DynaTail: A Method for Hybrid Software Process Tailoring

Jacqueline Marín[1], María Cecilia Bastarrica[1],
Julio Ariel Hurtado[2], and Luis Silvestre[3]

[1] Computer Science Department, Universidad de Chile, Chile
{jamarin,cecilia}@dcc.uchile.cl
[2] IDIS Research Group, Universidad del Cauca, Colombia
ahurtado@unicauca.edu.co
[3] Computer Science Department, Universidad de Talca, Chile
lsilvestre@utalca.cl

Abstract. It is widely accepted that a good software process is essential for developing good software products. International studies have seen that most software companies apply hybrid processes for developing software, i.e., an organized combination of traditional and agile practices. Software process improvement and tailoring have been researched for several years, but there is less experience when dealing with hybrid software processes. Therefore it is not always clear to determine the combination of practices that best fits a certain context, either for the company or a certain project. Moreover, the most appropriate process also depends on a particular goal. In this paper we propose the Dynamic Tailoring for Hybrid Software Processes (DynaTail), a method that aims to achieve three purposes: (1) allow for the tailoring of the organization process in order to adapt to the project context when pursuing certain goal; (2) provide a framework for trying different process and context modifications in order to obtain better results for a particular project. We describe the method in full detail illustrating it with a running example taken from industry. We have found that the proposed method is a promising path for the quantitative evaluation of hybrid software processes.

Keywords: Hybrid software process · Agile practices · Software process tailoring · Process measurement

1 Introduction

Defining software processes has been a means for systematizing software development in several organizations [11]. There is a plethora of processes and standards that have been proposed in this line. The Waterfall model has guided the software industry for several decades and still has a high incidence. It has provided support for managing projects, training new developers, and minimizing uncertainty and improvisation. However, it has sometimes been felt as too restrictive, limiting innovation and flexibility [6]. Agile approaches address these issues by introducing a series of practices that each development team can adopt

and adapt according to their needs. Since the late '90s, several agile methods have been in use such as XP, Scrum, Lean software development, or RUP. But only in 2001, the Agile Manifesto synthesized the philosophy that guides all of them.

In general, agile methods focus on people involved in software development, are mainly appropriate for small teams, and promote productivity, especially in projects with high uncertainty. On the other hand, they do not provide strong support for project management activities [16].

Intending to bring a trade-off between structure and flexibility, hybrid software development processes have been applied [5]. Here, some activities are addressed by applying agility while others still follow traditional practices. However, it is not always clear which activities should be addressed with each paradigm, provided that this depends on the characteristics of the organization and the project context. For example, innovative projects tend to favor a higher level of agility, while projects within a well-known application domain and technology result in more efficient applying structured processes. But the best combination of practices also depends on the intended goal of the project. For example, if time constraints are a priority, structured processes that manage a highly detailed plan could be better, but at the expense of managerial costs. On the other hand, innovative projects require agility, but time and cost cannot always be managed because of the inherent uncertainty. Nevertheless, there are some recently proposed models for addressing hybrid software processes management [20].

According to the literature, there are numerous improvement goals [10] such as quality, complexity, maintainability, performance, effort estimation, maturity, and usability, among others. Most of these works are applied in the context of XP and Scrum, and they mainly address *integrate often*, *test-first*, *daily meeting*, *pair programming*, *retrospective*, *on-site customer*, and *product backlog*.

In this paper, we propose DynaTail, a method for tailoring a software process to the particular context of the project where it will be applied, also considering the intended goal. The method consists of three steps:

1. Tailor the software process: starting with a process, it is adjusted to the particular context applying tailoring rules.
2. Evaluate the process in terms of the goal according to its practices: an influence graph is built, and the process is evaluated in terms of the weighted influences in the graph.
3. Adjust the process or the context through "what-if" activities.
   - The process could be adjusted by adding, deleting, or changing activities.
   - Adjust the context by changing the value of certain factors.

We describe the method in full detail and illustrate each activity with the *Sprint Planning* process followed in a medium-size Chilean software company. This running example was able to replicate the way the company defines its process. This is an encouraging result that backs the potential of the method.

The rest of the paper is structured as follows. Section 2 describes some related work. The complete method, along with the running example is described in section 3. Finally, section 4 states our conclusions and describes future work.

2 Related work

DynaTail focuses mainly on tailoring and evaluating hybrid software processes. In this section, we discuss some existing proposals in these areas.

