Semantic Bridging between Conceptual Modeling Standards and Agile Software Projects Conceptualizations

Cătălina Iulia Floruț
Babeș-Bolyai University
Cluj-Napoca, Romania *catalinaiulia.florut@gmail.com*

Robert Andrei Buchmann
Babeș-Bolyai University
Cluj-Napoca, Romania *robert.buchmann@econ.ubbcluj.ro*

Abstract

Software engineering benefitted from modeling standards (e.g. UML, BPMN), but Agile Software Project Management tends to marginalize most forms of documentation including diagrammatic modeling, focusing instead on the tracking of a project's backlog and related issues. Limited means are available for annotating Jira items with diagrams, however not on a granular and semantically traceable level. Business processes tend to get lost on the way between process analysis (if any) and backlog items; UML design decisions are often disconnected from the issue tracking environment. This paper proposes domain-specific conceptual modeling to obtain a diagrammatic view on a Jira project, motivated by past conceptualizations of the agile paradigm while also offering basic interoperability with Jira to switch between environments and views. The underlying conceptualization extends conceptual modeling languages (BPMN, UML) with an agile project management perspective to enrich contextual traceability of a project's elements while ensuring that data structures handled by Jira can be captured and exposed to Jira if needed. Therefore, concepts underlying the typical software development project management are integrated with established modeling concepts and tailored (with metamodeling means) for the domain-specificity of agile project management. A Design Science approach was pursued to develop a "modeling method" artifact, resulting in a domain-specific modeling tool for software project managers that want to augment agile practices and enrich issue annotation.

Keywords: Agile Project Conceptualization, Metamodeling, Domain-specific Conceptual Modeling, Backlog Traceability, Jira Interoperability

1. Introduction

Since the Agile Manifesto prioritized "working software over comprehensive documentation", issue tracking tools like Jira [3] became widely adopted, while diagrammatic modeling is often seen as overhead documentation. Issue tracking tools are tailored for software production workflows relying on a rudimentary conceptualization – task, story, epic etc. This sacrifices relational traceability and contextual semantics - of why a backlog item or a user story exists, how it connects outwards (to the business context) or inwards (to the project management processes and assets).

It might also be the case that established tools determine how project team members understand agile concepts through the lens of their tool-specific practice, which may be diverging from how they have been conceptualized by the agile literature. For example, the concept of "epic" was introduced [13] as a large user story requiring slicing due to size and complexity, but issue trackers commonly employ it as a level of aggregation (grouping) for reporting [36]; in other contexts, epics are referred to as "containers of initiatives" cf. [38] or even as a distinct type of backlog item (as observed by this paper's authors). The structure of a user story ("as a ...., I want ... because ..."), although it informed the engineering of languages like Gherkin [41] and provide a “conceptual anatomy” for story

visualizers [29], is often ignored in practice as user stories tend to be conflated with features or tasks. Conceptualizations introduced by the literature are flattened or distorted by operationalization and tools – the awareness of this problem, originating in practical development experience of the authors with local IT industry practices motivated the project reported in this paper. Consequently, we turned to agile conceptualization methods in order to build a domain-specific modeling tool that achieves the following:

- It creates an overarching layer of abstraction over Jira (i.e. it is technology-specific) as well as over the agile software development domain (i.e. it is domain-specific);
- It provides a visual management approach of assets that are traditionally handled in Jira, hereby enriched by diagrammatic semantics to contextualize those assets with aspects pertaining to business process management and systems architecture;
- It complements the decomposition view of Scrum and Kanban practices with a business process view (BPMN-based) and a UML-based view (granularly mapped to the components of the user story pattern "as a ..., I want ... because ...");
- It interoperates with Jira in the sense that it can transfer the diagrammatic project structure (visualized as a Scrum or a Kanban board) and its annotations (story points, priorities, responsibilities etc.) to a Jira environment via a CSV export or HTTP requests to a Jira server.

Depending on the user's angle of view, the result may be seen as (a) an extension of a BPMN/UML tool (BEE-UP [32] was used as the starting point since it provides an open source implementation of BPMN and UML), or as (b) a domain-specific modeling tool that encompasses aspects of BPMN and UML - where the "domain" is that of agile project management, with first-class "domain concepts" (epic, story, sprint, Kanban board etc.).

Domain-specific conceptual modeling, compared to what established modeling standards offer (i.e. UML, BPMN), aims for conceptualizations of narrow scope and deeper specialization [20]. When powered by a metamodeling methodology, conceptualizations can be tailored for any purpose or specificity, having their design space shifting accordingly [13]. Specific flavors of agile methodologies support the realization of modeling tool prototypes implementing such tailored conceptualizations [16].

The project was organized as a Design Science Research (DSR) effort, currently in the second iteration of the DSR cycle. An earlier version of the tool (its first DSR iteration) was described as a poster in the demo track of REFSQ 2022 [18]. The current report reframes it as a regular research paper and also points to further technical developments realized during the second iteration: the extension of interoperability to HTTP (in addition to the CSV export/import), the inclusion of a "Kanban board" diagram type (in addition to a Scrum decomposition) and metamodel extensions to enrich semantics.

The remainder of the paper is structured as follows: Section 2 will outline the problem and will give a bird's eye view on the developed artifact, under the Design Science framing. Section 3 will describe the engineering methodology. Section 4 will detail implementation, design decisions and the application methodology. Section 5 will discuss related works. Section 6 will provide evaluative reflection and will indicate limitations to be addressed by future iterations. The paper ends with Conclusions.

2. Problem Awareness and Solution Overview

According to [5], in Design Science a conceptualization is a form of early-stage theorizing and we believe that the inverse is also true – theoretical frameworks can inform conceptualizations, which in turn can be made operational in the form of diagrammatic modeling tools. We're primarily referring to [14] and subsequent efforts (see SAFE [38]). The Agile Manifesto is too abstract and principle-oriented – i.e. of little use in acquiring first-class conceptual citizens that are needed for an ontology or a modeling language.

