# A Metamodel for Enhancing Program Increment (PI) Planning: Towards a Framework for Modeling and Impact Analysis

Flavien Hervé SOMDA[1], Désiré GUEL[2], Kisito Kiswendsida KABORE[3]
{flavien.somda@gmail.com[1], desire.guel@gmail.com[2], kisitokab@gmail.com[3]

Université Joseph KI-ZERBO (U-JKZ), Ouagadougou, Burkina Faso[1,2,3]

**Abstract.** This article introduces a novel approach to addressing challenges in Program Increment (PI) Planning within Agile methodologies and large-scale software development. Our research develops a metamodel and framework to formalize the PI Planning domain, enabling systematic modeling and effective impact analyses.

PI Planning is crucial in Agile software development, helping teams align their efforts towards a common goal. By breaking projects into manageable increments, teams maintain flexibility and adapt to changing requirements. Increment planning promotes regular inspection, adaptation, continuous improvement, and early issue identification. It enhances transparency, collaboration, and stakeholder engagement, leading to successful, customer-focused software development.

Despite its importance, organizations struggle to model the PI Planning process and analyze its impact on project outcomes.

Key contributions include:

- **Metamodel Design:** A detailed metamodel capturing essential structural concepts, relationships, and constraints within the PI Planning domain, providing a foundation for modeling PI Planning processes.
- **Framework Design:** A practical framework leveraging the metamodel, offering multiple perspectives on the PI Planning process. This framework enables stakeholders to create tailored views and conduct impact analyses, supporting informed decision-making.

**Keywords:** Program Increment, Planning, Metamodel, Agile modeling, Modeling Framework

JRI 2023, December 18-20, Ouagadougou, Burkina Faso

DOI 10.4108/eai.18-12-2023.2348135

# 1 Introduction

The Program Increment (PI) Planning process is a critical component of Agile methodologies, especially in large-scale software development projects. Ensuring effective PI Planning is essential for achieving project success. However, organizations often face challenges in comprehensively modeling and analyzing the elements of PI Planning.
In recent years, the importance of formalized models in Agile methodologies has gained significant attention. These models help organizations streamline their processes, enhance collaboration, and improve decision-making.
Our research draws upon a comprehensive literature review of PI Planning, Agile methodologies, and modeling techniques. We build upon existing research findings, emphasizing the need for formalized models to support PI Planning.
This paper contributes to the ongoing discussion on improving Agile methodologies by introducing a metamodel and framework to formalize the specific domain of Program Increment (PI) Planning to enhance organizations' planning processes and enable them for informed decisions regarding software development initiatives. We propose a structured approach to modeling PI Planning artifacts, offering various views to support different stakeholders and facilitating impact analysis. Through a sample of conctrete traceability or dependency problems, we demonstrate the practical benefits of our approach in real-world scenarios.

In summary, we propose an approach that automatically derives analyses from modeling artifacts resulting from PI events, facilitating decision-making processes, which current tools do not enable.

The structure of this paper is as follows. We will initiate with a comprehensive literature review, examining Agile methodologies' contributions to improving software development, as well as reviewing program increment planning and model-driven engineering. Afterward, we will explore current Program Increment (PI) planning practices. Following this, we will clarify the importance of adopting a formal model-based approach to enhance efficiency in PI planning and introduce a metamodel crafted to support this objective. Finally, we will demonstrate how our proposal serves as the foundation for a framework that facilitates advanced impact analysis through diverse representations organized within specific viewpoints.

# 2 Literature Review

In this section, we provide a comprehensive review of the existing literature pertaining to Agile methodologies, Program Increment (PI) Planning, and modeling techniques. By examining prior research and publications, we establish the context and foundation upon which our metamodel and framework are built. The literature review is organized into key thematic areas:

## 2.1 Agile Method in support to successful software development

The utilization of agile approaches in software development has been on the rise ever since the Agile Manifesto was published in 2001 [1]. Agile Methods have been used to build High-Performance Teams [2], i-e, teams that emerge when individuals within the group depend on one another, align

their actions with a shared vision, foster open communication to facilitate the growth of their endeavors, cultivate trust, and embrace collective leadership, thus fostering innovation through the unique contributions of each team member. As an effect, agile methods have contributed to lower software development failures and improve quality [3] [4]. The involvement of Customer decreases the chance of software rejection in the last phases [5].