2.1 Tailoring agile software processes

Software process tailoring refers to the activity of adjusting it to meet the needs of a certain context. Clarke and O'Connor proposed a reference framework of the situational factors that affect the software development process [3]. This framework consists of 8 context classifications, 44 factors, and 170 sub-factors. Although illuminating, the size of the framework makes it impractical to be applied directly. There are a lot of different approaches proposed for software process tailoring [12], but only a few for agile processes.

Ambler and Lines propose the Situation Context Framework (SCF) to support selecting and tailoring an agile situation-dependent strategy [2]. SCF defines several context factors that affect how a team chooses its way of work. These factors are organized into two categories: those that have a significant impact on the choice of the software process, such as team skills, and those, that motivate the choice of the practices/strategies, as team availability.

Diebold and Zehler create the Agile Practices Impact Model (APIM) for representing the influence of agile practices on different impact characteristics [4]. This influence can be positive or negative depending on the influence factors (context factors or other practices) that affect the relationship between agile practices and characteristics. APIM can be used to support the tailoring of software processes according to a specific context.

Although the two last-mentioned publications consider only agile development, our proposal is aligned with their concepts. On the contrary, tailoring approaches for hybrid software processes are scarce and new.

2.2 Tailoring hybrid software processes

Vijayasarathy and Butler [19] conducted a survey on the relationship of organizational, project, and team characteristics with the methodologies used. They found that the characteristics associated with hybrid software processes are medium projects concerning the budget, highly criticality of the project, and small team size. The study also suggests that the organization size does not impact the use of hybrid processes.

Kuhrmann et al. [9] report a survey on hybrid software development showing which approaches are used in practice, how different approaches are combined, and what contextual factors influence the use and combination of hybrid software development. The study also shows that hybrid practices are mainly configured applying different strategies: a process improvement program, evolution information from past projects, and situation-specific definition. This study shows that process evolution and improvement are as relevant as project-specific tailoring.

Klünder et al. [8] analyzed 829 data points from the HELENA dataset. They found that a few context factors, e.g., project/product size and target application domain, significantly influence the selection of methods. They also found that certain practices are used in specific contexts, e.g., *daily meetings* are better for colocated teams.

Prenner et al. [15] identified three process patterns: Waterfall-Agile-Approach, the Waterfall-Iterations-Approach, and the Pipeline-Approach. However, their study does not show how the practices could be combined for achieving a certain goal or suit a particular context.

2.3 Evaluation of Agile and Hybrid Software Processes

Software process evaluation has been extensively addressed by organizations [10], being CMMI (Capability Maturity Model) and GQM (Goal Question Metric) the most identified measurement model and method, respectively. Evaluations based on quality models measure the grade in which the process fulfills the objectives and practices defined by it. In contrast, evaluations based on improvement goals follow a measurement system defined by the organization following the GQM method. Our work follows a goal-oriented process evaluation strategy for hybrid software processes, from practices to improvement characteristics.

According to Prenner [14], software companies have difficulties evaluating and analyzing if their hybrid processes are suitable for their context and goals. The author identified some challenges for configuring hybrid processes; however, the study did not identify the characteristic to evaluate this suitability. Alaidaros et al. [1] reviewed 48 primary studies and found that *applicability, effectiveness, efficiency, and quality* are the most frequent key factors used for measuring agile process quality. Perkusich et al. [13] propose a procedure to detect problems in software processes using a Bayesian network. The goal is to improve the project's chances of success (probabilistic). Their evaluation scenario was a Scrum-based development process applied by a capable and organized team without a committed and skilled Product Owner or an involved client. They then analyzed the chance of a project to succeed in terms of *team capabilities*, *team commitment* and *stakeholder involvement*.

Wlodarski et al. [21] planned and executed a controlled experiment with student projects to study the implications of introducing a hybrid approach in student projects, particularly adding incremental delivery. They evaluated *the team productivity* and the *external quality*, contrasting iterative and sequential processes. Their results conclude that the proposed hybrid approach contributed to a significant increase in *team productivity*.

The method presented in this paper includes evaluating a hybrid software process with respect to different improvement process characteristics. In this way, the process engineer can choose both a good process and an appropriate context to maximize the intended characteristic. The method is described using a running example evaluating a process with the *customer value* as improvement characteristic considering either an 'experienced' or an 'inexperienced' Scrum team as the different contexts.

