling tool screenshot in Fig. 8 depicts on the left-hand side the BOMI model instance example from Fig. 2. As explained in Sect. 3.4, this BOMI model instance encompasses a usage association between the Product Owner role and the User Story boundary object, where the usage association has a high criticality paired with a low stability, which we consider a BOMI smell (cf. Table 2). In the screenshot, this association is depicted as an arrow labeled Association: PO / User Story Interaction between the role and the boundary object. An exclamation mark denotes the detection of a formalized smell, and on hovering over the exclamation mark a corresponding warning message for the smell is displayed.

Thus, the BOMI modeling tool provides automatic checks that are fast to execute and that support the user particularly for large BOMI models, thereby preventing issues as discovered in the workshop with company D in cycle 3 (cf. Sect. 4.3.1). The output of such automatic checks should then be discussed within an organization, to determine if the smell is a problem in reality, and to discuss what sort of changes could be made.

4 Methodology & evaluation results

In order to develop the BOMI framework presented in Sect. 3, we applied a design science approach [17] with the BOMI framework as an artifact. We went through several cycles designing the artifact and performing evaluations of the artifact both locally and with five companies. In this section, we present both the design science methodology, including information on all cycles, and describe the results of each cycle, the final version of which has been described in Sect. 3.

The five companies we collaborated with are described in Table 3. An overview of the input artifacts and cycles of our methodology is shown in Table 4. We describe the design of each cycle and the results in the following subsections.

4.1 Cycle 1: authors' experience and industrial examples

As part of the first cycle, we used in an initial iteration our informal drawings and lists of collected BO and MIs in practice, along with our knowledge gathered from the companies, to come up with a first draft of the artifact. The paper's authors discussed the artifact and made local improvements. In a second iteration, we used historical data gathered from workshops with two of the companies to create trial models of their BOMI situation. After discussion, this caused a fur-

Table 3 Descriptions of participating companies

Company A	Develops telecommunications products. Separate organizational units exist for sales, product management, and other purposes
Company B	Develops mechanical products, both for consumer markets and for industrial development and manufacturing. The systems are decomposed into several elements, which is also reflected in the organizational structure
Company C	Is an automotive Original Equipment Manufacturer (OEM). Traditionally, the company has been structured according to vehicle parts (e.g., powertrain, chassis,...), but has undergone restructuring into agile teams
Company D	Develops high-tech solutions for vehicular systems. Software development teams are largely independent of hardware development
Company E	Develops hardware and software products for consumers and industries. Follows a test-driven, scaled agile methodology

ther improvement of the artifact. Very initial versions of the BOMI metamodel can be found in our online appendix [52].

4.2 Cycle 2: short workshop with four companies

The second cycle in our method (previously presented in [54]) was a 1.5-hour online workshop in April 2020 to try out the metamodel, method, and smell ideas with seven representatives from companies A–D, described in Table 3. The participants included systems engineers, requirements specialists, and tooling specialists. During the workshop, we reserved 20 min for a review of BOMI concepts and to introduce the new metamodel using prepared material [52]. We then split off into four virtual break-out rooms for 30 min of modeling instance models in focused sessions. Each room had at least one researcher and the representatives from one company. The researchers went through the guiding method questions from Sect. 3.3 and drew an instance model based on the answers of the participants, sharing their screen. The final 30 min (allowing for short breaks) was used to discuss our experiences and gain feedback, with several of the authors taking notes. The authors then met to share and review our notes, consolidating and discussing experiences.

Despite the short time-frame of the modeling part of the workshop, we were able to get four relatively complete models (e.g., Fig. 2), with the statistics in terms of element type used shown in Table 5. We opted to focus on one BO at a time; thus, each model had only one BoundaryObject. The modelers were also able to capture 2–5 MethodologicalIslands, 1–5 UsageAssociation classes, 1–4 Drivers, and one GovernmentTeam and BOTeamGovernance association class per model. Some of the attribute information for each model was filled in, but many attributes were left blank due to time restrictions.

The final 30 min of the workshop was used to discuss our experiences and elicit feedback. Feedback included that the current typing hierarchy for MIs was often hard to apply, and MIs are often multi-dimensional. To deal with this, we allowed MIs to have more than one type in the updated metamodel. We also acknowledge that our current list of possible enumeration types (MIType in Fig. 1) may not be complete. Previously, instead of the Driver class, we had an Ocean association class between MethodologicalIslands with a driver attribute. We noted in our modeling exercises that MethodologicalIslands can have many drivers and can share drivers. Thus, we reworked the Ocean association class

Table 4 Overview of the design science cycles of our research method

Cycle	Description	Problem	Solutions	Evaluation
1	Authors' experience and industrial examples	Lack of a systematic way to assess coordination issues	Initial draft of the BOMI framework (method, metamodel, and smells) through generalizing knowledge from experiences with companies	Internal discussions and local improvements
2	Short workshop with four companies	Ideas about BOMI framework not tested	Improvements of the BOMI framework through experimentation in a 1.5h workshop with seven representatives from four companies	Discussions with and feedback from companies; consolidating and discussing own experiences
3	Multiple in-depth workshops	Unknown degree of stability of BOMI framework	Stable BOMI framework plus additional modeling guidelines through application in three 3 h in-depth workshops with a different company each plus one internal workshop	Feedback by companies; own reflections
4	View types and tool support	Information overload and lack of dedicated tool support	Bachelor's thesis [50] eliciting needs for BOMI views from companies; implementation of the BOMI framework including the view concept and smells into the BOMI modeling tool	Feedback on view types by companies; feedback on modeling tool ongoing through further Bachelor's thesis and cooperation with companies

Table 5 Element count of four instance models from the cycle 2 workshop

Model	BO	MI	Usage	Driver	Role	Governance Team	Governs
C1	1	2	2	1	2	1	1
C2	1	3	1	1	5	1	1
C3	1	5	5	4	0	1	1
C4	1	3	1	2	2	1	1

to the current Driver class. We also made note that most of the attribute descriptions were hard to capture with enumerations (High/Medium/Low) and that we often needed free-text descriptions to capture the subtleties, e.g., frequency of change varying depending on the lifecycle stage. Finally, we made many small improvements to the class attributes. These changes are reflected in previous versions of the model in our online appendix [52].

Our modeling sessions in this cycle did not give us extensive time to apply the smell analysis examples as described in Sect. 3.4, and we were also hindered by the incompleteness of some of the instance model attributes. However, we presented some draft smells and asked for feedback from the participants. We generally asked "Can the current issues with the BO be captured in the model?" Although the participants were not opposed to automated checks as described in Sect. 3.4, they were more interested in human-centered manually-detected smells, e.g., "Can I draw this?" For them, the first and most important smell is whether the participants had the knowledge to instantiate the metamodel. Our participants also suggested a smell having to do with the complexity of the overall model: "I can draw it, but it is a mess", indicating that the overall design of their BOMI situation could be overly complex and poorly thought-out. Therefore, model complexity checks or basic checks such as for cohesion and coupling may be useful. Our participants also suggested the check that those responsible for governance should also be users, and that the governance team should consist of a diverse set of roles or islands, i.e., not just be made up by one type of user. Some of these smells could be expressed formally over the model, as in Sect. 3.4, but others can instead be included as points to consider in the methodology.

Overall, our company partners were positive about the experience. Based on their interest, we were able to arrange longer sessions for the same and a further company, inviting internal participants knowledgeable about key BOs.