### 2.2 PI Planning in the literature

Despite the wealth of research in the field of Agile methodologies and software development, providing a rich source of information and insights, a notable gap emerges upon closer examination of the literature: the limited attention given to the intricate domain of Program Increment Planning (PI). Surprisingly, our extensive literature review yielded scant or, in some cases, no mention of this critical aspect of Agile development, which is increasingly gaining prominence in large-scale software development environments.

This conspicuous absence not only underscores the lack of comprehensive understanding and documentation but also reveals a significant research opportunity. The unique challenges and intricacies associated with Program Increment Planning beg for a deeper exploration. As organizations increasingly embrace Agile practices, especially at scale, there is a pressing need for empirical studies, frameworks, and methodologies tailored to the specific nuances of PI planning.

In light of this gap, our study aims to pave the way for a more informed and evidence-based approach to Program Increment Planning. By delving into this underrepresented domain, we hope to shed light on its critical role in Agile software development and uncover new avenues for improving efficiency, collaboration, and decision-making within large-scale Agile contexts. Our research thus presents a timely and essential contribution, offering fresh perspectives and insights into an area that remains largely uncharted in the current body of literature.

### 2.3 Model Driven Engineering

Previous research has extensively demonstrated the benefits of Model Driven Engineering for software and system development. Model-Driven Engineering (MDE) represents a software engineering approach that actively employs models for tasks such as system specification, testing, simulation, verification, validation, analysis, and maintenance, among various other activities [6][7]. The array of advantages offered by Model-Driven Engineering encompasses the following:

- **Abstraction and Simplification:** Metamodeling allows the creation of high-level, abstract representations of complex systems and processes [8]. This abstraction simplifies the modeling process and makes it easier to understand and communicate about complex systems.
- **Consistency and Reusability:** Metamodels enforce consistency by defining a set of rules and constraints for models created based on them. This ensures that models conform to established standards and best practices [9]. Additionally, metamodels promote reusability, as they can serve as templates for creating similar models in different contexts.

- **Automation:** MDE enables automation through code generation [8] [10]. By defining models and their relationships in a metamodel, you can automatically generate code, documentation, or other artifacts. This reduces the likelihood of errors and speeds up development.
- **Traceability:** MDE enables traceability [11] by establishing relationships between different elements in models. This traceability helps in understanding the impact of changes and in managing complex systems with multiple dependencies.
- **Visualization and Documentation:** Metamodels often come with visualization tools that help developers and stakeholders better understand and visualize complex systems. Additionally, the structured nature of metamodels supports automatic documentation generation [10].
- **Domain-Specific Modeling:** Metamodeling allows the creation of domain-specific modeling languages [8] tailored to organization's needs. This enables developers to work at a higher level of abstraction, focusing on domain-specific concepts rather than low-level technical details.

## 3 Current Practices in PI Planning Event

While the specific sequence of activities within a PI Planning Event may naturally differ from one organization to another, informal discussions and participation in a few Program Increment Planning events have revealed that certain key activities are consistently present in most PI Planning events:

1. **Business Context and Overview:**
    - Leadership provides a high-level overview of the business context, market dynamics, and strategic themes.
    - The organization's vision, mission, and goals are discussed to ensure alignment.
2. **Pre-Planning:**
    - Teams and individuals have already conducted pre-planning activities to review their backlogs and prepare for the PI Planning event.
3. **PI Objectives and Team Breakouts:**
    - PI Objectives are defined, outlining the specific outcomes and goals for the upcoming Program Increment.
    - Teams break out into separate sessions, often organized by Agile Release Trains (ARTs), to create their plans.
    - Teams collaborate to select and commit to the features, stories, and work items they will complete during the PI.
4. **Dependencies and Risks:**

- Teams identify and document dependencies between their work items and other teams' work. A visual example of this phase is demonstrated in the Fig 1 figure, which highlights the interdependencies among various team activities.
- Risks and impediments are raised and discussed, with action plans to mitigate them.

5. **Consolidation:**
   - Teams finalize their PI plans, ensuring that all dependencies are documented and understood.
   - Adjustments are made as needed to accommodate any changes that arose during the event.
6. **Confidence Vote:**
   - After the breakouts and discussions, teams often conduct a confidence vote to assess their collective confidence in their PI plan. This is a way to gauge the team's alignment and understanding of the plan.
7. **Closing and Commitment:**
   - The PI Planning event concludes with a commitment to the plan. Teams express their commitment to achieving the PI objectives.
   - Key takeaways and action items are summarized, and the event is officially closed.

