A Tool for Modeling and Tailoring Hybrid Software Processes

Andrés Wallberg[1], Daniel González[2], Luis Silvestre[1][a] and María Cecilia Bastarrica[2][b]
[1]*Computer Science Department, Universidad de Talca, Chile*
[2]*Computer Science Department, University of Chile, Chile*

Keywords: Hybrid Software Process, Model-Based Tool, Process Evaluation.

Abstract: Hybrid software processes that combine agile and traditional practices are currently the most frequently used in industry. Most of the time, development is addressed with agile practices while management activities apply more traditional methods. However, the best combination of practices does not only depend on project attributes like project of team size, but also the characteristics that need to be emphasized, e.g. time to market or early value added. DynaTail has been proposed as a method that combines hybrid process tailoring and evaluation according to an intended characteristic to be optimized. It was evaluation in industry and, although it was well received, they highlighted the need for a supporting tool so that software developers only need to deal with elements of their processes and not technicalities of the method. In this paper we present DynaTool, a model-based tool to support the DynaTail method. We formalize it and illustrate its application by replicating the same case. We found that DynaTool can fully support DynaTail. Nevertheless, we still need to go back to industry to confirm its potential adoption.

1 INTRODUCTION

A software process is defined as a combination of roles, activities and work products (Humphrey, 1988). Processes have been valued by software companies as a means for managing development in an organized manner so that it is possible to plan, schedule and provision software projects (Münch et al., 2012). However, a single process does not fit all kinds of projects, even within the same organization. For example, the expertise required from developers highly depends on the complexity of the product (Clarke and O'Connor, 2012). Similarly, a small project could skip building several work products. The activity of adjusting the company's process to the particular characteristics of the project being addressed is called tailoring.

Agile methods propose a series of practices that software development teams adopt and adapt to the project at hand. These methods focus on people involved in software development, are specially appropriate to promote productivity in projects with high uncertainty but they do not provide strong support for project management (Raharjo and Purwandari, 2020). Therefore, most companies follow hybrid processes, i.e., a combination of agile and traditional practices. Large companies used to define traditional software processes, but pragmatism has lead them to adopt some agile practices. Conversely, small companies that intend to follow a completely agile methodology, soon realize that a more structured process is required (Kuhrmann et al., 2017). However, it is not easy to assess which combination of agile and traditional practices adopted is the most appropriate one provided that this depends on the intended process characteristic (Klünder et al., 2019; Gill et al., 2018).

The DynaTail method is intended for companies that should adapt their processes frequently according to the needs of their clients with largely different needs (Silvestre et al., 2021). In particular, tailoring in this case does not only refer to adapting the general process to a particular project context but also adopting different practices according to the characteristics that need to be optimized. To this end DynaTail defines an evaluation activity in order to decide how appropriate is the resulting process. For example, if the context corresponds to a new development in a well known domain and counting on a highly competent development team, the development time will be maximized maybe at the expense of cost, therefore, if the cost is to be minimized, we can evaluate the

[a] https://orcid.org/0000-0003-1806-8647
[b] https://orcid.org/0000-0002-8616-2144

Wallberg, A., González, D., Silvestre, L. and Bastarrica, M.
A Tool for Modeling and Tailoring Hybrid Software Processes.
DOI: 10.5220/0012436500003645
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2024)*, pages 264-271
ISBN: 978-989-758-682-8; ISSN: 2184-4348
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tailored process and decide to change the context by assigning a less competent team even at the expense of the development time.

DynaTail has been validated in a real software development company (Marín et al., 2023). Although the company's process engineer highly appreciated the method, he highlighted the complexity of applying DynaTail without a supporting tool. The models that support DynaTail have already been partly defined in (Silvestre et al., 2021). In this paper we present DynaTool, the tool that implements a supporting tool for DynaTail based on a refined and extension version of these models. DynaTool allows the process engineer to define, tailor and evaluate a process to be followed for a certain project dealing only with domain-specific concepts. We illustrate its use by replicating the case presented in (Marín et al., 2023).

2 BACKGROUND

2.1 Process Tailoring and Improvement

Software process improvement (SPI) is the area in software engineering that deals with the evaluation of software development processes, to assess and potentially improve them (Humphrey, 1995).

SPI used to follow strategies based on models such as CMMI and standards such as ISO that define a series of process areas that should be considered in order to reach a certain maturity level. This approach added repeatability but they have shown to be rigid for some kinds of applications, e.g. innovation projects.