4.3 Cycle 3: multiple in-depth workshops

In the subsequent cycle, new to this work, we conducted longer, in-depth workshops with three different companies (A, D, and E), one of whom did not participate in the previous cycle. We also conducted an internal workshop focused on BOMI analysis of an open source tool for versioning of textual requirements (T-Reqs [27, 28, 30]). In the company cases, we coordinated with our company contacts beforehand to select a topic of focus for the workshop (e.g., agile product owners and user stories, feature capture and changing tools, test case coordination). Our contacts picked scopes that were either critical to the company, or interesting or problematic in some way, deserving exploration. This helped to determine whom to invite to the workshop, aiming for approximately 3–8 participants who had knowledge of the agreed-upon scope. Typically, if the focus was on a particular BO (e.g., user stories, test cases), the company contact invited one to two people holding each of the types of roles who interacted with these BOs (e.g., project managers, product owners, developers).

Each of the company workshops was three hours in total, held virtually via MS Teams with screensharing. Workshops were held between autumn of 2020 and spring of 2021. We spent one workshop hour providing a background on BOMI modeling, including motivation, background, the metamodel, method, and smells. We then conducted a two-hour modeling workshop. In two out of three cases, the one-hour tutorial was given during a different date, roughly a week before the modeling session, in order to allow participants to digest the material.

During the two-hour modeling session, in cases where the one-hour BOMI tutorial was not on the same day, we showed a short reminder of the BOMI concepts and metamodel (similar to the material for cycle 2 available online) and, in all cases, we would re-iterate the target scope of the modeling session. Modeling started with a blank version of the BOMI metamodel (see Sect. 3.1) using an online model drawing tool, diagrams.net / draw.io [23]. One of the researchers facilitated the meeting and made changes to the model with screensharing, asking for opinions and fixes. Following the method (see Sect. 3.3), the facilitator asked questions (e.g., what is the BO? Which roles interact with it?) and use the answers to instantiate the metamodel. The facilitator continuously asked for feedback on the model (e.g., is this right? Should this be...?) and made updates based on the discussion. After the model creation process neared to a close (roughly 1 h), we reserved 15 min to evaluate model smells, i.e., looking for coordination issues highlighted or revealed by the

model (see Sect. 3.4), using our existing list and asking for further smells.

The workshop design reserved a final 10–15 min to ask for general feedback on the process and metamodel. Notes were recorded either by the facilitator, other participating researchers, or both. We experienced a good level of interactivity and engagement from almost all participants as part of these meetings. A "cleaned up" version of the model, usually with better layout and straightened connectors was shared with the participants after the workshop, asking for further corrections and/or feedback. We present results for each of the workshops in the following.

4.3.1 Workshop with company D: agile product owners

In this case, the company was interested in understanding and analyzing the tasks and coordination efforts of the product owners as part of their agile methodology. The BO of focus was user stories. Specifically, they took the opportunity to compare BOs and MIs associated with product owners in different teams in different areas of the company, thus producing two final BOMI models. The motivation was to understand perceived differences in ways of working between the teams. We started modeling focusing on one team, then took the final model for that team and modified it to capture differences for the second team. We had our company contact as well as members from each of the different teams, including the product owners, present virtually in our workshops.

Statistics in terms of element type used for these models and the other models created in cycle 3 are shown in Table 6. One can see that these later-cycle models are much larger, compared to cycle 2, given the increased modeling time in the workshop. In this workshop, the models workshop company D #1 and #2 were created, focusing on product owners in different teams. A slightly anonymized, with some company-specific organizational names changed, final version of one of the models from this cycle (workshop company D #1) is shown in Fig. 3.

In the end, the workshop company D #1 and #2 models focusing on coordination for product owners in different areas of the company were relatively similar, with a few differences in MIs and drivers. One of the main differences between the ways of working between the product owners was tooling and coordination mechanisms. The different teams used different tools to capture their user stories, and they followed different flavors of agile methodologies for coordination. These differences appeared in the attributes of the BO (*representationFormat* in the BoundaryObject class and *coordinationMechanism* in the BOTeamGovernance association class), but did not have a great effect on the overall configuration (presence or absence of classes) in the model. This issue of tooling arose again in later workshops; we return to this topic as part of the workshop with company A (cf. Sect. 4.3.2) and of the internal workshop (cf. Sect. 4.3.4).

One issue that emerged was how to capture flow or bottlenecks in the model. In one case, there was a bottleneck in the governance process, perhaps leading to a kind of over-governance. This was difficult to capture with the current attributes and classes. However, after the fact, we can notice that the frequency of governance coordination was high, while the frequency of BO (user story) change was low. This could perhaps be considered as an undetected smell. Having a tool which supports auto-checking of such smells could have helped us to discuss this potential issue during the workshop. Thus, we implemented such tool support as part of cycle 4 (cf. Sect. 4.4).

Another point brought up was how to capture dependencies between BOs, when a BO depends on another BO. The participants wanted to focus on the relationships between user stories and other BOs. We could argue that this may be captured by more of an information architecture view, as opposed to a focus on coordination, for which BOMI is intended.

Another point made both in this workshop, and later by our main company contact was a discussion on whom to invite to such a workshop. Such a model may not be relevant for those who are not concerned with the big picture, and we may think about who creates BOMI models vs. who reads them. In this case, developers may not be interested in co-creating such a model, but perhaps would be interested in reading such a model, e.g., as part of training. We discuss this issue further in Sect. 6.

Overall, the model was not obviously "smelly", that is, we did not detect many smells on our list. However, the company still found the BOMI modeling to be a useful exercise, even though they did not experience many revelations (i.e., they did not gain extensive new knowledge).

4.3.2 Workshop with company A: features

We repeated the same workshop structure with company A. In their case, they were transitioning from one tool to another to store and manage features, a key BO for them. They wanted to understand the coordination and interaction surrounding features, and how this may change with a new tool. The workshop resulted in a rather large model (e.g., nine roles, six MIs, two governance teams) describing how the company interacts around features.

In this case, the tool change did not make significant differences to their ways of working in terms of organizational coordination. The feature BO would become more *upToDate*, an attribute of our BoundaryObject class, and the representation format would change when a new tool was introduced. One potential smell found in the model was the presence of two different governance teams with different coordination

Table 6 Element count of instance models from the workshops for cycle 3

Model	BO	MI	Usage	Driver	Role	Governance Team	Governs
Workshop company D #1	1	4	1	2	7	1	1
Workshop company D #2	1	3	1	1	7	1	1
Workshop company A	3	6	8	5	9	2	2
Workshop company E	3	9	8	4	8	1	1
Workshop internal	3	4	10	1	5	1	1

mechanisms and frequencies, and different roles as members. Speaking to the stakeholders at the workshop, these two levels of governance were working well, as they had different levels of abstraction and areas of focus (one focusing on the backlog grooming, and another on the product). This serves as an example of why we use the terms smell and not problem or issue: although a flag for potential concern, the stakeholders felt this double role was not a problem in practice. Overall, the model was again not particularly 'smelly'. Although it was complicated, the stakeholders pointed out that this was one of their core BOs, and because they were an established, successful company, coordination surrounding this feature BO must work well. Thus, smells were not expected.