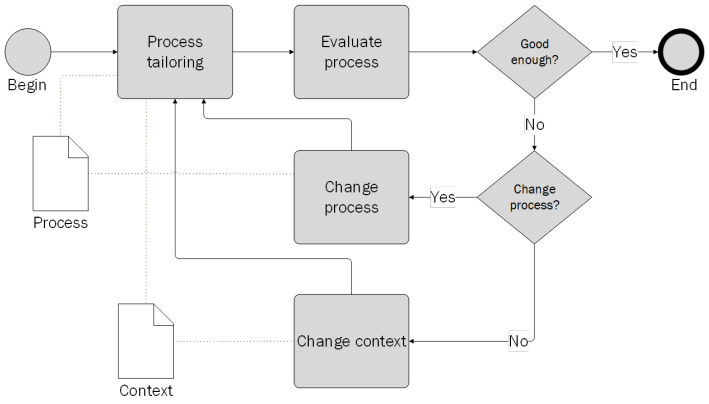

Fig. 1. Dynamic Tailoring Method for Hybrid Processes- DynaTail

3 The DynaTail method

DynaTail is a context-driven and goal-oriented method that intends to provide dynamism to the hybrid process tailoring activity. It provides a framework for obtaining the process adapted to a certain context that best contributes to reaching the desired goal.

The method obtains a process that the process engineer considers good enough within a particular context concerning a certain characteristic, as depicted in Figure 1. Its inputs are the standard process, the project context, and the characteristic to be improved. The method includes a tailoring loop where either the process or the context can be changed to achieve a better process. Changing the process implies including new activities, deleting activities, or substituting activities with others implementing different practices. The change in the context represents potential changes related to resources assigned to the project. This new process and context are the input for the new tailoring cycle. The output of the method is a process to be applied in the project as well as its context. This context may be the original one or a newly defined one.

Each activity will be described in detail in the following sections, including its objective, inputs, tasks, and outputs.

3.1 Process Tailoring

DynaTail's tailoring requires three elements: a process, a context, and a set of tailoring rules, and it consists of adapting the process to the particular context by applying the rules. The result is a new process.

Figure 2 shows the *Agile Sprint Planning* process modeled in BPMN as a starting process. We can see that the Scrum Team is in charge of the *Estimate tasks* and *Converge to consensus* activities, while the Scrum Master executes *Estimate sprint story*.

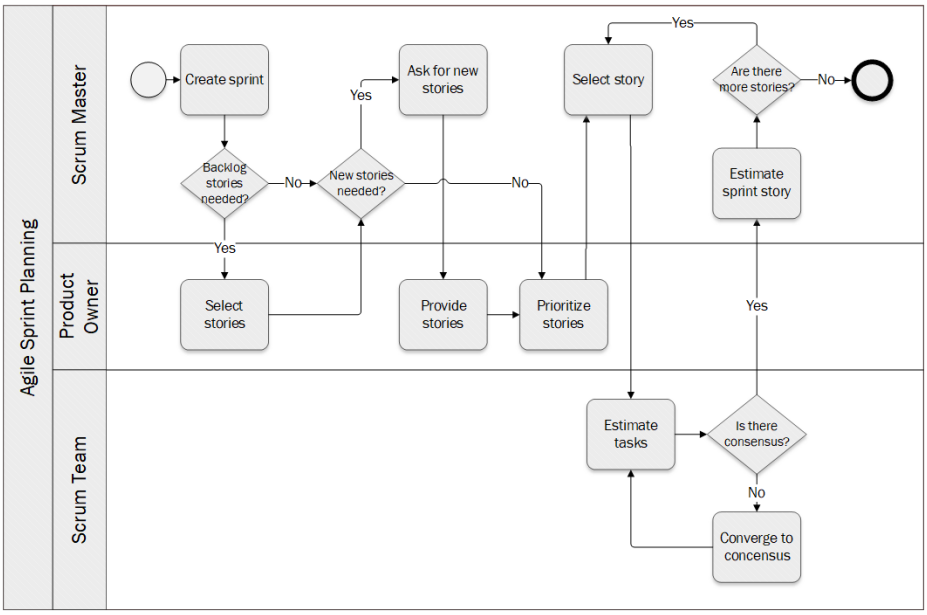

Fig. 2. Agile Sprint Planning process

Table 1. Project Context

Category	Context Factor	Context Factor Value
Team	Experience	experienced
		inexperienced

For simplicity, in this running example, we will consider that the context is determined just by one factor: Team experience, as indicated in Table 1. The particular context value for the Tailoring activity will be initially 'experienced', i.e., the Scrum Team is experienced and is perfectly capable of appropriately estimating.