Software process tailoring is the act of adjusting the activities of a process in order to create a new process suited to a different (and likely narrower) context (Ginsberg and Quinn, 1995).

But the context is not all that matters. Peng Xu and Balasubramaniam (Xu and Ramesh, 2007) found that tailoring is not only influenced by the project context, but also a set of environmental factors, challenges, project goals and process tailoring strategies. Vijayasarathy and Butler found similar results (Vijayasarathy and Butler, 2016).

Kalus and Kuhrmann (Kalus and Kuhrmann, 2013) present a systematic review on criteria used for tailoring software processes. By demonstrating the relevance of particular components of the organizational setting, these tailoring criteria can build guidelines for directing the selection of agile methods.

Pedreira et al. present a systematic review on software process tailoring approaches analyzing their degree of formality (Pedreira et al., 2007). They found that most proposals include certain degree of formality but most of them are only applicable for large organizations.

SPEM and BPMN are the OMG standards for specifying software processes and business processes, respectively. Although SPEM is expressive for capturing subtleties of software processes, it lacks supporting tools. On the other hand, BPMN counts on a plethora of tools but it is not expressive for certain particularities of software development (Dumas and Pfahl, 2016). However, in most cases BPMN supporting tools are enough.

Hurtado et al. (Hurtado Alegria et al., 2013) propose a MDE-based strategy for software process tailoring. They consider a process specified in SPEM and a tailoring transformation that takes the process and the context models as input and yields a project specific process also specified in SPEM. This approach makes use of SPEM's variation primitives for identifying the process variation points. Pillat et al. (Pillat et al., 2015) introducte BPMNt, an extension of BPMN for defining variability in software processes specified using BPMN. Even though this approach makes use of the user friendlier notation of BPMN, drifting from the standard makes them incompatible with supporting tools preventing automatic transformation.

2.2 Hybrid Software Process Improvement

Traditional software processes intend to bring structure into software development so that projects are easily managed. They define steps in order to avoid uncertainty and improvisation. However, in projects for innovative domains or not well defined requirements, these processes do not result effective. Agile software development methods have been proposed to deal with these difficulties.

Several companies have adopted agility, but completely agile projects are difficult to provision and schedule. Hybrid software processes that combine some agile and some traditional practices are a tradeoff approach. Kuhrmann et al. (Kuhrmann et al., 2017) defined a hybrid software process as "any combination of agile and traditional approaches that an organizational unit adopts and customizes to its own context needs". One of the first proposal for hybrid software processes is "Water-Scrum-Fall" (West et al., 2011) where management activities are addressed with traditional practices while software development follows Scrum. But not any combination of practices is appropriate (Prenner et al., 2021) for the organization and the project goals.

Evaluation of each combination of practices is a challenging task (Unterkalmsteiner et al., 2012). So, deciding the appropriate degree of agility is not an easy task either (Diebold and Zeher, 2016): Which activities should be addressed with each approach? What practice should be used for implementing each activity? Although there are some empirical guidelines (Tell et al., 2019), process desired characteristics and available practices evolve over time and thus choosing the appropriate combination of practices should be continuously adjusted (Klünder et al., 2019).

2.3 DynaTail

DynaTail acknowledges that context-based tailoring is not enough for process improvement, since two or more tailored processes may be consistent with the context but they improve different attributes. Therefore, DynaTail explicitly considers these two dimensions: context and desired attribute to be improved.

An initial version of DynaTail has already been presented in (Silvestre et al., 2021). In order to build a supporting user-friendly tool, the method should be fully formalized.

The method involves two main activities: tailoring the process to the context, and evaluating it according to the chosen characteristic. Tailoring is defined as a model transformation that takes the process and context models and yields an adapted process. The evaluation is carried out by using an influence graph for the intended characteristic.

Here, the attributes that influence the intended characteristic to be improved are specified along with the weight of this influence. Similarly, the set of activities in the tailored "Process model" may influence each of these attributes with different weights. Finally, each activity may be implemented with different practices, each one with its own influence weight. There is a different influence graph for each characteristic, and they are specific to the organization since the included activities are those in the "Process model" and the practices are the ones regularly applied in the organization. The weights are organization-specific too and they range between -2 and 2, as suggested by Diebold et al. (Diebold and Zehler, 2015), where -2 indicates a highly negative influence and 2 is a highly positive influence. These weights may be adjusted in time by the company as a consequence of empirical results in previous projects. Once the tailored process is evaluated, the process engineer may decide that is good enough of he/she may decide to manually change the process or the context.