The origin of this endeavor is twofold: (a) on-going discussions among agile project management practitioners and theorists regarding how key concepts in the field deviate between the original definitions [36], the theoretical frameworks [38] and operational tools like Jira; (b) a requirement to bridge the gap between agile project management tooling and modeling standards. Both points have been identified in the local IT industry where

the authors have been active, via direct work experience (with Jira or FusionForge) and by questioning managers of projects. It is not statistically meaningful enough to reflect on how spread the problem is, but in the spirit of Action Research the following open issues have been picked in at least three cases of local outsourcing companies:

- The agile concepts (particularly "epic" and "user story") are distorted by practitioners to fit oversimplified views or shortcuts taken by a project manager – e.g. the "user story" conflated with "task" or "feature", the "epic" treated simply as a bottom-up grouping container of a manager's priorities (i.e. a top-down slicing would lead to a different project structure);
- A lacking BPM culture causes business analysts to fail in mapping user stories on business processes in a consistent manner. This leads to late change requests and a need to fill gaps that could've been easily identified along an As-Is business process model. This also relates to the prevalence of Jira's project slicing view, which neglects a process-centric view or traceability to stakeholder goals;
- Occasionally diagrammatic models are attached to Jira backlog items with the help of integrated tools such as Draw.io for Jira [2] but only as JPEG files attached to an issue/item (e.g. a class diagram, a database schema), not as granular diagram elements that can be semantically related and traced to backlog items.

The particular conceptual modeling approach proposed in this paper was motivated by:

- The user story description pattern clearly showing a structured conceptualization [29] that can be further mapped on types of diagrams to place story elements in business or architectural context;
- Business process models provide a semantic, contextual "glue" for user stories, supporting the explainability of their dependencies and slicing. We're also employing the main structural diagram types of UML – the deployment/component diagram and the use case diagram, whose elements can be granularly linked to a story or other enabling backlog items;
- It gives the possibility to use BPMN for multiple flow abstractions: for describing processes that are internal to the software development company (richer than the Jira workflows), for describing business processes to be supported by the developed software, for describing the user experience flows on a click/keystroke level of detail (as also advocated by the practice of Robotic Process Automation);
- Sometimes a project manager may come from a BPM background and this should be leveraged rather than replacing the process-oriented view with the typically decompositional Jira experience.

Using a problem statement template recommended by the Design Science literature [45] we therefore structure the problem as follows:

Improve **Agile Software Project Management (ASPM)** *(problem context)*
By treating it with a **domain-specific modeling method** *(artifact)*
That satisfies **interoperability with Jira and semantic integration with established modeling languages (BPMN/UML)** *(requirements)*
In order to achieve a **consolidated, visually manageable, semantically traceable project description that integrates a process view, an architectural view and the traditional decompositional views – Scrum, Kanban** *(goals)*

3. Methodological View

In addition to the generic Design Science Research process [34,45], the nature of the developed artifact imposed a dedicated engineering approach embedded in the DSR cycle. One such approach is Agile Modeling Method Engineering (AMME) which was employed in the past for developing semantically rich mind mapping environments [10], domain-specific simulation environments [9] or as a foundation for semantics-driven engineering methods [8].

AMME is a core methodology for the Open Models Initiative (OMiLAB) community of practice [32] - a network of researchers interested in value capture and value delivery for domain-specific conceptual models. The AMME methodology was crystallized by iteratively

applying it across diverse projects and became recently the key methodological enabler for OMiLAB's Digital Innovation environment [22].

AMME translates agile development principles to the development of diagrammatic modeling tools [44] and makes possible iterative prototyping in alignment with a DSR cycle. For this purpose, it prescribes several phases supported by specific tooling, skill profiles and activities, instantiated for the current project as follows:

- ***Create*** phase: refers to the identification of (modeling) purpose and required depth of specificity – i.e., of the so-called "modeling method requirements" [24] and an early catalog of concepts or properties that are traceable to those modeling requirements. For the artifact hereby reported, the requirements for the proposed conceptualization are (a) to be interoperable with Jira; (b) to granularly contextualize project/backlog components with BPM and a systems architecting perspective; (c) to be diagrammatic in order to support not only the BPMN/UML contextualization, but also visually-driven requirements gathering approaches - e.g. Design Thinking, Mind Mapping, for which AMME-based modeling tools already exist (see [4, 9]). Agile management methods tend to employ visual decompositions of a project – see the "Kanban boards" or studies on the role of visualization in agile [28].
- ***Design*** phase: refers to the design of modeling method building blocks: notation, syntactic rules, semantics (conceptual schemas including relationships and specializations), mechanisms (model processing features). The early stage metamodel realized in the previous phase is refined for the metamodeling platform of choice – AMME typically employs ADOxx [7] for iterative prototyping, which is also the platform used here. For the artifact hereby reported, the design aspects will be discussed in Section 4.
- ***Formalize*** phase: a default set-theoretic formalism reflects the ADOxx modeling constructs [17] and its metamodeling approach. A graph-theoretic formalism was also defined as a foundation for converting any ADOxx-based diagrams to RDF graphs with a metamodel-based RDF schema, see [25], which is also applicable to the underlying graph structure of the hereby discussed tool. For the purposes of this paper, no additional, domain-specific formalism was found relevant, as current focus is placed on the pragmatic value of the artifact.
- ***Develop*** phase: the modeling tool was realized with the help of ADOxx metamodeling features: a metamodel designer, a notation designer (for graphical symbols), the internal scripting language ADOScript (to program the interoperability features). The development effort started from an open source BPMN+UML implementation available in the BEE-UP tool [26,32]. The BEE-UP source was extended in ADOxx [6] and the internal scripting language ADOScript was employed to implement the CSV and HTTP exports conforming the data structures and interfaces of Jira. The RDF converter from a previous OMiLAB project [25] was repurposed to support traceability via SPARQL queries along all relationships captured in the metamodel, including those that connect it to concepts from BPMN and UML. Since ADOxx manipulates a graph-like structure of diagrammatic models with semantic links between them, this enabled certain traceability features tailored for the new conceptualization - in the modeling tool itself and outside the modeling tool, based on RDF graph patterns documented in [12].
- ***Deploy/Validate*** phase: a modeling tool can be deployed in various forms, depending on the ADOxx version (desktop installation, Web service etc.) and feedback from using it informs the reiteration of the entire AMME process based on change requests. In terms of validation, a certain level of quality checking is already enforced by the prototyping platform – e.g. consistency against the metamodel is guaranteed, the look-and-feel and diagrammatic usability are inherited from the ADOxx canvas template. Therefore, the key concern was completeness relative to its purpose/requirements, to be discussed in Section 6.