The third required element is a tailoring rule that defines the changes that must take place in the process according to the particular project context. For our example, in the tailoring rule shown in Rule 3.1, we state that whenever the Scrum Team is 'inexperienced' the *Estimate tasks* and *Converge to consensus* activities will be removed while the *Estimate sprint story* will be replaced by *Plan sprint story*.

Then, applying the rule, we see that in the initial context, i.e., when the value of *Experience* is 'experienced', the rule condition does not hold. Therefore no change is applied, and the process remains the same.

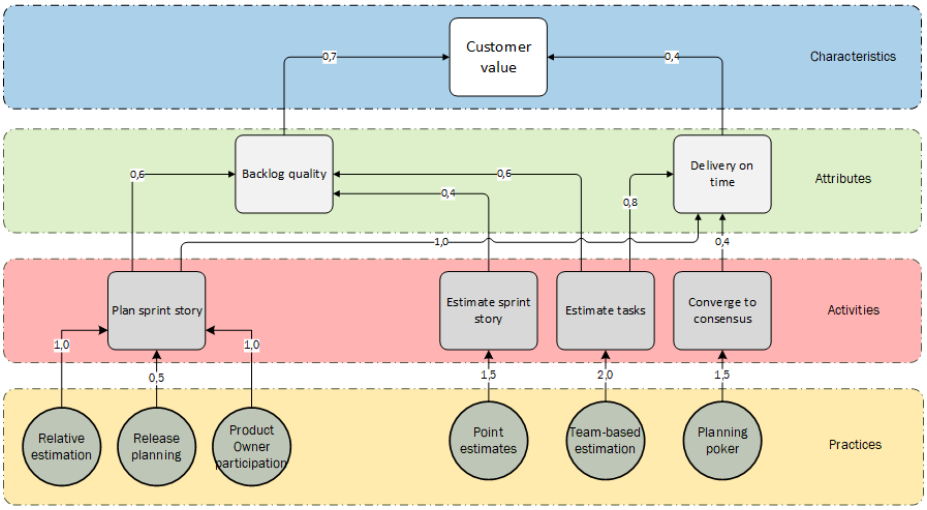

Fig. 3. Characteristic graph

Rule 3.1: Example of Tailoring Rule

```
if Experience is 'inexperienced'
then
    remove 'Estimate tasks'
    remove 'Converge to consensus'
    replace 'Estimate sprint story' by 'Plan sprint story'
endif
```

3.2 Evaluate process

In this activity, the project-specific process is evaluated according to a characteristic aligned with the intended goal. The project-tailored process evaluation is divided into two tasks: build the characteristic graph and evaluate a project-specific process using a model based on this graph.

Build the characteristics graph A characteristics graph in our method considers the following concepts that are shown in each level of figure 3: (1) Goal is a sentence that evidences a problem that the company has identified, (2) Characteristics are indicators derived from the goal that allows for measuring its achievement of it, (3) Attributes are features or properties that influence the characteristic, (4) Activities are a set of process steps that influence the achievement of an attribute, and (5) Practices that are proven ways or strategies for addressing activities in order to achieve their goals.

All characteristics, attributes, activities, and practices required for achieving the improvement goal are organized as part of an influence graph. The process

engineer includes all alternative activities that may be part of the process that can be applied when carrying out a project; they could have been applied before or considered as potentially applied in future projects.

We describe each step followed for building the influence graph of the *Sprint Planning* process:

1. Define and prioritize the improvement goals that will address the identified problem. In our running example, the company identified a problem in delaying adding value to the customer due to his/her poor involvement with the development process principles and values. Thus, the improvement goal to achieve an early delivery of value to the customer.
2. Identify characteristics related to goal achievement. *Customer value* is the only characteristic to be considered in the example, and thus the one to be maximized.
3. Identify attributes related to the achievement of the characteristic. In this case *"Backlog quality"* and *"Delivery on time"* are specific attributes of *Customer value*.
4. Link the standard process activities influencing the attributes and include other activities either from other company processes or from external sources that can improve the goal. In the *Sprint Planning* process, there are four activities influencing *"Backlog quality"* and *"Delivery on time"*: *Estimate sprint story*, *Estimate tasks*, and *Plan sprint story* influence *"Backlog quality* while *Plan sprint story*, *Estimate tasks*, and *Converge to consensus* influence *"Delivery on time"*.
5. Associate the practices influencing each process activity. These associations can be given by the company's experience or by the experience reported in the literature. In our example, *'Relative estimation'*, *'Release planning'*, and *'Product Owner participation'* influence *Plan sprint story* while *'Point estimates'* influences *Estimate sprint story*, *'Team-based estimation'* influences *Estimate tasks*, and *'Planning poker'* influences *Converge to consensus*.
6. Assign weighted values to each influence relationship. The process engineer can assign these values based on his/her experience or considering the company's experience collected either through workshops with all stakeholders or by conducting interviews with them. Another option can be using the experience reported by empirical studies from the literature. For this example, we use values from our experience working with hybrid processes and Diebold's study. We indicated each of these influence values in the diagram in figure 3.

Evaluate a project-specific process The evaluation of the project-specific process consists of obtaining a number that represents how good is the process for achieving the intended goal. This evaluation follows the following steps:

1. The process engineer in charge of defining the process to be applied considers the process resulting from the tailoring activity as well as the characteristics graph. In our running example, these are the *Agile Sprint Planning* process in figure 2 and the characteristics graph in figure 3.

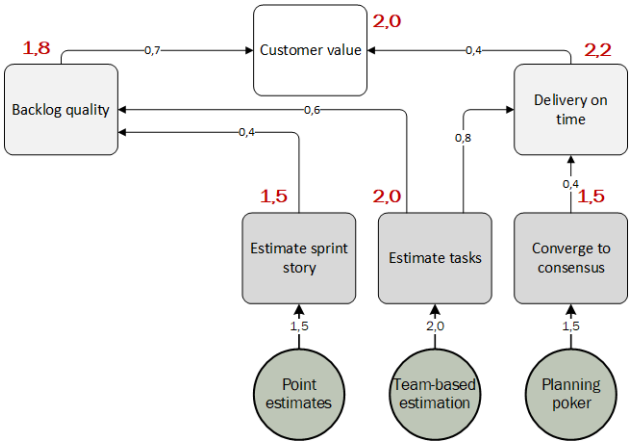

Fig. 4. Agile Sprint Planning process evaluation

2. A particular evaluation graph is built for the process, including only those activities in the tailored process. The evaluation graph for the *Agile Sprint Planning* process is shown in figure 4.
3. Then, the value for the improvement characteristic is depicted in figure 4 and is computed as follows.
   - The value of each activity is the average value of all of its associated practices. In the *Agile Sprint Planning*, there will be three activities considered, each one associated with a unique practice: *Estimate sprint story* with *'Point estimate'* with value 1.5, *Estimate tasks* with *'Team-based estimation'* with value 2.0, and *Converge to consensus* with *'Planning poker'* with value 1.5.
   - The value of the attributes is the average of the weighted values of all associated activities. The activities associated with *"Backlog quality"* are *Estimate sprint story* and *Estimate tasks*; their weighted average is then 1.8. Similarly, the activities associated with *"Delivery on time"* are *Estimate tasks* and *Converge to consensus*, and their weighted average is 2.2.
   - The value of the characteristics is the average of the associated attributes. Similar to the way other levels are calculated, the final value for the *Customer value* characteristic of the *Agile Sprint Planning* process is 2.0.

According to the value obtained in this step, the process engineer may decide one of the following three options, as stated in Figure 1:

- The process is good enough, so it will be applied in the project as it is.
- The process may be improved, and some changes may be considered:
  - Change the process
  - Change the context

These two last options are described in the following sections.