Because of experiences in this and the previous cycles, we made minor changes to the BOMI metamodel. We added *tooling* as an attribute to the BoundaryObject class, as our previous *representationFormat* did not quite capture this aspect, coming up as important in several workshops. The company also expressed the importance of explicitly capturing the purpose of the BO for each role, MI, and governance team. Thus, we created a new abstract class PurposedElement carrying a *purpose* attribute and added inheritance relationships from the respective association classes to this class. Other attributes which were consistently difficult to fill-in and did not generate good discussion were dropped.

4.3.3 Workshop with company E: test-driven development

Company E wanted to try out BOMI in a workshop focusing on test cases and their test-driven development. This particular company has had a lot of success using tests as first-class objects, capturing requirements and other key information, so it is particularly important to understand coordination around these BOs. Particularly, there were long-term concerns about the scalability of the current ways of working. However, the company coordinator opted to first pick a topic that was not so problematic to try out the technique. In this case, we had all three hours of the workshop at once. For this particular company, we were also invited to two follow-up meetings ran by the company which discussed issues brought up in the workshop. The modeling workshop produced a rather large BOMI model with a BO of focus (Test Cases), but the model also included two other related BOs without the details filled out (Test Runs and Test Documentation). We identified eight roles and nine MIs.

In this case, the governance team and way of working were a bit more difficult to elicit, but we were able to add these elements after discussion. The discussion on this topic revealed some issues on traceability of changes and new information, and non-optimal ways of working in terms of notification of updates, according to at least one of the roles in the meeting. Although these issues were associated with elements in the BOMI model, it was tricky to capture the specific issue of notification and awareness using the BOMI constructs. In this case, there was also a lot of discussion regarding the organization structure with its roles and islands, as the landscape was somewhat complex and company specific. However, the company is also long-standing and successful, and their current structure works well for them at this time. Similarly to the other cases, we did not find many specific smells, other than the difficulty in drawing governance and in the researchers understanding and capturing the organization structure. Like in the last iteration, we covered a core BO for a successful company, and as such should expect to find a way of working that is overall successful.

The follow-up meetings discussed related issues in testing. For the first one-hour meeting, the BOMI model was not especially relevant for the discussion. However, the second one-hour meeting focused more on the issues of notifications and communication as unearthed during the BOMI modeling workshop. In this case, rather than showing the large model with the class diagram syntax, our company contact created a simple model distilling the model down to what he believed were the key elements and relationships. That is, they adapted the model to five BOs (the original model had three, so detail was added here), and reduced it to three roles from eight and to four MIs from nine. All attributes were hidden, and the relationships were simplified to "creates", "links to", etc. The discussion was then guided by this model with simplified views.

We believe that this practice of creating simpler or distilled version of the full model is very useful, particularly for communication with those who were not present in the initial workshop. However, it is hard for us to extract rules to automatically create this transformation, but we believe it is

useful to pursue the idea of more simplified views. Thus, we address this notion of views as part of cycle 4 (cf. Sect. 4.4).

4.3.4 Internal workshop: the T-Reqs textual requirements tool

We repeated the workshop design evaluating the BOMI metamodel and method on an internal tool, T-Reqs [27, 28, 30]. T-Reqs supports storage of requirements as text in Git, supporting version control, and treating requirements like a design artifact using the same type of tooling and methods. The tool also supports traceability to code. The aim of the BOMI workshop was to understand the collaboration of the proposed usage case and to test out BOMI in the face of multiple BOs in one model. Previously, we had only one BO of focus. In some cases, we added other, related BOs, but did not get into their attributes or relations to roles and MIs. In this case, we focused on three BOs: Stakeholder Requirements, System Requirements, and RIM/TIM (Requirements Information Model and Traceability Information Model). The latter artifacts are themselves metamodels which capture what requirements and traceability information is stored and managed by the organization and tooling [32, 55]. A simplified version of the resulting model for this workshop is found in Fig. 4.

In addition to the three BOs, we captured five roles, four MIs, and seven Usage associations, making a model with comparable sizes to the others. We noticed early on that having multiple BOs in one model made the model difficult to scale, as one has to capture each role which interfaces with each BO. Moreover, connections between every role and BO need to be established, as there can be an individual Usage association. In this model, we could have had as many as 15 usage associations (three BOs × five roles); however, we mitigated this by sharing some associations between roles and BOs, when their usage had similar attributes and purposes. We return to discuss this sort of scalability issue and model scoping in Sect. 6.

As part of this iteration, we added a new smell to our list, when the frequency of change for a BO is high, the stability of a usage association for that BO should be low, and vice versa. In our example, the System Requirements had a high frequency of change, but from the perspective of a role like the customer facing role (product manager), the stability was judged to be medium. Unlike the developers, such roles would not carefully follow all the detailed changes with this BO. One of the consequences of such a mismatch in perceived stability could be that related artifacts are not necessarily kept up to date and traced to the BO and that stakeholders might have difficulties relying on (potentially outdated) information. Whether this is identified as an actual issue or not, differences in the frequency of change of a BO vs. the perceived stability can raise interesting discussion points.

We also discussed issues with the prescriptive (descriptive) attribute (i.e., does the BO describe what should be or describe reality). This attribute was often difficult for participating companies to fill, but it also generated good discussion about the purpose of the BO. The modeling brought up issues centered around how to best capture tooling in the model, and we return to this topic in Sect. 6.

4.4 Cycle 4: view types and tool support

The most pressing issues identified in the cycle 3 were partial information overload in larger models due to missing views on the models as well as the lack of a dedicated modeling tool despite the existence of a metamodel. The tooling used previously was general purpose without any dedicated template or support for BOMI.

To address these issues, we elicited needs for views in the BOMI context as part of a Bachelor's thesis [50]. This was done first through interviews with BOMI authors, and then through a series of workshop presentations with live feedback and short surveys with company participants. We received feedback from participants from three participating companies.

The thesis identified a number of key BOMI stakeholders, their concerns, and related BOMI elements from the metamodel. The stakeholders included managers, system engineers, developers, requirements engineers, architects, as well as process/method/tool experts. Identified stakeholder concerns included understanding and managing MIs, understanding coordination aspects, and providing a BOMI overview. From these roles and needs, various metamodel elements were determined to be shown or hidden in the resulting views. BOMI views have been presented in Sect. 3.5.

Beyond the need for the view types, we missed particularly in the workshop with company D in cycle 3 (cf. Sect. 4.3.1) BOMI-specific tool support that provides more adequate possibilities than the general-purpose drawing tools applied until cycle 3. Thus, in order to provide a convenient modeling experience exploiting the BOMI metamodel, to enable the automatic checking of smells, and to transfer the view types mentioned above into practice, we developed in this iteration a BOMI modeling tool [19] that incorporates all of these aspects. We have described this tool support in Sect. 3.6.

5 Related work

A number of related conceptual modeling approaches have been proposed.

Knowledge Management Our work bears similarities to approaches that focus on modeling for knowledge man-

agement, e.g., [1, 46]. Here, the focus is often knowledge creation, distribution, representation, and retrieval. Our approach captures some of these elements in the BOMI metamodel, including the format of the BO, its purpose, and users. However, our focus is less about capturing implicit knowledge through a global strategy and more about understanding the way that diverse organizational islands coordinate knowledge through artifacts.

Other related work uses patterns to detect potential problems in information flows, e.g., consecutive transformations, which are similar to our notion of smells [42]. Our contributions focus less on the flow of information but more on effective coordination, thus our specific smells are quite different compared to [42].