4. The Proposed Artifact

4.1. Implementation

We started from the BPMN implementation, extended in the sense of specialization (new types of tasks added to the default task taxonomy, new types of informational resources subsuming the data object concept) and in the sense of semantic linking (with constructs from the newly introduced structural part) by following principles of multi-perspective modeling, where structural and behavioral aspects are separated in different types of models [24].

A “project structure” diagram type (in two versions, Scrum and Kanban) was first developed to reflect the typical structures handled by Jira, extended with conceptualizations found in other frameworks – e.g. the conceptual structure of an Epic in SAFe [38].

Furthermore, the project structure conceptualization was semantically linked to the BPMN specialization. Structural and goal-related diagrams of UML have also been linked – particularly concepts in the use case diagram (repurposed as a map of stakeholders linked to user stories pertaining to their user experience or goals), and concepts in the deployment diagram and component diagram (nodes and components linked to the backlog items they represent or are decomposed into).

Fig. 1 depicts these conceptualization layers: on top, the legacy BPMN and UML which can be used by a Business Analyst and an Architect in a traditional way to describe business and architecture context; next, BPMN and UML specializations for the "agile project management" domain, comprising agile-specific taxonomies of tasks, deliverables, roles; on the bottom layer, the project/backlog structure diagram and its interoperability to Jira (via CSV export or HTTP requests). The semantic integration between layers is achieved via semantic hyperlinks which will be shown on metamodel level in Section 4.2. Concrete examples from which the BPMN fragments in Fig. 1 are extracted are expanded in Fig. 2.

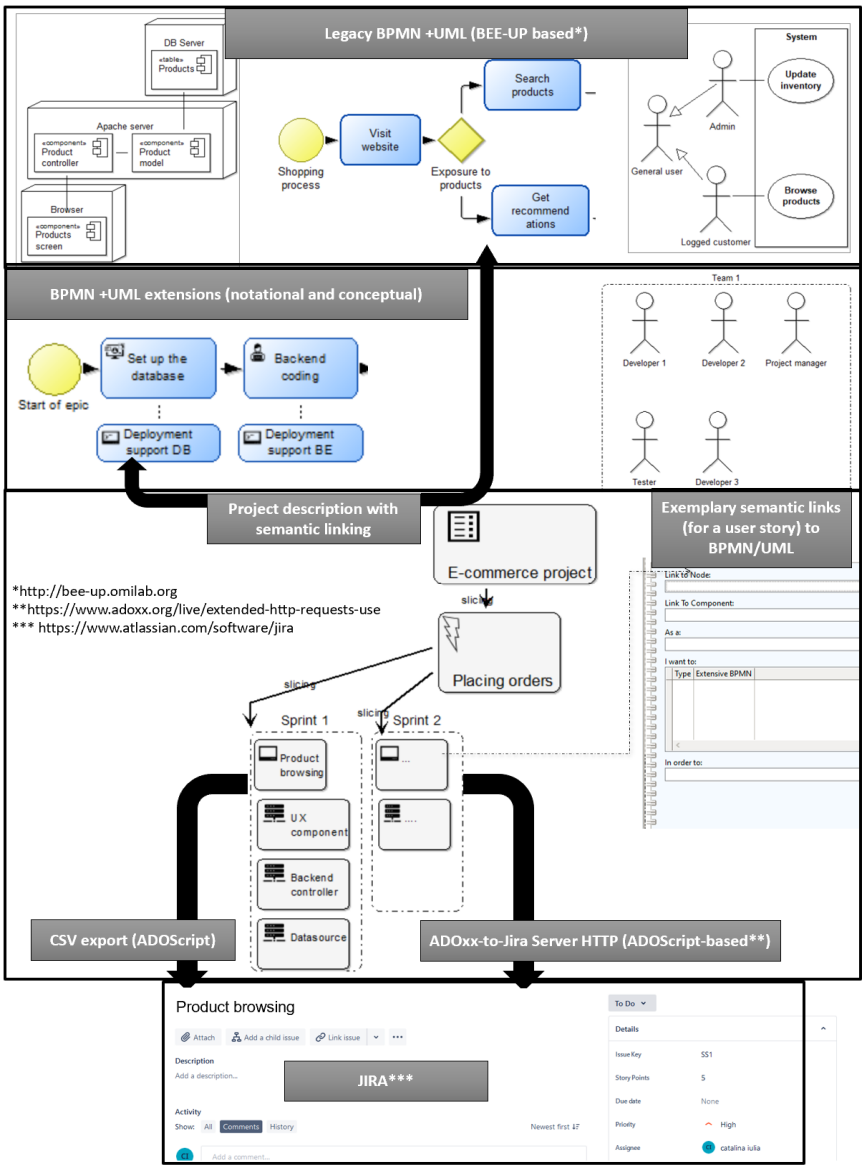

Fig. 1. Layers of conceptualization in the proposed domain-specific modeling tool, with semantic links and Jira interoperability

User stories are motivated by business scenarios and those scenarios are business processes or user experience (UX) processes. Moreover, Business Analysts or Requirements Engineers with

a process thinking mindset typically collect or design business process models which can serve as context or as grounds for explainability of requirements [27]. Therefore, BPMN has been specialized in the three flavors depicted in Fig. 2:

a) traditional BPMN may be used for customer-side business process that motivate the user stories – this is only extended with some semantic links to the project's user stories that a Business Analyst would derive in a step-wise approach along the process flow;
b) the user experience may be granularly mapped on user stories and other backlog items considering a taxonomy of UX actions;
c) the project execution processes may serve the agile manager viewpoint with a Business Process Management perspective, supported by agile-specific taxonomies.