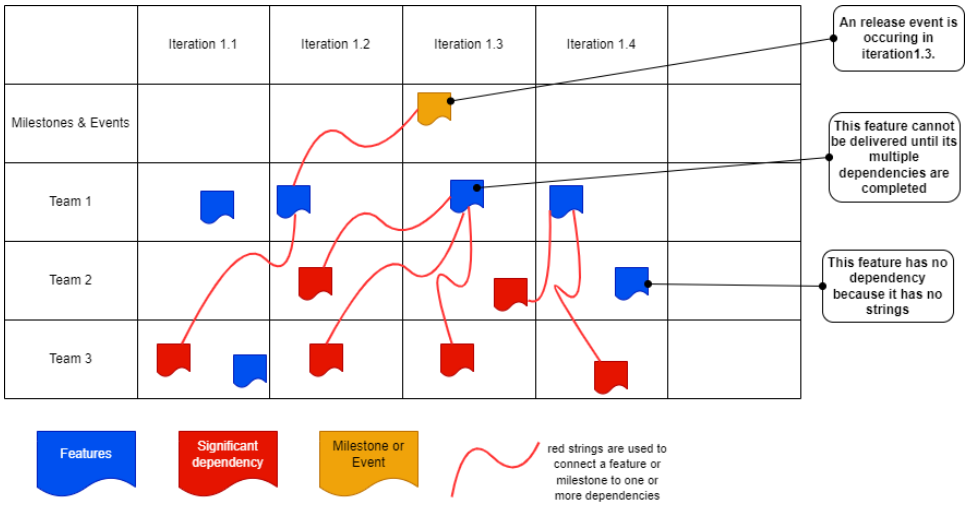


**Fig. 1.** Sample artefact illustrating dependencies in PI Planning

## 4 Why employing a formal model-based Tool to steer PI Planning events

PI Planning, a cornerstone practice in Agile at scale frameworks such as the Scaled Agile Framework (SAFe) [12], has seen a significant shift in recent years with the adoption of digital collaborative tools like Mural or klaxoon. While these tools offer flexibility and virtual collaboration, they also present unique challenges when compared to traditional, formalized planning methods. One of the primary challenges lies in the absence of a formal model or framework to guide the planning process. Here are some key difficulties that organizations may encounter:

- **Lack of Structured Guidance:** Traditional PI Planning often relies on structured agendas and predefined templates to ensure consistency and alignment. Collaborative tools like Mural, while versatile, may lack this level of structured guidance. As a result, teams may struggle to maintain a consistent and standardized planning process.
- **Dependency Management:** PI Planning involves identifying and managing dependencies between teams and work items. Without a formal model, dependency tracking can become more complex and error-prone in collaborative tools. Teams may find it challenging to visualize and address intricate interdependencies effectively.
- **Visibility and Traceability:** In a formal PI Planning model, there is often built-in visibility into the progress of work items and their alignment with strategic objectives. Collaborative tools might not provide the same level of traceability, making it harder to assess whether the PI plan is on track to meet its objectives.
- **Alignment Challenges:** Achieving alignment across teams and ARTs (Agile Release Trains) is a core goal of PI Planning. In the absence of a formal model, it can be more challenging to ensure that all teams are aligned with the same priorities and objectives, potentially leading to misalignment.
- **Documentation and Reporting:** Formal models often support automatic documentation and reporting, which can be lacking in collaborative tools. Generating accurate and up-to-date documentation may require manual effort.
- **Integration with Other Tools:** Collaborative tools need to integrate seamlessly with other software used in the organization's Agile process. Without a formal model to guide integration, ensuring data consistency and accuracy can be challenging

Our objective in furnishing a metamodel for PI Planning is to establish a foundational framework upon which it becomes feasible to construct more organized and cohesive layers, facilitating the planning methodology and housing the data it generates. As an illustration, the metamodel could facilitate the creation of perspectives and representations, such as tables and diagrams, which can be employed to model planning results and automatically identify discrepancies between Agile teams' capacities and work items allocated to them during a specific iteration.