Agent-Orientation Our work bears some similarity to agent-oriented or multi-agent system modeling which emphasizes the rational behavior of individual agents in a system (e.g., [16, 24]). Most of this work has an exchange of resources by agents through some form of dependency. Although agent concepts could be used to capture MI, the islands are more like social groupings emerging due to various drivers, and often do not act together as a sentient and autonomous whole. Similarly, BO could be resource dependencies, but our concept of BO is richer, and we place more emphasis on the means of use and attributes of BO, compared to resources in agent-oriented modeling.

BOMI is in line with the Comakership organizational pattern [7], with our notion of smells fitting with the idea of continuous improvement. However, these patterns focus on inter-organizational coordination, while BOMI covers inter-team coordination, and BOMI does not make use of i* or intentions, with attributes such as "Purpose" in the BO fulfilling this role to a lesser degree.

Communication The work by Oliveira et al. [38] introduces ontologies for collaboration, communication, and cooperation, with several elements and components echoed by our BOMI metamodel. However, their focus is not on supporting diverse groups as with our MI, or on the attributes and specifics of the boundary objects or artifacts. Some of the work which has focused on modeling communication focused on autonomous agents and their protocols (e.g., [8]), while we focus on communication between MIs, always consisting of humans.

Coordination Related work on coordination modeling focuses on coordination between information systems rather than human-oriented MIs [35]. In this view, coordination between systems can be captured via APIs, a type of BO. Previously, benefits and limitations of languages for capturing APIs have been investigated [21], e.g., i* and e^3 value modeling. Although the focus lay more on the use and value of APIs and less on coordination between methodologically diverse groups, BOMI may still be beneficial for API analysis.

Further work is more process-oriented. Wieringa et al. [51] apply e^3 value modeling, process modeling, and physical delivery modeling to support cross-organizational coordination. ActivityFlow focuses on supporting incremental and flexible workflow definitions, allowing for workflow coordination between organizations [31]. BOMI takes a static, rather than process-oriented view, as our partner companies, with an agile mindset, focus less on workflows and more on practices.

Ecosystems Work in ecosystem modeling is also related (e.g., [5, 59]), as our BOMI approach can be said to produce a type of ecosystem model; however, existing ecosystem models focus more on external coordination, where the internal methodologies of a partner are more opaque. Our BOMI models tend to have a mix of internal and external MIs and BOs, often with a particular focus on supporting diversity in internal ways of working.

Collaborative or Participatory Modeling Previous work has focused on experiences and guidelines in collaborative or participatory modeling, where models are created with stakeholders or the subjects of the models [40, 45].

Stirna et al. provide recommendations based on their experiences with participative enterprise modeling, including accessing the organizational context, assessing the problem at hand, assigning roles, acquiring resources and preparation efforts, and conducting the modeling session [45]. In our case, our previous experiences with the companies as part of an ongoing research project focusing on requirements engineering issues in large-scale agile development allowed us an understanding of the organizations' culture and allowed for company resources to be spent in the workshops. We discussed the problem or motivation for the modeling session with each company before proceeding, in fact, this was often discussed as part of the end of the first hour of our workshop, after presenting BOMI and before the modeling session.

Similarly, Renger et al. identified a number of topics and lessons learned with collaborative modeling, including roles, the interactive process, the modeling method, and model quality [40]. In terms of process, we were able to work simultaneously and collaboratively, but the modeling was driven by one of the researchers. This was positive in that BOMI expertise drove the workshops, but negative in that it may have been challenging for participants to modify the model. However, we continually asked for feedback and confirmation of our models. Our framework includes a modeling method for guidance, and we believe our models were of sufficient syntactic and semantic quality to facilitate our goals of discussion and knowledge sharing.

In terms of recommended roles as per [45] and [40], in some cases, our sessions included a modeler and a facilitator/reflector/recorder, but in other cases our online modeling sessions were facilitated by only one member of our team, who served as both the modeler, reflector, facilitator and

recorder. Given the use of electronic modeling via an online meeting with screen sharing, we did not perceive all mentioned roles to be necessary. Regarding guidelines for the modeling session, we deviated from the advice by first offering a training session on BOMI, rather than picking a more generic and understandable modeling notation. However, our experiences with this were positive, as training helped to set the goals and intentions for the modeling session, and helped the participants to think about the concepts before the modeling started.

Agile Modeling A large body of work looks at how model-driven development or engineering could be used as part of an agile or modified agile environment (e.g., [2, 20, 29]). These works focus on the question of how agile practitioners can use modeling as part of their daily design or development work, while still working in an agile way. In our work, we focus on a different level of abstraction, using modeling to guide or improve agile processes, modeling *for* agile development, not modeling *as part of* agile development. For example, in the first line of work, one would model the details of the artifacts as part of the product under development (architecture, design), while BOMI models process-oriented coordination over artifacts, which could include instance of these technical artifacts, e.g., design specifications, but also process-oriented artifacts like requirements specifications, or test plans.

Coordination in Architecture and Design Similarly, the focus of BOMI is on process-oriented artifacts to facilitate coordination in agile development. Other work has looked at technical coordination artifacts to facilitate coordination among design components, for example, creating clean boundaries and coordination from a software architecture viewpoint (e.g., [33]) or the creation of APIs (e.g., [37]). BOMI focuses on process and human coordination, rather than on the technical details of interfaces or coordination. While an API could be modeled as a boundary object in BOMI, the model would focus on what islands and roles use and change the API for what purpose, rather than which software components use the API for architectural coordination.

6 Discussion

We have presented a conceptual model for BOMI, described how we instantiated it together with five large-scale systems development companies, following our presented methodology, and derived example smells over the instances that can be checked with OCL constraints. Here, we discuss a number of issues which arose during our BOMI modeling and provide guidance in order to use BOMI effectively.

Duration Concretely, in the first cycle with companies we have found that the BOMI model allowed us to create initial models with a rather low time effort (20 min of introduction of general concepts plus 30 min of modeling). In the subsequent cycle, we had longer workshops, but found that roughly one hour of modeling was sufficient to get a detailed model. The whole modeling part of the workshop was two hours, but this included a BOMI refresher, breaks, a discussion about smells, a discussion about BOMI, and in a few case a discussion on visualization. Although more detail could be added, after one hour, we could see signs of modeling fatigue, perhaps exacerbated by the use of online meetings.

In the future, a follow-up 30–60 min meeting to fill out any missing details in the model could be useful, after participants have had time to reflect and recharge. This also allows the facilitator to "clean up" the model, shifting around shapes and lines to improved readability, and highlighting any missing fields or element types.

Timing In our workshop design, we split the one-hour training session from the two-hour modeling and discussion session in two out of three of our company workshops. We found this worked well, allowing participants to take a break and digest the content. The company who chose to have both sessions in one three-hour slot voiced that they would also prefer this split in the future.

Roles vs. MIs We found that several instance models included many roles at the start, and the facilitator had to guide the participants toward the addition of MIs. In another case, there were many MIs but few roles. Similar issues were seen with groups and teams (e.g., a collection of roles) vs. MIs. By default, this part of the model may at first look like a standard organization chart. It may seem that the concepts of roles, teams, groups, and MIs overlap such that some can be avoided; however, the idea with MI is to capture the groupings which are divided by some key coordination factors (e.g., methodology, distance, technology, product). Therefore, two groups may be part of the same MI, and have an easier time coordinating, or different roles in the same team may feasible be part of two MIs. Although the MI concept is admittedly more abstract, we believe it is useful to help participants think beyond organizational structures to capture coordination challenges.