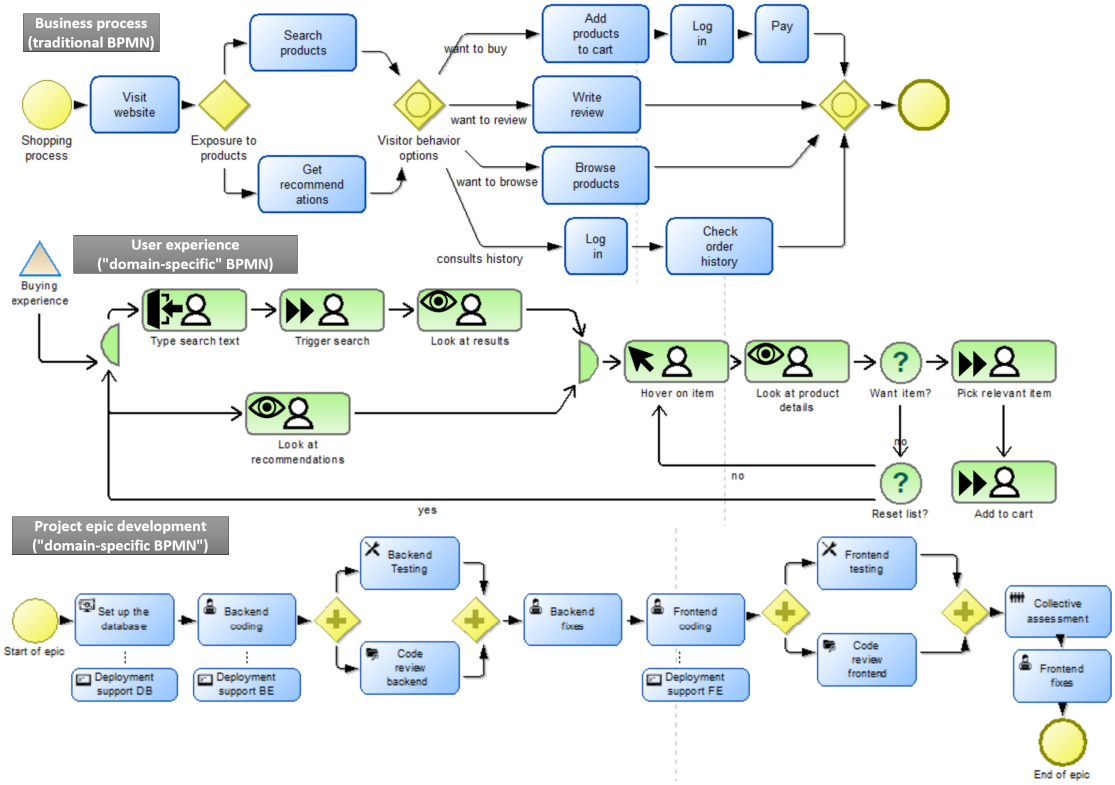

Fig. 2. Three flavours of BPMN linked to project elements: business processes (top), UX processes (middle) and project execution processes (bottom)

4.2. Metamodel

Both the taxonomy of UX actions and of software production tasks are detailed in Fig. 3, for the notational customizations visible in Fig. 2.

The metamodel isolates from BPMN and UML only the parts that are subjected to extensions/specializations. Semantic links (perceived as hyperlinks in the tool) are established between their concepts and the agile concepts (story, epic etc.) present in the project structure diagram. The tool makes the semantic links traceable on a machine-readable level, beyond the dependencies that are typically set up in issue tracking platforms. BPMN tasks (describing customer operations or user experience steps) can be connected through semantic links to the user stories or features required for those tasks. A complex task may be decomposed in granular stories detailing who should do what in support of that business task; or, a user story could be detailed in a BPMN diagram fully depicting the UX flow designed to satisfy the story – the UX action taxonomy prescribes actions such as data input, triggers (independent of modality), dragging, cognition (a user reading a label or taking the time to understand something that informs subsequent actions) or background action (when the user perceives that something is happening without requiring any user action). Similarly, when BPMN is used to describe project execution processes, another flavour of taxonomy and notational customization is applied as already shown in Fig. 3: coding tasks, reviewing tasks, collective tasks, devops tasks

etc. – with their domain-specific information resources.

On the architecting side, UML already provides a rich palette of diagrams that can specialize or decompose a system from an architectural perspective (deployment diagram, component diagram) or from a user goal perspective (stakeholders and their use cases). The proposed approach also ensures that the rich conceptualization of UML becomes a source of annotations of backlog items managed in Jira, to provide architectural context.