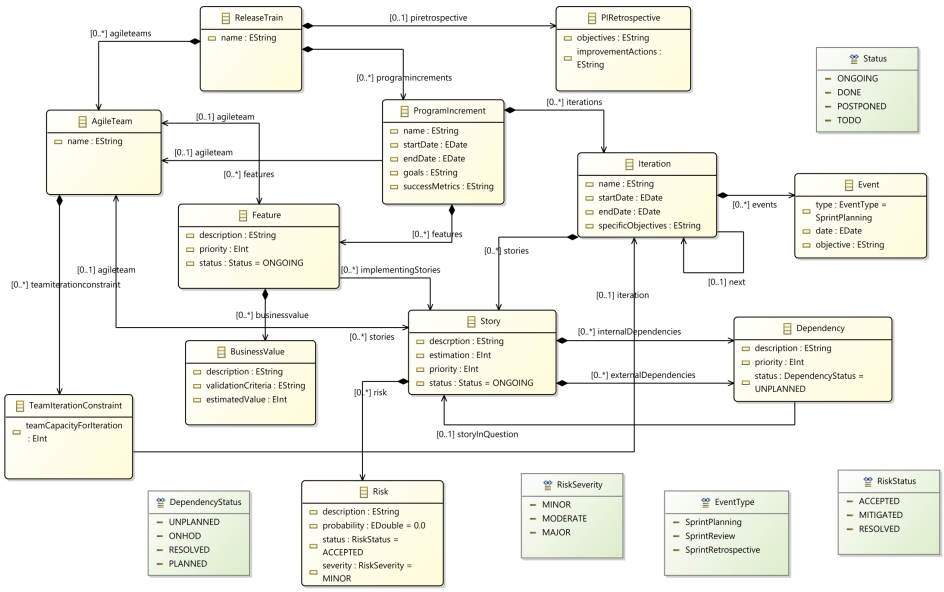


**Fig. 2.** A Metamodel for PI Planning (implemented using Eclipse EMF)

## 5 Proposing a metamodel to enhance Program Increment Planning

Through the creation of a Domain-Specific Model (DSL) for PI Planning, the metamodel proposed in Fig. 2 intentionally eliminates the possibility of unplanned structural relationships between instances, effectively guarding against human errors. Furthermore, additional constraints can be incorporated into the model using OCL to provide an extra layer of prevention against context sensitive mistakes [13]. For instance, the following OCL constraints can serve as illustrations. They illustrate how OCL can enhance clarity regarding what is anticipated within an internal dependency relationship, distinct from an external dependency relationship in the context of a specific story. In essence, an external dependency with respect to a particular story pertains to a dependency linked to a story encompassed by a different agile team's scope.

```
context Story
inv:  self.internalDependencies.storyInQuestion->forAll(s|
 s.agileteam = self.agileteam)


context Story
inv:  self.externalDependencies.storyInQuestion->forAll(s|
 s.agileteam <> self.agileteam)
```

Let's consider another sample OCL example to demonstrate how powerfull can a tool backed by a

formal model be with regard to classic collaborative tools.

```
context TeamIterationConstraint
inv:  self.teamCapacityForIteration >=
 self.iteration.stories.estimation->sum()
```

This OCL constraint ensures that no instantiated model can be considered valid if the capacity of a specific agile team for a given iteration is less than the cumulative estimations of the stories assigned to that team for that iteration. Traditional PI Planning tools do not include mechanisms to deter team members from allocating stories with a cumulative estimation exceeding their capacity. As a result, planning team members must exercise caution and repeatedly assess the alignment of their plan with capacity requirements manually. Given the dynamic nature of the planning process, where the planning board frequently undergoes changes, this verification can become an unnecessary burden. Using a formal model, this cumulative estimation can be automatically calculated by the model, and the diagramming board can incorporate a signaling mechanism to indicate misalignment if it occurs.

## 6 Towards an MDE-Powered Framework Tool for Modeling PI Planning Artifacts

Program Increment (PI) planning involves orchestrating work items for multiple teams with the ultimate goal of achieving a shared outcome. The planning process generates various representations to address specific areas of concern. For instance, one representation may emphasize the dependencies between work items to highlight the chosen strategy for managing them, while another representation might concentrate on the identified risks to underscore the mitigation measures taken to address them.Moreover, since various user profiles participate in the process and may exhibit varying preferences for different representations, the concepts and capabilities of viewpoints and views offered by Model-Driven Engineering (MDE) align well with this diversity.A tool such as Eclipse Sirius is well-suited for implementing diverse viewpoints and views to cater to the distinct requirements of planning stakeholders as represented in Fig 3. Each viewpoint corresponds to a defined collection of representations, which may include diagrams, tables, matrices, or trees. These representations can be customized and expanded as needed. To achieve this, Eclipse provides model handling capabilities with EMF as well as graphical editing capabilities with GMF [14].
To make it concrete let us demonstrate how well-designed and thought viewpoints and views can enhance efficiency in PI management with out-of-the-box traceability and impact analyses. This is made possible because of the capability available for model querying using OCL for example [15]. Table 1 summarizes PI concerns that can be efficiently handled without human effort. This is possible because these representations are backed by a formal model that allows transitive computations.