3 DynaTool

In this section we present DynaTool, its user interfaces and its supporting models (figure 1). We also illustrate its application by replicating the *Trade-off Agile Planning Process* tailoring presented in (Marín et al., 2023) considering the context factors of the: Product Owner and Willingness to negotiate. It is worth noting that DynaTool differentiates activities carried out by the Process engineer from those of the Project engineer. The former correspond to organization-related activities while the later are project-related.

3.1 Process Definition and Modeling

There are two stages when the process may be defined. At the beginning, the process engineer defines the organizational process and afterwards, when DynaTool is applied for a particular project, the project engineer may eventually decide to modify it if the results are not considered good enough.

In order to integrate a process in DynaTool we needed to use an established modeling standard. We have chosen BPMN for DynaTool. The BPMN is the OMG standard language for modeling process models. This notation is widely used in industry even though it has are some limitations for modeling software development processes such as defining activities that carried out by different roles simultaneously that is common in software processes. However, it has the big advantage of being user friendly and there is a plethora of tools for process definition. This later property implies benefits and drawbacks. Companies may choose among different tools, either free or proprietary for modeling their processes. DynaTool allows the process engineer to use a series of different tools for process definition such as BonitaSoft, Eclipse Modeler, BPMN.io and Bizagi. However, each of these tools implements its own *flavor* of BPMS adding some extra characteristics that are not necessarily compatible with BPMN 2.0 that is the strict standard.

To address this issue, we have built a set of projectors: an injector that transforms the BPMN process (BPMN file) into BPMN process model (XMI file), and an extractor that transforms the BPMN process model of the configured process (XMI file) into the BPMN process description (BPMN file) afterwards during the *Process Evaluation* activity.

The projectors consider matching elements between the BPMN process and the BPMN process model. In this sense, there are process elements in BPMN process that are not used for the injector, but

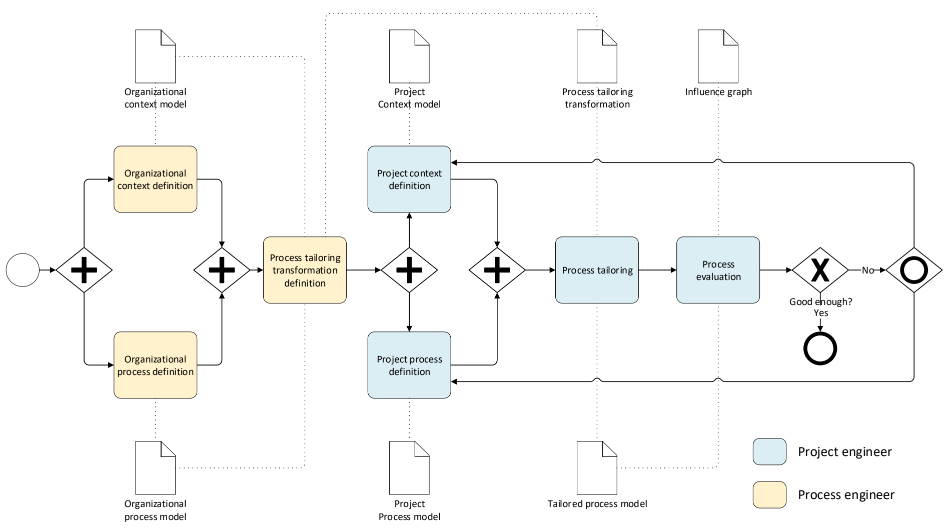

Organizational context model
Project Context model
Process tailoring transformation
Influence graph
Organizational context definition
Project context definition
Process tailoring transformation definition
Process tailoring
Process evaluation
No
Good enough?
Yes
Organizational process definition
Project process definition
Organizational process model
Project Process model
Tailored process model
Project engineer
Process engineer

Figure 1: DynaTool's models and user interfaces.

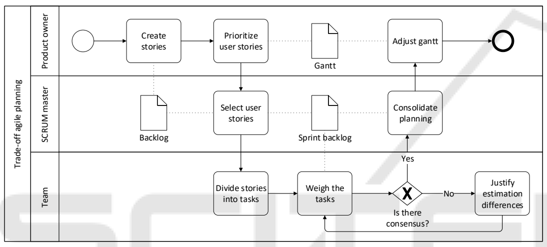


Figure 2: Trade-off Agile Planning Process.

