Strengthen Process Debt Identification through Process Assessment Standards

Giuseppe Lami and Francesco Merola

CNR - Istituto di Scienza e Tecnologie dell'Informazione
Pisa, Italy
{giuseppe.lami, francesco.merola}@isti.cnr.it

Abstract. Process Debt (PD), a concept derived from the Technical Debt, consists of a sub-optimal activity that might have short-term benefits but generates a negative impact in the medium-long term. PD identification is a key phase of PD management as it aims to determine the type of PD, where it is located, and how to estimate its impact. However, PD identification is the most challenging phase, as practitioners find it the most effort-intensive, mainly due to the immaterial nature of the process. In this paper, to mitigate the difficulties of PD identification, we propose a methodology that relies on frameworks compliant with the ISO/IEC 33000 family standard, the reference standard for process assessment. We also provide an exemplar application using the data from a process assessment performed using the Automotive SPICE, a framework compliant with the ISO/IEC 33000 requirements. An excerpt of the case study results is presented, showing the potential for a systematic and effective approach to PD identification.

Keywords: Process Debt Identification, Process Assessment, ISO/IEC 33000, Automotive SPICE.

1 Introduction

The importance of the quality of the process for software organizations to significantly improve the quality of their products has been a consolidated concept for decades [1]. Defining, documenting, and deploying the software process allows companies to measure current performance, identify areas of weakness, and initiate improvement actions. Nowadays, having a defined and documented software process, together with the appropriate techniques and tools to measure its effectiveness, is generally considered a key element for producing quality software products [2]. A vast body of literature deals with principles and techniques related to Software Process Assessment and Improvement (SPAI), along with a variety of related models and standards [3], [4]. These models and standards provide mechanisms to measure process indicators to identify and possibly solve process weaknesses.

Through a financial metaphor, the concept of Technical Debt (TD) identifies sub-optimal technical implementations that give a short-term benefit but create the conditions for long-term interests to pay [5]. More recently, the concept of Process Debt (PD)

has been introduced as a derivation from the original concept of TD. PD is generally classified as a Non-Technical Debt (along with social and people debts) [6], but the key concepts and the approach to managing TD can also be applied to PD.

PD consists of an activity performed in a sub-optimal way that might have short-term benefits but generates a negative impact in the medium-long term. The basic concepts related to the TD metaphor, such as the principal, the interests, and the debt, apply to PD too; what differs is the origin of the debt: in the case of PD, the debt is not a sub-optimal technical implementation as for TD, but it is a sub-optimal process.

SPAI and PD Management (PDM) are different from a conceptual point of view, but they can be approached synergically from a pragmatic perspective. In particular, the SPAI discipline is more mature and supported by methodologies, schemes, and standards, while the PDM is still much more immature. We are interested, starting from a clear understanding of the differences and similarities between these two disciplines and the related key concepts, in exploring the extent to which they can support each other. Despite a significant amount of research activity devoted to analyzing and solving issues related to the TD identification phase [7], the issues related to the identification of PD are not investigated in the literature.

The purpose of this paper is to define a methodology that allows PD identification in software projects in a systematic way, taking advantage of the existing reference frameworks and standards for process assessment and improvement. We rely on the existing reference standard for process assessment (the ISO/EC 33000 family [8]), and we apply the defined methodology in the automotive field using Automotive SPICE [9] (a framework compliant with the ISO/IEC 33003 [12], and ISO/IEC 33004 [10]).

This paper aims to contribute to answering the following research question: How can Process Assessment outcomes support PD Identification and Documentation?

This paper is structured as follows: Section 2 provides background information and related works on Process Debt, starting from the originating metaphor of Tech Debt. In Section 3, we discuss the differences and similarities between PD and process weaknesses detected by process assessments. Section 4 describes an approach based on the ISO/IEC 33000 family standard to support the identification of PD. In Section 5, we present an exemplary application of the methodology, and we discuss the related outcomes. Finally, in Section 6, conclusions and future research directions are provided.

2 Background and Related Works

In this section, we recall key concepts of SPAI, discuss relationships between TD and PD, and present the relevant literature on PD.

2.1 Software Process Assessment and Improvement

Process Assessment is a mature discipline that is essentially based on a bottom-up approach that uses work products (i.e., the results of the execution of a process) to rate low-level process indicators; rated process indicators are then combined, according to predefined schemes and rules, to achieve a rating of process attributes. The ratings of

process attributes are, in turn, combined, based on a measurement framework, to assign a quantitative rating to the process. Compliant and valid process assessment shall provide justification for any rating, particularly those related to process attributes that have achieved a sub-optimal rating score. Moreover, all the work products used for the assessment shall be precisely identified.

Although several process assessment frameworks exist (one of the most relevant is the CMMI [3]), today, the reference approach for process assessment is given by the ISO/IEC 33000 family standard [8], which provides a process assessment model and relies on a process reference model. In the following, we provide an overview of the ISO/IEC 33000 to introduce the key concepts and terminology.

The key elements of the overall model for a software process assessment according to the ISO/IEC 33000 are the Process Reference Model (PRM), Process Assessment Model (PAM), and Measurements Framework (MF).