BO Selection When designing the BOMI framework, we envisioned that companies would select BOs that were problematic in some way as focal points. However, as our company partners are generally successful, there is a correlation between successful BOs and commonly used BOs. Furthermore, it seemed that some of the companies did not necessarily want to highlight any pain points in their ways of working. They seemed to focus on BOs that were undergoing some change, e.g., new tooling, or for which the company champions felt like there was no common understanding of how the BOs were managed. It is interesting to note that the modeling process can be beneficial even if the area of focus is not particularly problematic.

Capturing Issues When modeling, we quite often felt the need to capture extra notes or issues not supported directly by the metamodel, for example, the presence of bottlenecks in the flow of information, or extra relevant information which did not fit into the attributes. In these cases, we made notes on top of the model. Examples can be found in Fig. 3 "Short lifecycle..." under the BO or "In the same release train" near one of the MIs. Although this works as an unstructured solution, this raises the question of whether or not the model should be extended to support some of this information, or if we should add the concept of "notes" or "issues" explicitly in the language, rather than try to cover everything with our metamodel. Adding notes or issues would allow us to potentially link such notes explicitly to elements in the instance models. However, drawing these notes as first-class elements also brings some overhead and makes the model look more crowded. This is a question we will continue to explore in future iterations, and may be something solved by tooling or additional visualizations.

Participants We often discussed with our industrial contacts whom to invite as BOMI modeling participants. In all cases, we had one-two industrial "champions" or main support contacts. Furthermore, we advised to invite those who work closely with the BO of focus. We had further discussions where our industrial contacts expressed that many more detail-oriented or more junior employees may not be concerned with the sort of discussions or big-picture views supported by BOMI. Thus, it was advised to focus more on middle-management/team leader/technical lead-type roles. It is possible that the models could be created by one group of people and then viewed, or be used as a reference by others, for example new employees, or those with more detail-oriented jobs. Further views could be used to support management discussions, we return to the topic of views later.

Scalability and Scope We have been recommending that our stakeholders pick a particular BO to focus the model on. As part of the last, internal iteration, we drew a BOMI model for the T-Reqs tool that instead focused on three different BOs, attempting to draw the roles and associations for each BO. Even though in this case, many of the roles for the BOs overlapped, and the usage association classes could be reused (two roles used the BO in the same way), we found this model quickly grew in size and was hard to manage. Thus, we continue with our recommendation of focus on one BO at a time. If one wants to explore the relationships between BO, they could draw one model per BO and link the other BOs in the model without exploring the details (attributes, usage, etc.), this was done successfully in a few of our company models.

Tooling vs. BOs Several of our examples were exploring issues with tooling, including a change in tooling for features and exploration of the T-Reqs textual requirements tool. It is tempting to model the tool as a BO, as the tool would certainly support coordination between different MIs who may use the same tool. However, the mapping between the tool and the BO supported by the tool is often not one-to-one. In the case of T-Reqs the tool supported three different BOs which were interrelated (stakeholder requirements, system requirements, TIM/RIM). In the case of the feature-focused model for the workshop with company A in cycle 3, the tooling used to capture features was going through a change, but we found that this actually did not significantly affect the way in which the actual BO was used. Only an attribute related to how up-to-date the BO would be changed in the model. Overall, we think that our original guidance of focusing on BO per model remains reasonable, and is supported by the experience in our cycles.

Graphical Syntax Thus far, we have been drawing BOMI models using a visual boxes-and-lines syntax similar to UML class diagrams. Another possibility would be to develop a domain-specific graphical language with specialized shapes, perhaps the use of intuitive icons, and other graphical conventions for links. In some initial discussions with our industrial partners, we have gotten mixed feedback on the possibility of developing a domain-specific language. One company expressed that they preferred the familiar UML syntax, while another said that they would prefer a more specialized graphical representation. We will continue to explore these possibilities.

BOMI Smells In many cases, the smells on our list did not apply to the models we drew. We believe this does not indicate that the notion or list of smells is not useful, but is instead a reflection of the effective operations of our companies. In most cases, the companies selected core BOs which over which there were only minor or medium issues in coordination. We thus retain the issues of smells as defined in OCL and implemented in our tool, but also encourage the use and consideration of more informal smells, e.g., "can we draw this model?", "does this make sense?", "is the model too complicated? Is the complexity justified?", "do we agree?", or "do key role representatives understand the model?". Such questions are important, but are supported more via informal discussion rather than formal checks on the model. We believe that both types of smells or checks (formal and informal) can be useful in further BOMI use.

BOMI Modeling Tool Our usage of general-purpose drawing tools led to some issues during the specification of particularly more complex BOMI models. Beyond the missing guidance, formality, and homogeneity during the BOMI modeling, we missed in the workshop with company D in cycle 3 (cf. Sect. 4.3.1) the automatic execution of checks for the BOMI smells. Furthermore, we discovered the benefits of having BOMI views for the particular models in the workshop with company E in cycle 3 (cf. Sect. 4.3.3). To address all these aspects in a dedicated tool support for BOMI modeling and to exploit the existing informal BOMI metamodel and

smells, we developed the BOMI modeling tool. It facilitates a guided and uniform modeling experiences with different concrete syntaxes as well as view types, and it enables the automatic checking of BOMI smells. We are currently evaluating the BOMI modeling tool as part of a further Bachelor's thesis.

Effort Although all modeling requires effort, in our case, our company partners did not complain about the amount of effort with the general-purpose drawing tools or the BOMI modeling tool. As the purpose was for understanding coordination, modeling efforts took 1-2 h, and the found insights were considered worthwhile. In terms of smells, as they have been implemented in our dedicated modeling tool and can be checked automatically, the effort to apply the smells is minimal. Of course, smell results should be discussed in context in order to potentially improve coordination, but such efforts can be very useful and are not automatable.

Transferability Although most BOMI activities described in this work were conducted or led by the BOMI creators, some activities were performed independently by students. We also note that Company E reported to us that they used BOMI modeling in an independent internal study to evaluate BOs, identifying missing BOs that were needed.

BOMI as Specific to Agile Although it is likely that BOMI concepts would be useful in contexts using any development process, these days, almost all companies use agile practices in their work, and we have only studied boundary objects in large-scale agile contexts. Agile software development typically comes with bigger challenges when it comes to inter-team coordination. Companies do not define processes that mandate how the coordination between organizational groups works, but instead it is more organic. It is more important to use coordination mechanisms, such as boundary objects, in these agile contexts compared in plan-driven software development. Thus, we focus our work on agile contexts.

Benefits In cycle 2, our participants were positive about the outcome of the session and the initial models allowed us to test our list of initial smells. In the subsequent cycle, we had an explicit discussion about benefits and drawbacks. Participants were generally positive, stating that the modeling exercise was useful, although in most cases, the modeling did not reveal any revelations in terms of communications and coordination. Generally, the process is helpful in better understanding how coordination with BOs occurs, and sometimes spurred interesting discussions. The benefit is more so the process of modeling, understanding, and discussion, and less so the resulting model, although in at least one case, the model was used to facilitate further discussion.

Our company partners are also interested in exploring the synergies between BOMI modeling and parallel work in the same project focusing on collaborative traceability [56], e.g., modeling traceability artifacts as BOs. A current thesis is looking at BOMI use for modeling interaction in software standards, for standard clarification purposes. We believe that our findings show promise for future BOMI use, and particularly look forward to application in cases which are more problematic and less core to participating companies.