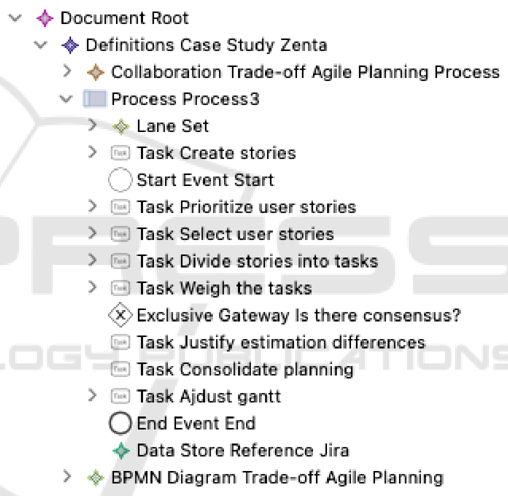

Document Root
Definitions Case Study Zenta
Collaboration Trade-off Agile Planning Process
Process Process3
Lane Set
Task Create stories
Start Event Start
Task Prioritize user stories
Task Select user stories
Task Divide stories into tasks
Task Weigh the tasks
Exclusive Gateway Is there consensus?
Task Justify estimation differences
Task Consolidate planning
Task Ajdust gantt
End Event End
Data Store Reference Jira
BPMN Diagram Trade-off Agile Planning

Figure 3: Trade-off Agile Planning Process Model.

that need to be preserved because they will be later needed for the extractor.

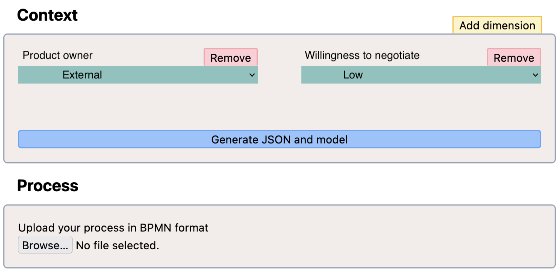
The injector applies the following steps: (1) Identify elements of the BPMN process that are relevant for building the process model, (2) Semantic analysis of the labels in the BPMN file, (3) Establish a dictionary that implements the matching elements, (4) Create a hierarchical structure from the BPMN 2.0 metamodel, (5) Build the process model (XMI file).

Figure 2 shows the Trade-off Agile Planning process defined using BonitaSoft while Figure 3 shows the BPMN process model generated from BPMN process after applying the injector. This BPMN process model conforms to the BPMN 2.0 metamodel and can be manipulated using EMF tools. However, the BPMN process model does not consider the graphical elements (only standard process elements).

The extractor applies the following steps: (1) Identify elements of the BPMN process model for building the BPMN process, (2) Semantic analysis of the labels in the XMI file, (3) Establish a dictionary that implements the matching elements, (4) Create a hierarchical structure from the BPMN description, (5) Build the BPMN process model (BPMN file). However the XMI file does not contain information about process elements sequencing, the original BPMN file is also used as input in order to build the BPMN file after the process model has been tailored.

3.2 Context Definition and Modeling

A project context is defined by the characteristics where it takes place. Some of the usual characteristics considered in software development projects are system size, software complexity, development team size and knowledge about the application domain, among others. Although a company can define its own particular development process and always apply it the the same way, it may also consider context characteristic in order to select or tailor the process so that it becomes more specifically appropriate for each project.



Context

Add dimension

Product owner	Remove	Willingness to negotiate	Remove
External		Low	

Generate JSON and model

Process

Upload your process in BPMN format

Browse... No file selected.

Figure 4: Context definition tool user interface.

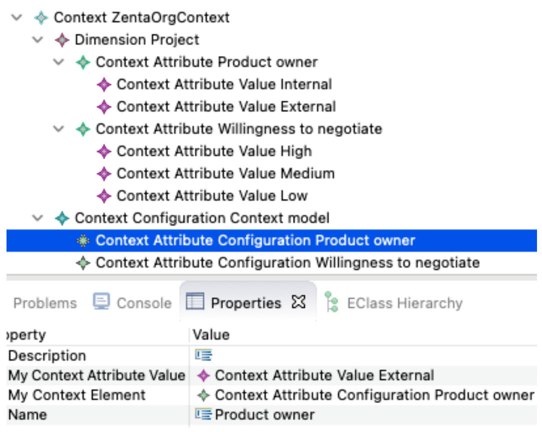
Even though any characteristic could be used to define the context for tailoring the process, in practice it was found that project/product size and application domain are the dimensions most widely considered for process tailoring (Klünder et al., 2020). In this paper, for generality, we consider any type of dimension.