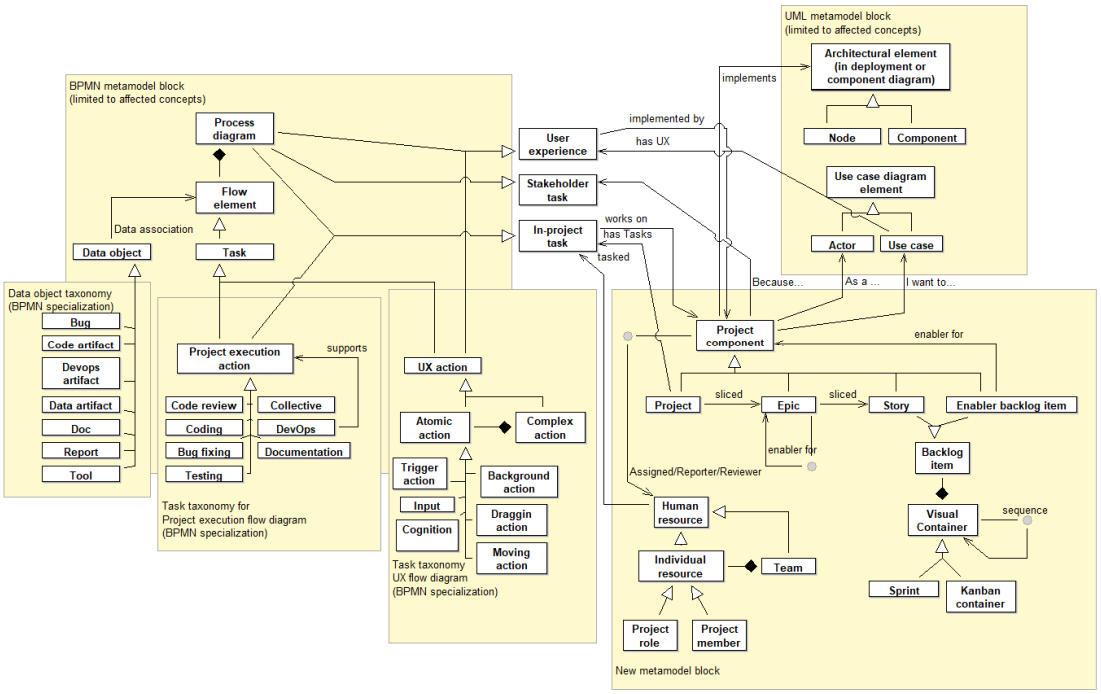

Fig. 3. Metamodel including links to inherited BPMN/UML concepts

4.3. Functionality

Besides the typical diagramming experience which is inherited from the metamodel platform ADOxx, the proposed tool also provides interoperability functionality that was scripted with the help of the platform's native language ADOScript – Fig. 4 shows samples of code: the top fragments deal with the parsing diagram elements and their annotations to build the CSV export; the bottom fragments build HTTP requests as expected by a Jira Server's REST API endpoints for submitting projects and issues in bulk [4].

4.4. Application Method

The proposed modeling method and tool does not intend to replace agile practices, which tend to downplay the role of documentation to the benefit of prioritizing interaction among project stakeholders for better responsiveness. However, some forms of documentation remain relevant even in an agile context – in the form of artifacts that support the stakeholder interaction or communicate design proposals that could not otherwise be specified with implementable granularity. Agile projects tend to rely on some form of "active documentation" [37], and their success is endangered in the absence of process awareness or in the presence of a knowledge gap between the business rationale and the software components being developed [40]. The modeling tool hereby presented provides a number of conceptualization layers that can be selectively employed depending on needs:

- simply for visualization of the project structure [28], see also the Kanban practice, with the ability to transfer the visual structure and its annotations (e.g. priorities) to Jira;
- as a Business Process Management view on development processes, using agile-specific taxonomies of BPMN constructs to capture process knowledge (no interaction to Jira);

- as a multi-perspective mapping of user stories or epics to support either pre-implementation identification of gaps (e.g. business process tasks for which no story was derived) or responsive tracing (e.g. propagation of a change request through UX, architecture, business processes by following semantic links from backlog).

Fig. 5 depicts a full application methodology if all these aspects are considered relevant, with the key stakeholders (lanes) collaborating along an agile project.