Table 7 Guidelines for BOMI modeling

1	Participants: BOMI modeling session should involve participants who are at a middle management level. Those too new or focused, or those too high up may not care about the view provided by BOMI. Inviting at least one representative for key roles in the model is helpful
2	Subject matter: During a first BOMI session, it can be useful to focus on a BO which is not particularly problematic, in order to test out the method and allow for BOMI training. However, avoid BOs that are well understood or not problematic in some way
3	Duration: It is useful to split BOMI training and modeling into two different sessions, giving time to think about the new information. One to two hours for the modeling and discussion session are typically sufficient
4	View: After creating the full model, consider creating a simplified, condensed view for upper-level management or those who only need to read the model (e.g., new employees)
5	Tooling: The tooling used for a BO should not be a BO itself. Focus on the information contained in the tool, and not the tool itself
6	Scope: For readability, focus on one main BO at a time. Further BOs could be explored in different models
7	MI focus: Elicit the MI view. Often, it is easy to think in terms of roles, but also think in terms of the different ways that roles and groups are separate from each other, e.g., by process, distance, etc
8	Smells: Be open to find and discuss issues highlighted by aspects of the BOMI model, even if they are not explicitly identified by a formal smell

6.1 BOMI guidelines

Summarizing our findings, we have created a series of guidelines for applying BOMI. These can be seen as an appendum to our BOMI method as described in Sect. 3.3. Whereas the method is a step-wise guide, these guidelines are general practices to support effective modeling with BOMI. The guidelines are described in Table 7.

6.2 Threats to validity

To improve *internal validity/credibility*, we used an interactive modeling process with open questions, triangulated the experiences of the participating companies, and aimed to provide detailed descriptions in this paper. A cross-company workshop was used to present the intermediate findings and perform member checking with the participants.

A threat to *construct validity* relates to the nature of the domain we model. The concepts of boundary objects and methodological islands can be misunderstood and interpreted in various ways. We intended to provide clear definitions and engaged in a long-term project with the participating companies to ensure a common understanding of the concepts.

Considering *external validity*, we used a sample of five large-scale companies that develop embedded systems. We believe this sample provides valuable insights, but acknowledge we may have different findings with a different sampling approach. We describe the companies' characteristics in this paper to facilitate the assessment of what findings might be transferable to other contexts. In particular, many of the companies had previous or current experience with modeling, although typically at the architecture or feature level, not at the organizational level. The use of BOMI in other companies without modeling experience may have produced less positive results.

With respect to *reliability*, the previously acquired knowledge of the participating companies in the project is a potential threat. As stated before, we have previously collaborated on boundary objects and methodological islands, which will not be the case for other researchers or research contexts. However, the general notation used in this paper is rather straight-forward and comprehensible for other modelers, which facilitates replication. Moreover, we have made the explanatory material and sample models available online.

7 Conclusions

In this paper, we have focused on the challenge of inter-team coordination and knowledge management in large-scale systems development using diverse development practices. While initial empirical studies existed, there has been a lack of systematic modeling approaches that can support practitioners in modeling their current and diverse coordination settings, and analyzing them to identify issues. Our experiences with multiple companies highlighted the need for more systematic and visual approaches to understand communication, coordination and knowledge issues as part of large-scale agile development. To address this issue, we proposed a conceptual model that can be used to model methodological islands (i.e., groups that work with a different methodology than their surrounding organization) and boundary objects between them (i.e., artifacts that can be used to create a common understanding across sites and support inter-team coordination). We presented an initial list of bad smells that can be leveraged to detect issues and devise suitable strategies for inter-team coordination in large-scale development.

Our contributions include a method to guide BOMI modeling, and a series of guidelines based on our experiences to aid in issues such as scope and participant selection, and pre-defined views over instance models to manage complexity. As an additional contribution for further guidance and harmonization in BOMI modeling, we present the BOMI modeling tool. It exploits the BOMI metamodel and smells, incorporates different concrete syntaxes as well as the identified view types, and it enables to automatically check the BOMI models for smells. As part of this work, we were able to collect extensive empirical evidence. We evaluated the conceptual model, method, and smells together with five large industrial companies developing complex systems and present our positive evaluation results.

Overall, we find that modeling coordination with the BOMI framework allows one to systematically describe governance and intended usage of knowledge artifacts in systems engineering. This offers a powerful tool to manage knowledge in scaled agile approaches.

We plan to build onto these findings to improve on our constructive method supporting the analysis of coordination issues and suggesting improvement strategies, as well as mechanisms to continuously assess the effect of these strategies.

Acknowledgements This work was partially supported by the Software Center Project 27 on Requirements Engineering for Large-Scale Agile System Development, the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by Vinnova under grant agreement nr. 2019-02382 as part of the ITEA4 project BUMBLE.

Funding Open access funding provided by University of Gothenburg.

References

1. Ale, M.A., Toledo, C.M., Chiotti, O., Galli, M.R.: A conceptual model and technological support for organizational knowledge management. Sci. Comput. Program. **95**, 73–92 (2014). https://doi.org/10.1016/j.scico.2013.12.012
2. Alfraihi, H., Lano, K.: The integration of agile development and model driven development—a systematic literature review. In: International Conference on Model-Driven Engineering and Software Development (MODELSWARD), vol. 2, pp. 451–458. SciTePress (2017). https://doi.org/10.5220/0006207004510458
3. Arendt, T., Taentzer, G.: UML model smells and model refactorings in early software development phases. Results of the SPES 2020 Project, AP4, Universität Marburg (2010)