In order to build DynaTool we formally represent the context in the form of a model that allows it to be seamlessly integrated into the tool. We have built a custom made context definition interface that allows the process engineer to define the context attributes that will be considered and the project engineer to define the value for these characteristics for a certain project.

For example, in Figure 5, on the upper part we can see that the *Project owner* attribute may take "Internal", or "External" values. In the lower part, i.e. in the configured "Context model", the *Project owner* is assigned to have a "External" specific value.

The context model includes two sections: *Organizational Context Model* that specifies all context attributes and their potential values, and *Project Context Model* that configures the context attributes with particular values for defining a project context. Only attributes that have been defined as part of the *Organizational Context Model* can be configured. Figure 5 shows the result of applying an injector to the context definition generated through the user interface in Figure 4. It can be seen that the *Project owner* attribute indicated in the upper part of the model, takes the value *External* as stated in the bottom of the model specification. The other attributes are managed similarly.

It is worth noting that in the lower part of the interface there is the possibility of uploading a software process model. This will be necessary for defining the tailoring transformation described in the following section.



```
Context ZentaOrgContext
  Dimension Project
    Context Attribute Product owner
      Context Attribute Value Internal
      Context Attribute Value External
    Context Attribute Willingness to negotiate
      Context Attribute Value High
      Context Attribute Value Medium
      Context Attribute Value Low
  Context Configuration Context model
    Context Attribute Configuration Product owner
    Context Attribute Configuration Willingness to negotiate
```

Problems | Console | Properties | EClass Hierarchy

›perty	Value
Description	
My Context Attribute Value	Context Attribute Value External
My Context Element	Context Attribute Configuration Product owner
Name	Product owner

Figure 5: Context model.

3.3 Process Tailoring Definition and Modeling

Process tailoring is addressed with an ATL transformation that takes the process and the context models as input and yields a tailored process model. Translating process and context definition to their respective model representations is addressed with model injectors while visualizing the tailored process is addresses through a model extractor. In what follows we describe each of these activities, their interactive domain specific user interfaces and their supporting MDE models.

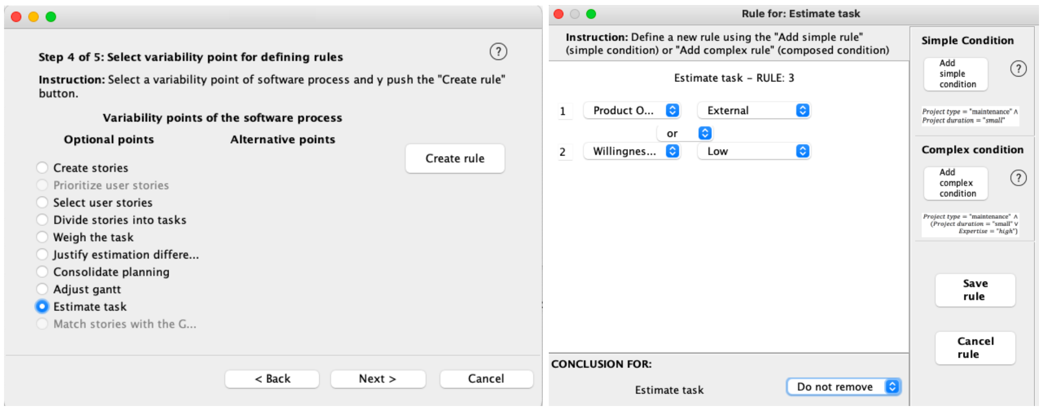
Figure 7 shows three rules that the process engineer defined for his process. For example, rule 3 establishes that whenever the *Product owner* is "External" or his *Willingness to negotiate* is "Low", then *Estimate tasks* and *Match stories with the Gantt* practices must be included in the tailored process.

Writing tailoring rules in a formal language is highly complex. This could be a determinant impediment if we intend DynaTool to be applicable for any process and any context. Therefore, we have designed a user interface that allows the process engineer to interactively define these tailoring rules.

To this end, the *Process model* and *Context model* should have been previously defined. This is why the lower part of the context definition interface allowed for uploading the *Process model*. First, DynaTool identifies all activities in the *Process model*, as shown in Figure 6a. Here, the process engineer can select an activity that could eventually be implemented differently according to certain context values.

For example, in this example "Estimate task" has been selected. Then, a specific rule for this activity con be defined interacting with the user interface shown in Figure 6b.

Finally, after defining all the desired rules, the


Step 4 of 5: Select variability point for defining rules

Instruction: Select a variability point of software process and y push the "Create rule" button.