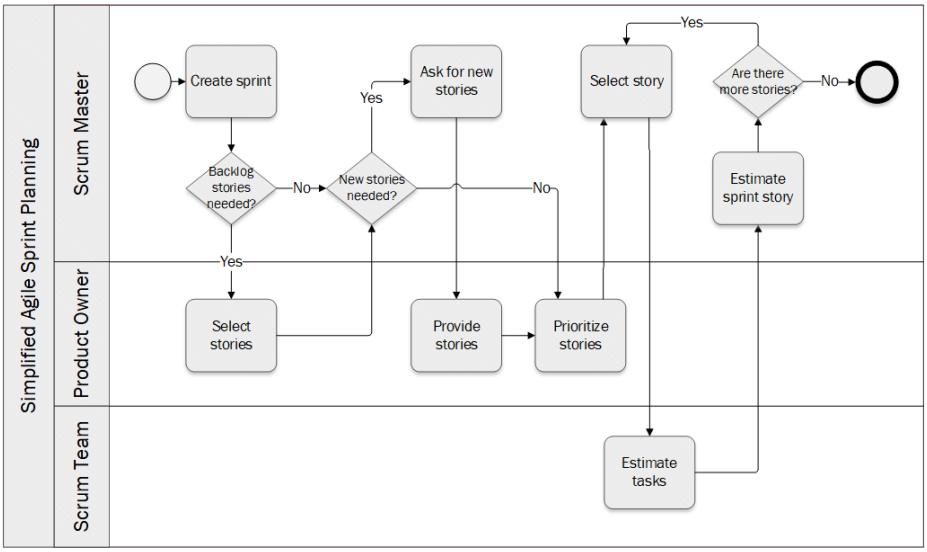

Fig. 5. Simplified Agile Sprint Planning process

3.3 Change the process

The evaluation of the project-specific process could eventually give a low value for the characteristic and some insight about the influence of the practices. So, the evaluation suggests some practices could be changed. Changing software processes includes activities, artifacts, and roles, technologies supporting process execution, and integrating these definitions with technologies.

For instance, if the characteristic *Customer value* is not satisfactory, the process engineer can change *Agile Sprint Planning* process in different ways. In our running example, the activity *Converge to consensus* could generate too many iterations to the *Scrum Team*; it is possible to keep them responsible for executing *Estimate tasks* but removing *Converge to consensus*.

The process is simpler now, and the Scrum Master must resolve the consensus problem according to her/his criterion. Figure 5 depicts the changed process identified as *Simplified Agile Sprint Planning*. As part of DynaTail's cycle, this process is now the input of the tailoring step. As the context did not change (experience is still 'experienced'), the tailoring rule condition does not hold, and thus the process remains the same.

These changes are evaluated again to determine if the new process resulted in an improvement with respect to the previous project-specific process. Figure 6 shows the process evaluation without the *Converge to consensus* activity. Now the value for *Customer value* changed since practices associated with *Converge to consensus* are not considered now. So, *Customer value* characteristic has 1.9 as value.

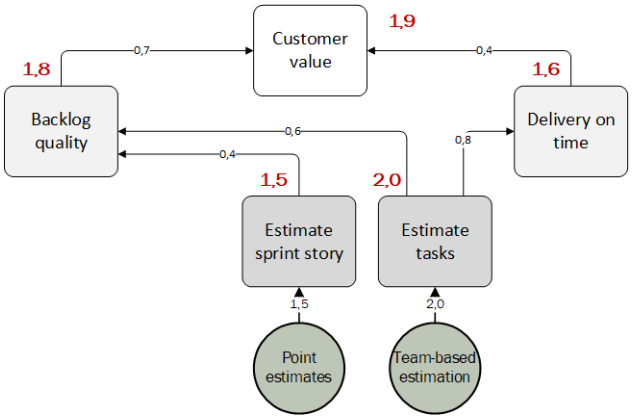

Fig. 6. Simplified Agile Sprint Planning process evaluation

3.4 Change the context

The evaluation process could eventually give a low value for the characteristic and some insight about the influence of weighted values. So, the process engineer may consider changing the project context. Provided that tailoring decisions are defined in terms of contextual factors, the process tailoring activity may result in a different process as a consequence of context change.

The new derived process could be better than the previous process for achieving the intended characteristic. If this is so, the process engineer may negotiate new resources represented by the newly changed context to achieve better results.

In our running example, the process engineer may consider changing the 'experienced' Scrum Team for an 'inexperienced' team and taking all estimation responsibility from them (see Table 1). In this case, it is necessary to have the Scrum Master execute a more complex activity *Plan sprint story* that includes both estimation and planning. These changes are already represented in the tailoring rule shown in Rule 3.1, and the new process resulting from its application is the *Hybrid Sprint Planning* process depicted in Figure 7.