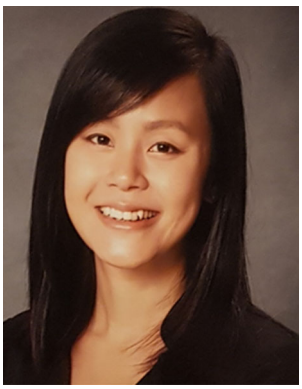
4. Bjarnason, E., Sharp, H.: The role of distances in requirements communication: a case study. Requir. Eng. **22**(1), 1–26 (2017). https://doi.org/10.1007/s00766-015-0233-3
5. Boucharas, V., Jansen, S., Brinkkemper, S.: Formalizing software ecosystem modeling. In: 1st International Workshop on Open Component Ecosystems, pp. 41–50 (2009). https://doi.org/10.1145/1595800.1595807
6. Ciccozzi, F., Tichy, M., Vangheluwe, H., Weyns, D.: Blended modelling: What, why and how. In: 1st International Workshop on Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS). IEEE Press (2021). https://doi.org/10.1109/MODELS-C.2019.00068
7. Colombo, E., Mylopoulos, J.: A multi-perspective framework for organizational patterns. In: International Conference on Conceptual Modeling, LNCS, vol. 4215, pp. 451–467. Springer (2006). https://doi.org/10.1007/11901181_34
8. Dignum, F., Dietz, J., Verharen, E., Weigand, H.: Communication modeling-the language/action perspective. In: 2nd International Workshop on Communication Modeling (1997)
9. Dingsøyr, T., Moe, N.B., Faegri, T.E., Seim, E.A.: Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empir. Softw. Eng. **23**, 490–520 (2018). https://doi.org/10.1007/s10664-017-9524-2
10. Eclipse Foundation: Acceleo Query Language (AQL) (2024). https://www.eclipse.dev/acceleo/documentation/. Last accessed: June 2024
11. Eclipse Foundation: Eclipse Modeling Framework (EMF) (2024). https://www.eclipse.dev/modeling/emf/. Last accessed: June 2024
12. Eclipse Foundation: Graphical Modeling Language Framework Sirius (2024). https://www.eclipse.dev/sirius/. Last accessed: June 2024
13. Eclipse Foundation: Language Engineering Framework Xtext (2024). https://www.eclipse.dev/Xtext/. Last accessed: June 2024
14. Eclipse Foundation: Sirius Specifier Manual (2024). https://eclipse.dev/sirius/doc/specifier/Sirius%20Specifier%20Manual.html. Last accessed: June 2024
15. Goldschmidt, T., Becker, S., Burger, E.: Towards a tool-oriented taxonomy of view-based modelling. In: Modellierung 2012, Lecture Notes in Informatics, vol. P201, pp. 59–74 (2012)
16. Gonçalves, E., Araujo, J., Castro, J.: iStar4RationalAgents: modeling requirements of multi-agent systems with rational agents. In: IInternational Conference on Conceptual Modeling, LNCS, vol. 11788, pp. 558–566. Springer (2019). https://doi.org/10.1007/978-3-030-33223-5_46
17. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**, 75–105 (2004). https://doi.org/10.2307/25148625
18. Holmström, H., Fitzgerald, B., et al.: Agile practices reduce distance in global software development. Inf. Syst. Manag. **23**(3), 7–18 (2006). https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93703.2
19. Holtmann, J.: BOMI Modeling Tool (2024). https://github.com/joerg-holtmann/BOMI-Modeling-Tool. Last accessed: June 2024
20. Holtmann, J., Liebel, G., Steghöfer, J.P.: Processes, methods, and tools in model-based engineering—a qualitative multiple-case study. J. Syst. Softw. (JSS) **210**, 111943 (2024). https://doi.org/10.1016/j.jss.2023.111943
21. Horkoff, J., Lindman, J., et al.: Modeling support for strategic API planning and analysis. In: International Conference of Software Business, Lecture Notes in Business Information Processing, vol. 336, pp. 10–26. Springer (2018). https://doi.org/10.1007/978-3-030-04840-2_2
22. ISO/IEC/IEEE: ISO/IEC/IEEE 42010:2022(E)—IEEE/ISO/IEC International Standard for Software, systems and enterprise–Architecture description (2022). https://doi.org/10.1109/IEEESTD.2022.9938446
23. JGraph Ltd: (2024). https://www.drawio.com/. Last accessed: June 2024
24. Jureta, I., Faulkner, S.: An agent-oriented meta-model for enterprise modelling. In: International Conference on Conceptual Modeling, LNCS, vol. 3770, pp. 151–161. Springer (2005). https://doi.org/10.1007/11568346_17
25. Kasauli, R., Wohlrab, R., Knauss, E., Steghöfer, J.P., Horkoff, J., Maro, S.: Charting coordination needs in large-scale agile organizations with boundary objects and methodological islands. In: International Conference on Software and System Processes (ICSSP), pp. 51–60 (2020). https://doi.org/10.1145/3379177.3388897
26. Knaster, R., Leffingwell, D.: SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering. Addison-Wesley, Boston (2017)
27. Knauss, E., Liebel, G.: T-Reqs (2024). https://gitlab.com/treqs-on-git/treqs-ng. Last accessed: June 2024
28. Knauss, E., Liebel, G., Horkoff, J., Wohlrab, R., Kasauli, R., Lange, F., Gildert, P.: T-Reqs: Tool support for managing requirements in large-scale agile system development. In: 26th IEEE International Requirements Engineering Conference (RE), pp. 502–503. IEEE (2018). https://doi.org/10.1109/RE.2018.00073
29. Kulkarni, V., Barat, S., Ramteerthkar, U.: Early experience with agile methodology in a model-driven approach. In: 14th International Conference on Model Driven Engineering Languages and Systems (MODELS), *LNCS*, vol. 6981, pp. 578–590. Springer (2011). https://doi.org/10.1007/978-3-642-24485-8_42
30. Liebel, G., Knauss, E.: Aspects of modelling requirements in very-large agile systems engineering. J. Syst. Softw. (JSS) **199**, 111628 (2023). https://doi.org/10.1016/j.jss.2023.111628
31. Liu, L., Pu, C.: ActivityFlow: towards incremental specification and flexible coordination of workflow activities. In: International Conference on Conceptual Modeling, LNCS, vol. 1331, pp. 169–182. Springer (1997). https://doi.org/10.1007/3-540-63699-4_14
32. Maro, S., Steghofer, J.P., Knauss, E., Horkof, J., Kasauli, R., Wohlrab, R., Korsgaard, J.L., Wartenberg, F., Strøm, N.J., Alexandersson, R.: Managing traceability information models: Not such a simple task after all? IEEE Softw. **38**(5), 101–109 (2021). https://doi.org/10.1109/MS.2020.3020651
33. Martin, R.C.: Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, Hoboken (2018)
34. Moe, N.B., Šmite, D., Šāblis, A., Börjesson, A.L., Andréasson, P.: Networking in a large-scale distributed agile project. In: 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–8 (2014). https://doi.org/10.1145/2652524.2652584
35. Norrie, M.C., Wunderli, M.: Coordination system modelling. In: International Conference on Conceptual Modeling, LNCS, vol. 881, pp. 474–490. Springer (1994). https://doi.org/10.1007/3-540-58786-1_97
36. Object Management Group (OMG): Object Constraint Language Specification Version 2.4 (2014). https://www.omg.org/spec/OCL/. Last accessed: June 2024
37. Ofoeda, J., Boateng, R., Effah, J.: Application programming interface (API) research: a review of the past to inform the future. Int. J. Enterp. Inf. Syst. (IJEIS) **15**(3), 76–95 (2019). https://doi.org/10.4018/IJEIS.2019070105
38. Oliveira, F.F., Antunes, J.C., Guizzardi, R.S.: Towards a collaboration ontology. In: Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering (2007)
39. Rajkumar, T.M.: Object-oriented software design. In: The Information System Consultant's Handbook—System Analysis and Design, chap. 66. CRC Press (1998)
40. Renger, M., Kolfschoten, G.L., de Vreede, G.J.: Challenges in collaborative modeling: a literature review. In: Advances in Enterprise