Variability points of the software process

Optional points | Alternative points

- Create stories
- Prioritize user stories
- Select user stories
- Divide stories into tasks
- Weigh the task
- Justify estimation differe...
- Consolidate planning
- Adjust gantt
- [✓ Estimate task]
- Match stories with the G...

Rule for: Estimate task

Instruction: Define a new rule using the "Add simple rule" (simple condition) or "Add complex rule" (composed condition)

Estimate task – RULE: 3

1 Product O... External

or

2 Willingnes... Low

CONCLUSION FOR:

Estimate task: Do not remove

Simple Condition
Add simple condition
Project type = "maintenance" ∧ Project duration = "small"
Complex condition
Add complex condition
Project type = "maintenance" ∧ (Project duration = "small" ∨ Expertise = "high")
Save rule
Cancel rule

(a) Selecting variable activity. (b) Defining transformation rules.

Figure 6: Transformation rules user interface.

```
helper def:ruleOpt1():Boolean=
    if(thisModule.getValue('Project owner type') = 'external'
            and thisModule.getValue('Willingness to negotiate') = 'low' )
        then true else false endif;


helper def:ruleOpt2():Boolean=
    if((thisModule.getValue('Project owner type') = 'external'
            and thisModule.getValue('Willingness to negotiate') = 'medium')
            or (thisModule.getValue('Project leader type') = 'internal'
                or thisModule.getValue('Willingness to negotiate') = 'high') )
        then true else false endif;

helper def:ruleOpt3():Boolean=
    if(thisModule.getValue('Project owner type') = 'external'
            or thisModule.getValue('Willingness to negotiate') = 'low' )
        then true else false endif;
```

Figure 7: Tailoring rules automatically generated.

transformation in Figure 7) can be automatically generated. It is worth mentioning that this operation is a M2T higher order transformation.

3.4 Process Evaluation

The process evaluation finally computes how good is the tailored process for optimizing an intended characteristic. To this end, the organization counts on an *Influence graph model* that specifies how much each agile practice influences each activity of the organizational process. Once the process is tailored in the previous step, only some of the activities are kept, and therefore the specific value should be computed accordingly. It is worth mentioning that there is a specific *Influence graph* for each potential characteristic to be optimized.

The *Influence graph model* is structured in four levels. First, the characteristic to be optimized that in our example is "value added". A second level of attributes that, according to the literature, have an influence on that characteristic. Third, we have the set of activities on the process model that influence each attribute. And finally, the set of agile practices that may be used for implementing each of these activities.

The "Influence graph model" is the input to the *Process evaluation* activity that is implemented as a M2T model transformation that takes the influence graph and tailored process models and yields a value.

Equations (1), (2), and (3) formally specify how these values are calculated. These formulas are further explained in detail in (Silvestre et al., 2021).

$$Ia(Act_i) = \sum_{j=1}^{|Prac|} \frac{I(Prac_j, Act_i)}{|\{Prac_j : I(Prac_j, Act_i) \neq 0\}|} \quad (1)$$

$$Iat(Att_k) = \sum_{i=1}^{|Att|} I(Act_i, Att_k) * Ia(Act_i) \quad (2)$$

$$Ich(Ch_t) = \sum_{k=1}^{|Ch|} I(Att_k, Ch_t) * Ia(Att_k) \quad (3)$$

If the value obtained is considered good enough, the method ends and the best process for optimizing the intended characteristic is the "Hybrid process model" that defines not only the activities to be followed but also the practices that should be applied in each step.

On the contrary, if the process engineer considers that the resulting value is not good enough, it may proceed to *Change process* or *Change context* and restart whole method. Modifying the process could be for example, adding new steps or roles not present in the previous process, while modifying the context could be for example, adding more developers to the team in charge of the project.

Finally, we apply the extractor that takes as input the XMI tailored process and generates the BPMN process. The BPMN process is automatically generated and can be visualized from a BPMN tool as Bizagi, BonitaSoft, BPMN.io or Eclipse Process Modeler.