Let's take, for instance, the entry in Table 1, which pertains to Dependency Tracking. Consider calculating, for a given agile team within the Program Increment (PI), how it relies on all other teams. Referring to the metamodel, it's evident that dependencies between distinct agile teams are not explicitly stated and must be inferred from the stories the teams are involved in, as well as the dependencies that impact those stories. The subsequent OCL expression illustrates how this

**Table 1:** Overview of scenarios where traceability and impact analyses can be managed using views supported by formal models

| Type of problem | Problem or case description | Benefit |
|---|---|---|
| **Feature to User Story Traceability** | Ensure that each Feature is traceable to one or more User Stories. | Helps ensure that Features are broken down into actionable User Stories, facilitating development and testing. |
| **Dependency Tracking** | Track and trace Dependencies between Features, User Stories, or Teams. | Helps in identifying critical dependencies and their impact on planning, ensuring that dependencies are managed effectively. |
| **Risk Management** | Trace Risks to Features, User Stories, or Teams and monitor their mitigation progress | Enables proactive risk management and ensures that potential disruptions are addressed |
| **Team Capacity and Commitment** | Ensure that User Stories are assigned to Teams, considering their capacity and commitment | Facilitates balanced work distribution among Teams and supports commitment-based planning |
| **Change impact analysis** | A change request is received for a specific User Story | Determine the impact of the change on related Features, other User Stories, and the overall PI Plan |
| **Resource Availability Change Analysis** | One of the Teams experiences a change in resource availability. | Assess how the change affects the Team's capacity and the distribution of work items |
| **Dependency Resolution Analysis** | A critical dependency is resolved ahead of schedule | Evaluate the positive impact on Features, User Stories, and Teams, potentially accelerating delivery |

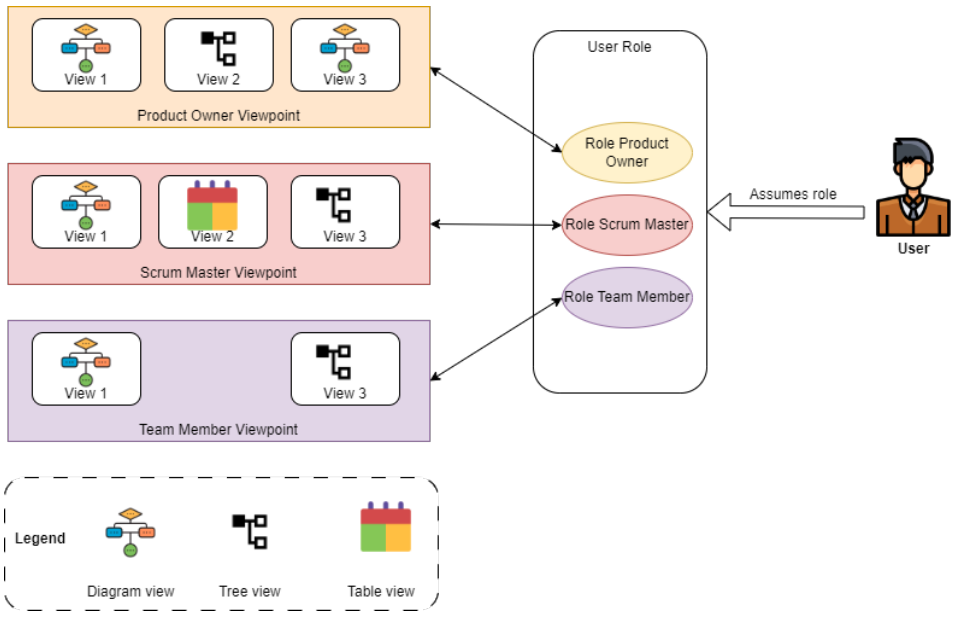


**Fig. 3.** An illustration of user interaction with Viewpoints and Views

collection can be computed to provide data for a specialized view presentation.

```
context AgileTeam
def teamsWeDependOn =
 self.stories.externalDependencies.storyInQuestion.agileteam
 ->asSet()
```

The process of mapping that connects a user to particular viewpoints and views typically considers factors such as the user's role, level of expertise, preferences, as well as their goals and objectives.