Engineering I, vol. 10, pp. 61–77 (2008). https://doi.org/10.1007/978-3-540-68644-6_5
41. Scaled Agile, Inc: Scaled Agile Framework (SAFe) (2024). https://scaledagileframework.com/. Last accessed: June 2024
42. Schneider, K., Lübke, D.: Modeling and improving information flows in the development of large business applications. In: Software Architecture Knowledge Management—Theory and Practice, pp. 175–197. Springer (2009). https://doi.org/10.1007/978-3-642-02374-3_10
43. Sedano, T., Ralph, P., Péraire, C.: The product backlog. In: 41th International Conference on Software Engineering (ICSE'19), pp. 200–211 (2019)
44. Star, S.L., Griesemer, J.R.: Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39. Soc. Stud. Sci. **19**(3), 387–420 (1989)
45. Stirna, J., Persson, A., Sandkuhl, K.: Participative enterprise modeling: experiences and recommendations. In: International Conference on Advanced Information Systems Engineering, pp. 546–560. Springer (2007)
46. Strohmaier, M., Yu, E., Horkoff, J., Aranda, J., Easterbrook, S.: Analyzing knowledge transfer effectiveness—an agent-oriented modeling approach. In: 40th Annual Hawaii International Conference on System Sciences (HICSS'07), pp. 188b–188b. IEEE (2007)
47. Van Emden, E., Moonen, L.: Java quality assurance by detecting code smells. In: 9th Working Conference on Reverse Engineering, pp. 97–106. IEEE (2002). https://doi.org/10.1109/WCRE.2002.1173068
48. Šmite, D., Moe, N.B., Levinta, G., Floryan, M.: Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations. IEEE Softw. **36**(2), 51–57 (2019). https://doi.org/10.1109/MS.2018.2886178
49. Šmite, D., Moe, N.B., Šāblis, A., Wohlin, C.: Software teams and their knowledge networks in large-scale software development. Inf. Softw. Technol. **86**, 71–86 (2017). https://doi.org/10.1016/j.infsof.2017.01.003
50. Vu, V.: BOMI view types: a design science research study. Bachelor's thesis, Chalmers | University of Gothenburg (2022). https://gupea.ub.gu.se/handle/2077/75220
51. Wieringa, R., Pijpers, V., Bodenstaff, L., Gordijn, J.: Value-driven coordination process design using physical delivery models. In: International Conference on Conceptual Modeling, pp. 216–231. Springer (2008)
52. Wohlrab, R.: Modeling boundary objects and methodological Islands: Supplementary Material. In: International Conference on Conceptual Modeling (2020). https://doi.org/10.6084/m9.figshare.12363764.v1
53. Wohlrab, R., Eliasson, U., Pelliccione, P., Heldal, R.: Improving the consistency and usefulness of architecture descriptions: guidelines for architects. In: International Conference on Software Architecture (ICSA), pp. 151–160. IEEE (2019). https://doi.org/10.1109/ICSA.2019.00024
54. Wohlrab, R., Horkoff, J., Kasauli, R., Maro, S., Steghöfer, J.P., Knauss, E.: Modeling and analysis of boundary objects and methodological islands in large-scale systems development. In: International Conference on Conceptual Modeling, LNCS, 12400, pp. 575–589. Springer (2020). https://doi.org/10.1007/978-3-030-62522-1_42
55. Wohlrab, R., Knauss, E., Pelliccione, P.: Why and how to balance alignment and diversity of requirements engineering practices in automotive. J. Syst. Softw. (JSS) **162**, 110516 (2020). https://doi.org/10.1016/j.jss.2019.110516
56. Wohlrab, R., Knauss, E., Steghöfer, J.P., Maro, S., Anjorin, A., Pelliccione, P.: Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requir. Eng. **25**(1), 21–45 (2020). https://doi.org/10.1007/s00766-018-0306-1
57. Wohlrab, R., Pelliccione, P., Knauss, E., Heldal, R.: On interfaces to support agile architecting in automotive: an exploratory case study. In: International Conference on Software Architecture (ICSA), pp. 161–170. IEEE (2019). https://doi.org/10.1109/ICSA.2019.00025
58. Wohlrab, R., Pelliccione, P., Knauss, E., Larsson, M.: Boundary objects and their use in agile systems engineering. J. Softw. Evol. Process **31**(5), e2166 (2019). https://doi.org/10.1002/smr.2166
59. Yu, E., Deng, S.: Understanding software ecosystems: a strategic modeling approach. In: 3rd International Workshop on Software, pp. 65–76 (2011)
60. Zahedi, M., Shahin, M., Babar, M.A.: A systematic review of knowledge sharing challenges and practices in global software development. Int. J. Inf. Manag. **36**(6, Part A), 995–1019 (2016). https://doi.org/10.1016/j.ijinfomgt.2016.06.007
61. Zaitsev, A., Tan, B., Gal, U.: Collaboration amidst volatility: the evolving nature of boundary objects in agile software development. In: European Conference on Information Systems, vol. 24 (2016)

Jörg Holtmann independently conducts research while working in the German railway industry. Formerly, he was a PostDoc at Chalmers University of Technology | University of Gothenburg in Sweden, after being a senior expert at the Fraunhofer IEM in Germany. His research interests center around model-based requirements/systems/software engineering for software-intensive systems.

Jennifer Horkoff is an Associate Professor at the Interaction Design and Software Engineering division in the Computer Science and Engineering Department shared by Chalmers University of Technology and the University of Gothenburg, Sweden. Dr. Horkoff is currently involved in projects investigating requirements engineering for machine learning, supported by the Swedish Research Council and Vinnova). She has been an author or co-author of more than 100 papers in peer-reviewed journals, conferences, or workshops. Jennifer has been a co-program chair of RE, REFSQ, ER and PoEM, and has served on program committees and organizing committees of several international conferences (e.g., ICSE, RE, ER, MODELS, CAiSE), and has been a (co-) organizer of several international workshops.

Rebekka Wohlrab is an assistant professor at Chalmers University of Technology in Sweden and an adjunct faculty member at Carnegie Mellon University. She received her Ph.D. from Chalmers University of Technology. Her research interests include requirements engineering and software architecture, especially in connection with human-on-the-loop autonomous systems. Contact her at wohlrab@chalmers.se.



Victoria Vu is working as a software engineer within the Software and Emerging Tech Department of Semcon AB. She holds a Bachelor of Social Sciences and Bachelor of Education from the University of Western Ontario, Canada, alongside a Bachelor of Software Engineering and Management from the University of Gothenburg, Sweden. She taught in numerous countries across the globe, including Canada, Japan, England and Sweden before embarking on her software engineering career, where her consultancy work has included research in the autonomous automotive and machinery industry, as well as functional safety and cybersecurity of steering systems.

Rashidah Kasauli is a Lecturer at the Department of Networks, Makerere University and holds a PhD in Computer Science and Engineering from Chalmers University of Technology. Her research interests include requirements engineering, large-scales system development and software engineering. As a consultant and researcher, Rashidah has carried out requirements modeling as well as enterprise modeling in different sectors.

Salome Maro is a lecturer in Computer Science and Engineering at the University of Dar es Salaam and current acting Director of the Directorate of ICT at the University. She holds a PhD in Software engineering from the University of Gothenburg, Sweden (2020), has an MSc in Software Engineering from the University of Gothenburg, Sweden (2014) and a BSc in Computer Science from the University of Dar es salaam (2010). Her consultancy work includes system analysis and design and on this. She has worked extensively with Tanzanian government institutions to aid with business process engineering and development of systems. Dr. Maro's research focuses on software engineering and EdTech.

Jan-Philipp Steghöfer is a senior researcher at XITASO and formerly an associate professor at the Interaction Design & Software Engineering Division of Chalmers University of Technology and the University of Gothenburg. His main research interests are software traceability, model-driven engineering, security assurance, and agile requirements engineering.

Eric Knauss is a Professor in Software Engineering at the Department of Computer Science and Engineering, Chalmers and University of Gothenburg, where he is also the Head of the Division of Interaction Design and Software Engineering. His research interests focus on managing requirements and related knowledge in large-scale and distributed software and systems development, in particular with respect to complex, distributed systems. Research topics include Requirements Engineering, AI Engineering, Software Ecosystems, Cross-Organizational Software Development, and Software Development Methods beyond Agility. Eric's research is based on strong collaborations with industry and academic partners, both nationally and internationally. He is involved in the organization and review boards of key journals and top conferences in his field, in which he has also published more than 100 peer-reviewed publications.