Provided that the main goal of Dynamic is defining the setting for addressing a certain project, i.e. the process, context and practices that can best

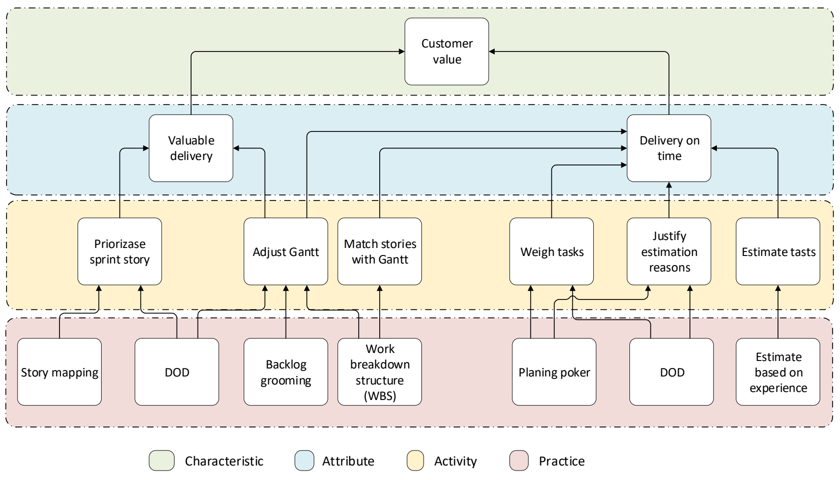
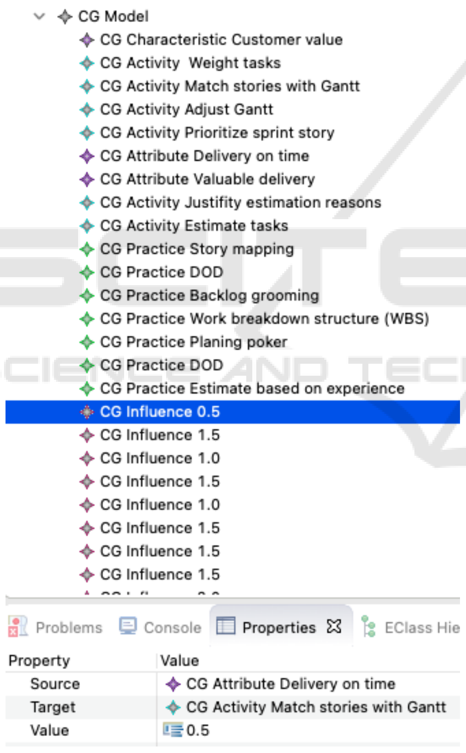


Figure 8: Influence Graph.



CG Model
CG Characteristic Customer value
CG Activity Weight tasks
CG Activity Match stories with Gantt
CG Activity Adjust Gantt
CG Activity Prioritize sprint story
CG Attribute Delivery on time
CG Attribute Valuable delivery
CG Activity Justifity estimation reasons
CG Activity Estimate tasks
CG Practice Story mapping
CG Practice DOD
CG Practice Backlog grooming
CG Practice Work breakdown structure (WBS)
CG Practice Planing poker
CG Practice DOD
CG Practice Estimate based on experience
CG Influence 0.5
CG Influence 1.5
CG Influence 1.0
CG Influence 1.5
CG Influence 1.0
CG Influence 1.5
CG Influence 1.5
CG Influence 1.5

Property	Value
Source	CG Attribute Delivery on time
Target	CG Activity Match stories with Gantt
Value	0.5

Figure 9: Influence Model.

achieve the desired characteristic, the concept of "good enough" depends on the process engineer criteria. However, a more objective method can be followed with a *what-if* strategy comparing different settings. In any case, it is still the process engineering the one who decides for example if the organization has the resources to configure a certain context or applied certain practices.

4 CONCLUSIONS

A software process can be considered good not only if it is appropriate for its context, but also it helps achieving a desired goal, i.e. a high value in a particular characteristic such as development time or value added. Agile software development inherently promotes adopting and adapting different practices but a priori it is not obvious which combination of practices may be appropriate for achieving the project's goal.

Model-Driven Development (MDD) proves to be a valuable formality for addressing software process tailoring, and specifically hybrid software processes too. Although it establishes a robust foundation for this purpose, its effectiveness is accompanied by the inherent complexity associated with formality.

This paper introduces DynaTool, a highly user-friendly tool designed to interactively addressing the identified problem. To assess its efficacy, we replicated a previously published example featuring the application of Dynamic, the original method, in an industrial context. Our findings indicate that DynaTool effectively addresses most of the reported limitations. However, further validation with diverse industrial processes is essential to comprehensively assess its versatility and performance.

REFERENCES

Clarke, P. and O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54(5):433–447.

Diebold, P. and Zeher, T. (2016). The Right Degree of

Agility in Rich Processes. In *Managing Software Process Evolution*, pages 15–37. Springer.

Diebold, P. and Zehler, T. (2015). The agile practices impact model: idea, concept, and application scenario. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 92–96. ACM.