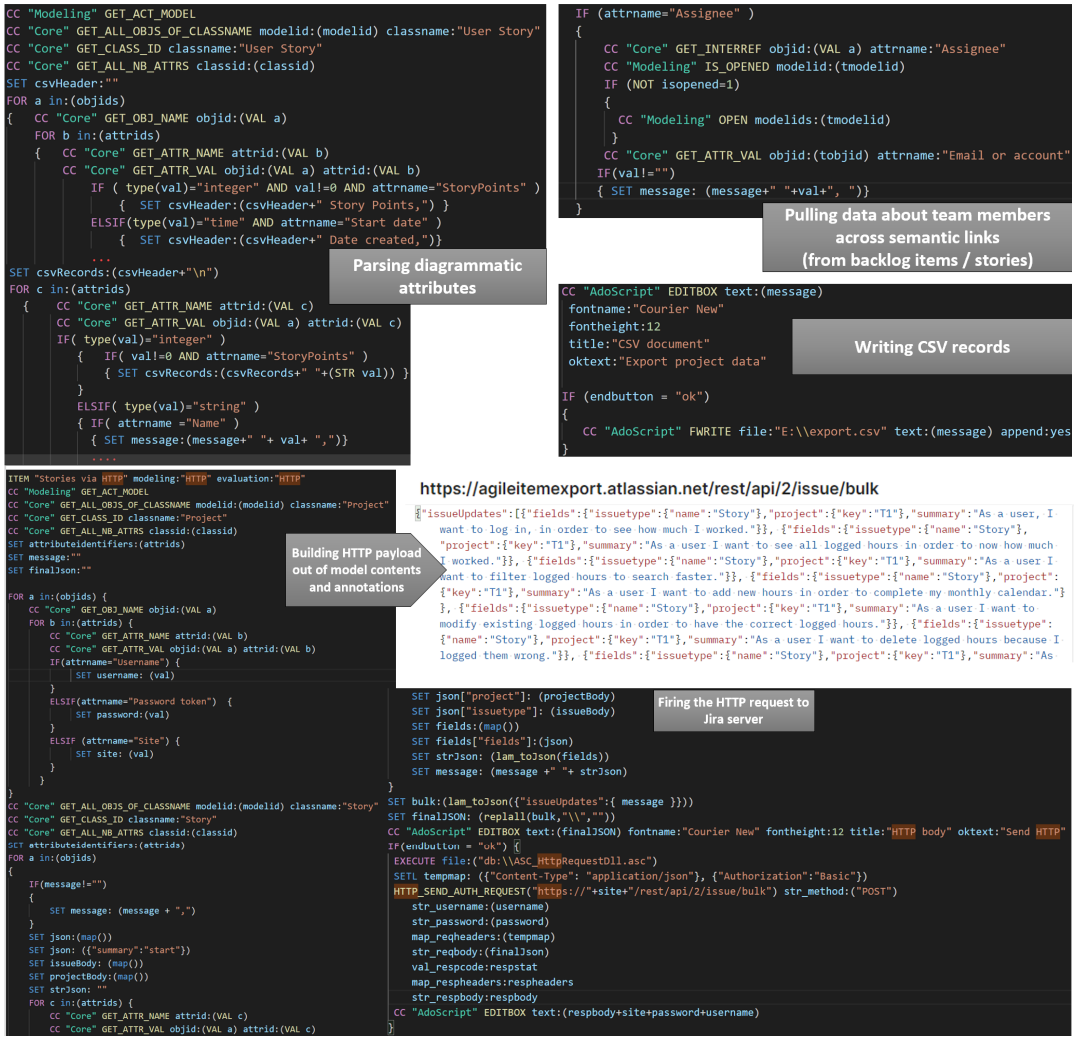

Fig. 4. Scripting samples for Model-to-Jira interoperability – via HTTP (top) and via CSV export (bottom)

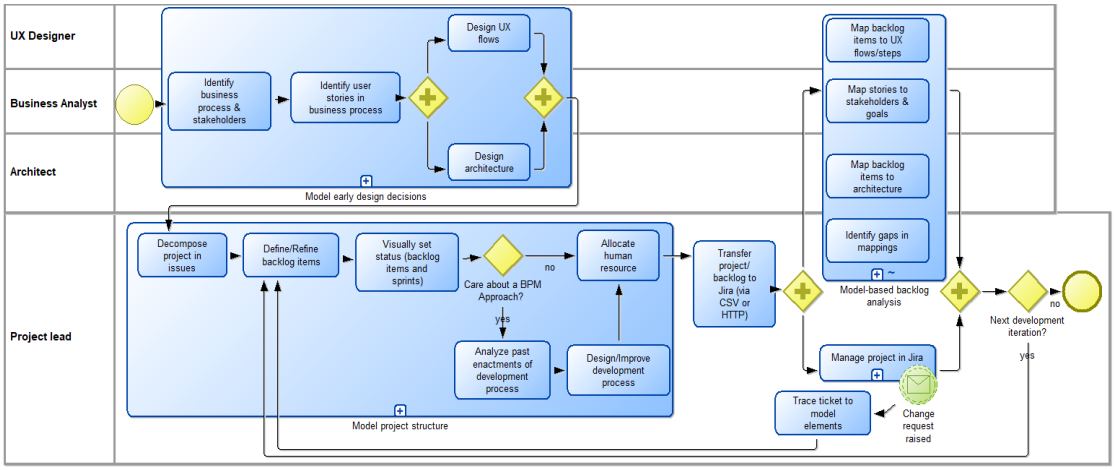

Fig. 5. Application procedure for prospective tool adopters

5. Related Work

Agile software development methodologies still evolve, particularly in terms of required operational support since the agile principles prescribe little in terms of operationalization – in [40] it was highlighted that agile methods are often misaligned with business objectives and a key role in this misalignment are process-related factors and a knowledge gap between project elements and their business context; we aim to bridge this gap and enable a Business Process Management view on top of the knowledge structure imposed by adopting tools like Jira. Although agile principles prioritize software over comprehensive documentation, [37] show that traditional software artifacts come in support of the constant interaction needed among agile project stakeholders. They further propose an approach of "active documentation" that serve agile processes via a reference system and point to conceptual models as being one possible form of such documentation. In [46], a modeling grammar was introduced to conceptualize agile processes, their tasks and dependencies, from a multi-agent perspective - the proposal is a meta-language and is not on an operational level that allows integration with tools such as Jira. The value of conceptual models and visualization for requirements engineering in agile projects was emphasized by [29], where a "visual narrator" generates diagrammatic visualizations from the "conceptual anatomy" of natural language user stories; the idea is gaining traction, as similar ambitions were reported in [21] - it could also be embedded in the tool hereby proposed since the machine-readable conceptual structure was already laid out, however this was not yet in scope as the focus was on the semantic contextualization of user stories and the interoperability with Jira.

Domain-specific modeling languages can play a key role for requirements traceability [42]. The work of [43] proposed a reference framework for software product management that introduces a conceptualization bringing together stakeholders, process areas and product features – however their proposed workbench is still table-based, not unlike traditional issue trackers. A conceptualization of product backlogs was proposed in [39], derived from the application of the constructivist grounded theory.

There exist however several predecessor modeling methods tailored for various aspects of requirements engineering, showing a growing interest in the non-textual representation and contextualization of requirements: reasoning mechanisms over early-stage requirements modelled with i* were demonstrated in [19]; recent proposals complemented i* with behavioral diagrams, leading to the efforts of standardization for a User Requirements Notation [1]. Closer to our approach is the User Story Mapping method [23] – it does not report concrete interoperability features, nor integration with established languages as it focuses on ontology-based annotation of backlog items. Also based on ADOxx technology, [30] reported a digital design thinking method that converts haptic storyboards to structured diagrams.