Then, when this process is re-evaluated for determining its *Customer value*, we found that it is worse with respect to the previous project-specific process. For this process, running the process evaluation again, we obtain 0.7 for *Customer value* that is lower than the previous evaluation (see Figure 4). At this point, the process engineer could decide that he/she has enough information to decide. *Agile Sprint Planning* will probably be the best choice for an 'experienced' team, although he/she may choose the *Simplified Agile Sprint Planning* since it achieves only a slightly lower evaluation.

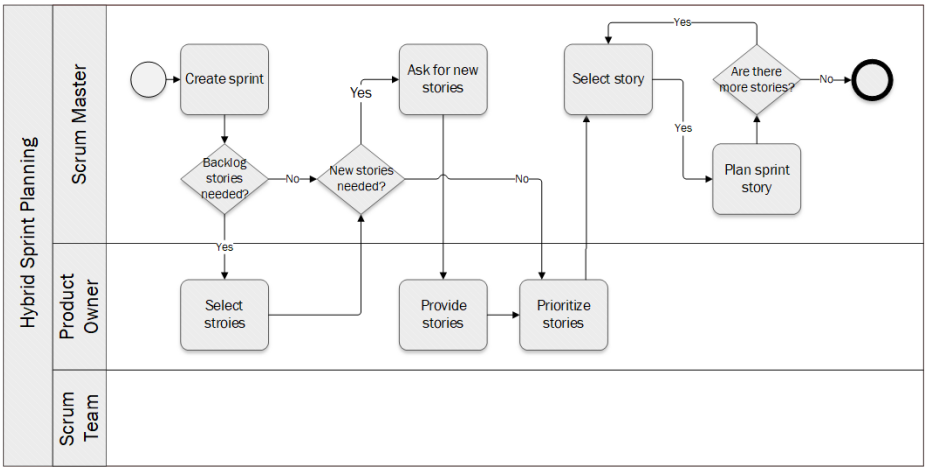

Fig. 7. Hybrid Sprint Planning process

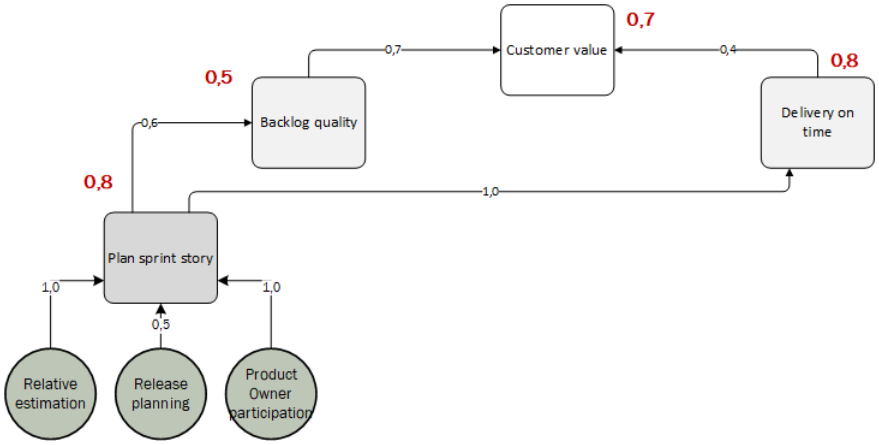

Fig. 8. Hybrid Sprint Planning process evaluation

4 Conclusion and Future Work

This work presents the DynaTail method, a dynamic approach for tailoring hybrid software processes to a given project context to achieve a specific goal. The method is illustrated with a running example from a medium-size Chilean software company. It shows the *Sprint Planning* process in different scenarios, how it is evaluated according to the goal, and the improvement steps that may be followed. We were able to demonstrate that the method can support the process engineer's work in choosing the most appropriate process for a particular project. Although our running example nicely illustrates the method, it is still quite simple to show some more sophisticated cases, such as having practices

that positively influence a certain activity and a negative influence on another one. Also, the case when we need to improve more than one characteristic simultaneously needs some more work.

We are currently formalizing all elements in the method: processes, models, and rules. On the one hand, BPMN has a defined metamodel that allows specifying and manipulating processes. On the other hand, we already have experience in defining context models and implementing process tailoring rules [7, 18], as well as building tools on top of these formalisms [17].

However, there are still some challenges that need to be addressed. We understand that the weights assigned to each arch in the influence graphs are based on expert knowledge derived from experience within the same organization. But we now need to check if processes with a high evaluation in DynaTail actually outperform in practice those with a lower evaluation.