Dumas, M. and Pfahl, D. (2016). Modeling Software Processes Using BPMN: When and When Not? In Kuhrmann, M., Münch, J., Richardson, I., Rausch, A., and Zhang, H., editors, *Managing Software Process Evolution: Traditional, Agile and Beyond – How to Handle Process Change*, pages 165–183. Springer International Publishing.

Gill, A. Q., Henderson-Sellers, B., and Niazi, M. (2018). Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. *Information Systems Frontiers*, 20:315–341.

Ginsberg, M. P. and Quinn, L. H. (1995). *Process tailoring and the software capability maturity model*. Citeseer.

Humphrey, W. (1995). *A Discipline for Software Engineering*. SEI Series in Software Engineering. Addison Wesley.

Humphrey, W. S. (1988). The software engineering process: definition and scope. In *Proceedings of the 4th international software process workshop on Representing and enacting the software process*, pages 82–83.

Hurtado Alegria, J. A., Bastarrica, M. C., Ochoa, S. F., and Simmonds, J. (2013). MDE software process lines in small companies. *J. Syst. Softw.*, 86(5):1153–1171.

Kalus, G. and Kuhrmann, M. (2013). Criteria for software process tailoring: a systematic review. In *International Conference on Software and System Process*, pages 171–180. ACM.

Klünder, J., Hebig, R., Tell, P., Kuhrmann, M., Nakatumba-Nabende, J., Heldal, R., Krusche, S., Fazal-Baqaie, M., Felderer, M., Bocco, M. F. G., et al. (2019). Catching up with method and process practice: An industry-informed baseline for researchers. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 255–264. IEEE, IEEE / ACM.

Klünder, J., Karajic, D., Tell, P., Karras, O., Münkel, C., Münch, J., MacDonell, S. G., Hebig, R., and Kuhrmann, M. (2020). Determining context factors for hybrid development methods with trained models. In *International Conference on Software and System Processes, ICSSP'2020*, pages 61–70. ACM.

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E., and Prause, C. R. (2017). Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In *Proceedings of the International Conference on Software and System Process, ICSSP'2017*, page 30–39. ACM.

Marín, J., Hurtado, J. A., Bastarrica, M. C., and Silvestre, L. (2023). Tailoring hybrid software processes in a medium-size software company. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC*, pages 1042–1050. ACM.

Münch, J., Armbrust, O., Kowalcyzk, M., and Soto, M. (2012). *Software Process Definition and Management*. Springer-Verlag, Germany.

Pedreira, O., Piattini, M., Luaces, M. R., and Brisaboa, N. R. (2007). A systematic review of software process tailoring. *SIGSOFT Softw. Eng. Notes*, 32(3):1–6.

Pillat, R. M., Oliveira, T. C., Alencar, P. S. C., and Cowan, D. D. (2015). BPMNt: A BPMN extension for specifying software process tailoring. *Inf. Softw. Technol.*, 57:95–115.

Prenner, N., Unger-Windeler, C., and Schneider, K. (2021). Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process*, 33(11):e2382.

Raharjo, T. and Purwandari, B. (2020). Agile Project Management Challenges and Mapping Solutions: A Systematic Literature Review. In *Proceedings of the 3rd International Conference on Software Engineering and Information Management*, page 123–129. ACM.

Silvestre, L., Bastarrica, M. C., Hurtado, J. A., and Marín, J. (2021). Formalizing the Goal-directed and Context-based Software Process Tailoring Method. In *XLVII Latin American Computing Conference, CLEI*, pages 1–9.

Tell, P., Klünder, J., Küpper, S., Raffo, D., MacDonell, S. G., Münch, J., Pfahl, D., Linssen, O., and Kuhrmann, M. (2019). What are hybrid development methods made of? an evidence-based characterization. In *2019 IEEE/ACM International Conference on Software and System Processes*, pages 105–114.

Unterkalmsteiner, M., Gorschek, T., Islam, A. M., Cheng, C. K., Permadi, R. B., and Feldt, R. (2012). Evaluation and measurement of software process improvement—a systematic literature review. *IEEE Transactions on Software Engineering*, 38(2):398–424.

Vijayasarathy, L. R. and Butler, C. W. (2016). Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software*, 33(5):86–94.

West, D., Gilpin, M., Grant, T., and Anderson, A. (2011). Water-scrum-fall is the reality of agile for most organizations today. *Forrester Research*, 26(2011):1–17.

Xu, P. and Ramesh, B. (2007). Software process tailoring: An empirical investigation. *Journal of Management Information Systems*, 24(2):293–328.