Requirements' interdependencies have also been recognized for a long time, see [15], but little was done in terms of a metamodeling treatment of those dependencies. Also, interoperability between business process management and project management tools has been tackled from a scheduler perspective [33]. Semantic traceability and contextualization of mobile app requirements was also tackled in [11] in a project-specific manner and not interoperable with Jira.

6. Evaluative Reflection and Limitations

Current evaluation of the artifact is limited to subjective sources and requires longitudinal studies to assess adoptability in productive environments without a risk of disrupting them. At this point, all evaluations have been performed within the work environments of the authors based on qualitative interviews, due to the prioritized criteria of integration and utility. In the DSR cyclic frame, the proposed tool is in its second iteration, aiming for some of the artifact evaluation criteria in [35]:

- *Consistency with technology* ("harnessing existing technology") – the initial implementation only provided a CSV export, but was considered cumbersome compared to the more streamlined HTTP newly introduced (see Fig. 4); a major limitation of the current version and an opportunity for future streamlining is that the

modeling tool is not fully in synch with Jira, it only supports asynchronous interoperability (the transfer in bulk of a project's issues/backlog to Jira);

- *Consistency with organization* – the initial implementation was tailored for Scrum, a Kanban board was added in the second iteration considering the widespread practice;
- *Completeness* relative to traceability requirements, which are satisfied on one hand through the metamodel-driven model query engine [6] and on the other hand through an experimental diagram-to-RDF export feature repurposed from other projects where the authors have been involved [31]. Examples below showcase traceability achieved through semantic queries (SPARQL) navigating across the links prescribed by the metamodel in Fig. 3 (query examples are syntactically simplified for readability):

... retrieving epics to which stories enabled by a particular backlog belong:
*SELECT * WHERE {?x a :EnablerItem; :isInside/^:sliced ?y. ?y a :Epic}*
... retrieving stakeholders whose user stories are enabled by a particular backlog item:
*SELECT * WHERE {?x a :EnablerItem; :enablerFor/:asA ?y. ?y a :Actor}*
... generating via deductive reasoning a "social network" of project workers whose backlog items depend one on the other's:
CONSTRUCT {?x :inputWorkTo ?y} WHERE {?x ^:assignee/:enablerFor/:assignee ?y}

It remains for future iterations to develop demonstrator apps that can leverage such semantic queries, currently only tested in a semantic database workbench.

7. Conclusions

The paper presented a repurposing of domain-specific conceptual modeling for the "agile software project management" domain. Conceptualizations in this domain have been distorted between theorizing and tooling, or between different schools of thought on agility, with the documentation role of diagrammatic modeling reducing the relevance of models in agile practices. The proposed conceptualization is embedded in a modeling method and a modeling tool where the new concepts describing a software project's structure, backlog and production environment are semantically linked to UML architecting concepts and to BPMN concepts. These are also specialized in taxonomies that better reflect the task and resource types involved in software development.

Since this is a Design Science project, future iterations will refine and expand the artifact based on feedback from volunteer users.

References

1. Amyot, D.: Introduction to the User Requirements Notation: learning by example. Computer Networks 42(3), 285-301 (2003)
2. Atlassian Marketplace, Draw.io for Jira, 2022, URL: https://marketplace.atlassian.com/apps/1211413/draw-io-diagrams-for-jira, Accessed March 26, 2022
3. Atlassian, Issue & Project Tracking Software (2022), https://www.atlassian.com/software/jira, Accessed March 26, 2022
4. Atlassian, Jira REST APIs, https://developer.atlassian.com/server/jira/platform/rest-apis/, Accessed March 26, 2022
5. Baskerville, R., Baiyere, A., Shirley, G., Hevner, A., Rossi, M.: Design science research contributions: Finding a balance between artifact and theory. Journal of the Association for Information Systems 19(5), 358-376 (2018)
6. BOC GmbH, Query Language AQL, https://www.adoxx.org/live/adoxx-query-language-aql. Accesed March 26, 2022
7. BOC GmbH, The ADOxx Metamodelling Platform, https://www.adoxx.org/live/home. Accessed March 26, 2022
8. Bork, D., Buchmann, R. A., Hawryszkiewycz, I., Karagiannis, D., Tantouris, N., Walch Michael, M.: Using conceptual modeling to support innovation challenges in smart cities.