References

1. Alaidaros, H., Omar, M., Romli, R.: The key factors of evaluating agile approaches: A systematic literature review. International Journal of Supply Chain Management **8**, 954–964 (2019)
2. Ambler, S.W., Lines, M.: The Disciplined Agile Process Decision Framework. In: Software Quality. The Future of Systems and Software Development, 8th International Conference, SWQD 2016. LNBIN, vol. 238, pp. 3–14. Springer (2016)
3. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology **54**(5), 433–447 (2012)
4. Diebold, P., Zehler, T.: The agile practices impact model: idea, concept, and application scenario. In: Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015,. pp. 92–96. ACM (2015)
5. Geras, A., Smith, M.R., Miller, J.: Configuring hybrid agile-traditional software processes. In: Extreme Programming and Agile Processes in Software Engineering, 7th International Conference, XP 2006. LNCS, vol. 4044, pp. 104–113. Springer (2006)
6. Günsel, A., Açikgšz, A., Tükel, A., Öğüt, E.: The Role Of Flexibility On Software Development Performance: An Empirical Study On Software Development Teams. Procedia - Social and Behavioral Sciences **58**, 853–860 (2012)
7. Hurtado Alegría, J.A., Bastarrica, M.C., Quispe, A., Ochoa, S.F.: MDE-based process tailoring strategy. J. Softw. Evol. Process. **26**(4), 386–403 (2014)
8. Klünder, J., Karajic, D., Tell, P., Karras, O., Münkel, C., Münch, J., MacDonell, S.G., Hebig, R., Kuhrmann, M.: Determining context factors for hybrid development methods with trained models. In: International Conference on Software and System Processes, ICSSP'2020. pp. 61–70. ACM (2020)
9. Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E., Prause, C.R.: Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In: Proceedings of the International Conference on Software and System Process, ICSSP'2017. p. 30–39. ACM (2017)
10. Meidan, A., García-García, J.A., Ramos, I.M., Escalona, M.J.: Measuring Software Process: A Systematic Mapping Study. ACM Comput. Surv. **51**(3), 58:1–58:32 (2018)

11. Münch, J., Armbrust, O., Kowalcyzk, M., Soto, M.: Software Process Definition and Management. Springer-Verlag, Germany (2012)
12. Pedreira, O., Piattini, M., Luaces, M.R., Brisaboa, N.R.: A systematic review of software process tailoring. ACM SIGSOFT Softw. Eng. Notes **32**(3), 1–6 (2007)
13. Perkusich, M., Soares, G., Almeida, H., Perkusich, A.: A procedure to detect problems of processes in software development projects using Bayesian networks. Expert Systems with Applications **42**(1), 437–450 (2015)
14. Prenner, N.: Towards Improving the Organization of Hybrid Development Approaches. In: International Conference on Software and System Processes, ICSSP'2020. p. 185–188. ACM (2020)
15. Prenner, N., Unger-Windeler, C., Schneider, K.: How are Hybrid Development Approaches Organized? A Systematic Literature Review. In: Proceedings of the International Conference on Software and System Processes. pp. 145–154 (2020)
16. Raharjo, T., Purwandari, B.: Agile Project Management Challenges and Mapping Solutions: A Systematic Literature Review. In: Proceedings of the 3rd International Conference on Software Engineering and Information Management, ICSIM '2020. p. 123–129. ACM, New York, NY, USA (2020)
17. Silvestre, L., Bastarrica, M.C., Ochoa, S.F.: A Usable MDE-based Tool for Software Process Tailoring. In: Proceedings of the MoDELS 2015 Demo and Poster Session co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems, MoDELS'2015. vol. 1554, pp. 36–39. CEUR-WS.org (2015)
18. Simmonds, J., Perovich, D., Bastarrica, M.C., Silvestre, L.: A megamodel for Software Process Line modeling and evolution. In: 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS'2015. pp. 406–415. IEEE Computer Society (2015)
19. Vijayasarathy, L.R., Butler, C.W.: Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? IEEE Software **33**(5), 86–94 (2016)
20. Wysocki, W.: A hybrid software processes management support model. In: Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES-2020, Virtual Event, 16-18 September 2020. Procedia Computer Science, vol. 176, pp. 2312–2321. Elsevier (2020)
21. Włodarski, R., Falleri, J.R., Parvéry, C.: Assessment of a hybrid software development process for student projects: a controlled experiment. In: Accepted to the 43rd International Conference on Software Engineering, ICSE'2021 (2021)