## 7 Conclusion and Future Works

From our experience thus far, it can be observed that effectively combining and integrating various Model-Driven Engineering (MDE) techniques has proven to deliver advantages to certain aspects of Agile methodologies, particularly in the context of Program Increment (PI) Planning. As illustrated in this paper, the utilization of Model-Driven Engineering (MDE) yields significant advantages, primarily due to the formalized models that serve as a fundamental basis for the planning process. This foundation allows for computational capabilities on model elements derived from the planning methodology. Building upon this foundational basis, additional functionalities can be developed, such as creating representations and organizing them into various viewpoints. These

resources can then be used by team members and managers during both the Program Increment planning event and its execution.
Numerous research prospects stem from this foundational proposition, beginning with the examination of how the metamodel and representations influence team collaboration and communication throughout the Program Increment (PI) planning and implementation phases. Additionally, there is potential for developing model-based tools to facilitate remote and distributed PI planning processes. Other opportunities encompass:

- Expanding the metamodel to encompass the incorporation of DevOps and continuous delivery practices along with exploring methodologies for aligning PI Plannings with deployment pipelines.
- Develop risk management models that use the metamodel to assess and prioritize risks during PI planning and develop risk management models that use the metamodel to assess and prioritize risks during PI planning

## References


[1] Silva CC, Goldman A. Agile Methods Adoption on Software Development -a Pilot Review. In: Proceedings of the 2014 Agile Conference. IEEE Computer Society; 2014. .

[2] Estácio B, Prikladnicki R, Morá M, Notari G, Caroli P, Olchik A. Software Kaizen: Using Agile to Form High-Perfomance Software Development Teams. In: Proceedings of the 2014 Agile Conference. IEEE Computer Society; 2014. .

[3] Jain P, Sharma A, Ahuja L. The Impact of Agile Software Development Process on the Quality of Software Product. In: 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). IEEE Computer Society; 2018. .

[4] Singh M, Chauhan N, Popli R. A Framework For Transitioning Of Traditional Software Development Method To Distributed Agile Software Development. In: 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE Computer Society; 2019. .

[5] Avasthi A, Mishra G. A New Framework for the Agile Software Development Method. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE Computer Society; 2018. .

[6] de Lara J, Guerra E, Cuadrado JS. A posteriori typing for model-driven engineering. In: 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE Computer Society; 2015. .

[7] Dalibor M, Jansen N, Rumpe B, Wachtmeister L, Wortmann A. Model-Driven Systems Engineering for Virtual Product Design. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE Computer Society; 2019. .

[8] Trask B, Roman A. Leveraging Model Driven Engineering in Software Product Line Architectures. In: 2011 15th International Software Product Line Conference. Munich, Germany; 2011. .

[9] Jossic A, del Fabro MD, Lerat JP, Bezivin J, Jouault F. Model Integration with Model Weaving: a Case Study in System Architecture. In: 2007 International Conference on Systems Engineering and Modeling. Haifa, Israel; 2007. .

[10] Alvarez ML, Sarachaga I, Burgos A, Estévez E, Marcos M. A Methodological Approach to Model-Driven Design and Development of Automation Systems. IEEE Transactions on Automation Science and Engineering. 2018;15(1):67-79.

[11] Sannier N, Baudry B. Toward multilevel textual requirements traceability using model-driven engineering and information retrieval. In: 2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE). Munich, Germany; 2012. .

[12] Ebert C, Paasivaara M. Scaling Agile. IEEE Software. 2017;34(6):98-103.

[13] Hassam K, Sadou S, Gloahec VL, Fleurquin R. Assistance System for OCL Constraints Adaptation during Metamodel Evolution. In: 2011 15th European Conference on Software Maintenance and Reengineering. IEEE Computer Society; 2011. .

[14] Bernardi ML, Lucca GAD, Distante D. A model-driven approach for the fast prototyping of web applications. In: 2011 13th IEEE International Symposium on Web Systems Evolution (WSE). IEEE Computer Society; 2021. .

[15] Iovino L, Rutle A, Pierantonio A, Rocco JD. Query-Based Impact Analysis of Metamodel Evolutions. In: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Kallithea, Greece; 2019. .