In: Proceedings of the 14th IEEE International Conference on Smart City, pp. 1317-1324. IEEE (2017)
9. Buchmann, R. A., Cinpoeru, M., Harkai, A., Karagiannis, D.: Model-Aware Software Engineering - A Knowledge-based Approach to Model-Driven Software Engineering. In: Proceedings of ENASE 2018, pp. 233-240. SCITEPress (2018)
10. Buchmann, R. A., Ghiran, A. M., Osman, C. C., Karagiannis, D.: Streamlining Semantics from Requirements to Implementation Through Agile Mind Mapping Methods. In: Proceedings of REFSQ 2018, pp. 335-351. Springer (2018)
11. Buchmann, R. A., Karagiannis, D.: Agile Modelling Method Engineering: Lessons Learned in the ComVantage Research Project. In: Proceedings of PoEM 2015, pp. 356-373. Springer (2015)
12. Buchmann, R. A., Karagiannis, D.: Pattern-based Transformation of Diagrammatic Conceptual Models for Semantic Enrichment in the Web of Data. In: Proceedings of KES 2015, Procedia Computer Science 60, 150-159 (2015)
13. Buchmann, R. A.: The Purpose-Specificity Framework for Domain-Specific Conceptual Modeling. In Domain-specific conceptual modeling: Concepts, Methods and ADOxx Tools, Springer, pp. 67-92 (2022)
14. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional (2004)
15. Dahlstedt, Å. G., Persson, A.: Requirements Interdependencies: State of the Art and Future Challenges. In: Engineering and Managing Software Requirements, pp. 95-116. Springer (2005)
16. Deme, A., Buchmann, R. A.: A Technology-Specific Modeling Method for Data ETL Processes. In: Proceedings of AMCIS 2021, paper 1128, Association for Information Systems, pp. 67-92 (2021)
17. Fill, H. G., Redmond, T., Karagiannis, D.: Formalizing Meta Models with FDMM: the ADOxx Case. In: Proceedings of ICEIS 2012, LNBIP 141, pp. 429-451. Springer (2012)
18. Floruț, C., Buchmann, R. A.: Modeling Tool for Managing Requirements and Backlogs in Agile Software Development. In: Joint Proceedings of REFSQ-2022 Workshops, Doctoral Symposium, and Posters & Tools Track, CEUR-WS vol. 3122 (2022)
19. Franch, X., López, L., Cares, C., Colomer, D.: The i* Framework for Goal-Oriented Modeling. In: Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 485-506. Springer (2016)
20. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Software & Systems Modeling 13, 941-962 (2012)
21. Gupta, A.: Generation of multiple conceptual models from user stories in agile. In: Joint Proceedings of REFSQ-2019 workshops, doctoral symposium, live studies and poster track, CEUR-WS vol. 2376 (2019)
22. Karagiannis, D., Buchmann R. A., Boucher, X., Cavalieri, S., Florea, A., Kiritsis, D.: OMiLAB: A Smart Innovation Environment for Digital Engineers. In: Proceedings of PRO-VE 2020, pp. 273-282. Springer (2020)
23. Kiritsis, D., Milicic, A., Perdikakis, A.: User Story Mapping-Based Method for Domain Semantic Modeling. In: Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 439-454. Springer (2016)
24. Karagiannis, D., Burzynski, P., Utz, W., Buchmann, R. A.: A metamodeling approach to support the engineering of modeling method requirements. In: Proceedings of RE 2019, pp. 199-210. IEEE CS (2019)
25. Karagiannis, D., Buchmann, R. A.: Linked Open Models – extending Linked Open Data with conceptual model information. Information Systems 56, 174-197. Springer (2016)
26. Karagiannis, D., Buchmann, R. A., Burzynski, P., Reimer, U., Walch, M.: Fundamental Conceptual Modeling Languages in OMiLAB. In: Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 3-30. Springer (2016)
27. Kirikova, M.: Explanatory capability of enterprise models. In: Data & Knowledge Engineering 33(2), 119-136 (2000)
28. Lehtonen, T., Eloranta, V. P., Leppänen, M., Isohanni, E.: Visualizations as a basis for agile software process improvement. In: Proceedings - Asia-Pacific Software Engineering

Conference, APSEC, pp. 495-502. IEEE (2013)
29. Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J. M. E. M., Brinkkemper, S.: Extracting conceptual models from user stories with Visual Narrator, Requirements Engineering 22, 339-358 (2017)
30. Miron, E. T., Muck, C., Karagiannis, D.: Transforming haptic storyboards into diagrammatic models: the Scene2Model tool. In: Proceedings of HICSS 2019, pp. 541-550. Springer (2019)
31. OMiLAB, ADOxx RDF export module, https://code.omilab.org/resources/adoxx-modules/rdf-transformation. Accessed March 26, 2022
32. OMILAB, Bee-Up for Education, https://bee-up.omilab.org/activities/bee-up/. Accessed March 26, 2022
33. Otálora Luna, J. E., Alarcón Aldana, A. C., Callejas, M. C.: Alternative Interoperability Between BPMN and Project Management Tools. In: Communications in Computer and Information Science, pp. 464-474. Springer (2018)
34. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 24, 45-77 (2007)
35. Prat, N., Comyn-Wattiau, I., Akoka, J.: Artifact evaluation in information systems design-science research – a holistic view. In: Proceedings of PACIS 2014, Association for Information Systems, paper 23 (2014)
36. Riley, M.: User stories: A tale of epic confusion, https://www.thoughtworks.com/insights/blog/user-stories-tale-epic-confusion. Accessed March 26, 2022
37. Rubin, E., Rubin, H.: Supporting agile software development through active documentation, Requirements Engineering 16, 117-132 (2010)
38. Scale Agile, Epic-Scaled Agile Framework, https://www.scaledagileframework.com/epic/. Accessed March 26, 2022
39. Sedano, T., Ralph, P., Perairee, C.: The Product Backlog. In: Proceedings of ICSE 2019, pp. 200-211. IEEE (2019)
40. Sithambaram, J., Nasir, M. H. N. M., Ahmad, R.: Issues and challenges impacting the successful management of agile-hybrid projects: A grounded theory approach. International Journal of Project Management 39(5), 474-495 (2021)
41. Smartbear, Gherkin Reference-Cucumber Documentation, https://cucumber.io/docs/gherkin/. Accesed March 26, 2022
42. Taromirad, M., Paige, R. F.: Agile requirements traceability using domain-specific modelling languages. In: Extreme Modeling Workshop, XM 2012 - Satellite Event of the IEEE/ACM 15th International Conference on Model Driven Engineering Languages and Systems, pp. 45-50. IEEE CS (2012)
43. Van De Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: On the creation of a reference framework for software product management: Validation and tool support. In: Proceedings of the First International Workshop on Software Product Management, pp.3-11. IEEE CS (2006)
44. Visic, N., Fill, H. G., Buchmann, R. A., Karagiannis, D.: A domain-specific language for modeling method definition: From requirements to grammar. In: Proceedings of RCIS 2015, pp. 286-297. IEEE CS (2015)
45. Wieringa, R.J.: Design science methodology, Springer, Berlin (2014)
46. Zhang, H., Kishore, R., Sharman, R., Ramesh, R.: Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems, Decision Support Systems 44(1), 266-284 (2007)