The ISO/IEC 33004 [10] standard provides requirements for a PRM. A PRM shall be composed of a set of definitions of interrelated processes. The defined processes belonging to such a model are expected to be part of a lifecycle. The definition of each process belonging to a PRM shall contain the statement of the process purpose that describes, at a high level, the overall objectives of performing the process, together with the set of expected process outcomes demonstrating the successful achievement of the process purpose.

A MF for process assessment is a scheme that can be used to assign a quantitative rating to a quality characteristic of a process. Despite the reference quality characteristic being process Capability (a synonym of Maturity), the Measurement Framework has been conceived to be general and applicable to different process-related quality characteristics [11]. In fact, the ISO/IEC 33003 [12] provides the requirements for defining an MF for any process quality characteristic. Given a process quality characteristic Q, the measurement framework for measuring Q is composed of:

- Q Levels: a value on a n-points ordinal scale that gives a quantitative measure of the Q of a process; each level builds on the Q of the levels below.
- Process Attributes: a measurable element of the quality characteristic Q of a process, applicable to any process.
- Process Indicators: they are the basic elements of the assessment framework consisting of practices to perform. The degree of achievement of such practices is measured.
- Rating Scale: a set of values or categories to which process attributes and process indicators are mapped.

Finally, the PAM is a model for assessing, based on the MF and the PRM, a quality characteristic of processes. According to ISO/IEC 33004, the PAM is composed of process indicators related to the processes to assess (the processes are taken from the PRM). The extent to which these process indicators are fulfilled is evaluated using evidence from real projects, and a rating is then assigned to them according to the MF.

The MF for process Capability is provided by ISO/IEC 33020 standard [25].

2.2 From Technical Debt to Process Debt

The concept of TD allows to articulate the notion of trade-offs between the short-term benefits of delaying certain software development tasks or doing them quickly and less carefully and the long-term cost of those delays or unreliable works.

The concept of TD was introduced by Cunningham (1992) [5], who described how "Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite". Since then, the software development community has used the TD concept to represent and explain the various factors of increasing costs throughout the life of a software system [13], [14], [15]. The TD concept introduces the idea that the deployment of shortcuts in software development (i.e., getting a TD) is not necessarily negative, as a limited amount of debt can help developers speed up the development process in the short term. Similarly to risks, TDs need to be managed according to a precise stepwise approach to maintain them under control and avoid troubles in development projects. First, a sort of TD life cycle was identified, and then several techniques, tools, and methods were defined and implemented to support each stage of the TD life cycle [7], [16]. A wide study on TD management is provided by [17]. As demonstrated by the abundant literature, TD has obtained a relevant interest both from researchers and practitioners. Moreover, the underlying TD metaphor has been used and extended to explore its applicability to other contexts that are not restricted to technical issues only (such as those related to code, tests, and architectural design). Different kinds of debts have been identified and studied to understand the suitability of the TD management approach. One of these is the Process Debt (PD).

PD is defined as a sub-optimal practice that might have short-term benefits but generates a negative impact in the medium-long term. Following the terminology from the original TD metaphor, a PD is composed of a debt (i.e., the divergence from an optimal process), it determines interests (i.e., the negative effects generated by the occurrence of the debt) and is associated with a principal (i.e., the cost of changing the process to avoid or repay the debt) [18]. To be managed and taken under control, PD also needs proper management.

In the next sub-section, we present and briefly discuss the relevant literature on Process Debt, both from a conceptual and a practical perspective.

2.3 Process Debt Related Works

In the literature, studies dealing with Process Debt are still limited. PD is addressed in [6] by a systematic mapping review of software engineering to identify the causes of Non-Technical Debts (process, social, and people debts) and the associated prevention strategies. A number of primary papers related to PD are identified and selected in that mapping review, which demonstrates that the research on PD needs to be further developed and consolidated. According to the outcomes of [6], the causes of PD may be different, but several of them are directly related to unsuitable defined processes, process divergence, lack of project management, and lack of follow-up assessments. That study identifies 33 causes of PD and categorizes them into three classes (organizational challenges, process divergence, and external dependencies).

In [18] an exploratory study on the concept of PD based on interviews is described. The study defines PD and proposes a related reference conceptual model. They also address the relationship between TD and PD. Authors of [18] also provide a taxonomy of the types of process debt derived from the results of the exploratory study, along with an investigation into the causes that can lead to PD and the consequences of PD. The types of PD identified in [18] are shown in **Fig. 1**. The Activity-specific debt relates to any activity that is performed in a flawed way by an organization (e.g., weak effort estimation leads to an estimation debt). The Roles debt is related to mismatching roles and responsibilities. This type of PD occurs when an activity is executed by someone with a different responsibility according to the defined process. This may lead to confusion and conflicts in the development project. Dependencies between different processes can determine the need for synchronization.
Thus, the Synchronization debt type is related to deficient synchronization among different processes that need to be performed synergically or according to a precise schedule due to a defined workflow. The Infrastructure debt may arise from inadequate tool chain or technological support with respect to the complexity or characteristics of the development project. The Documentation debt is related to an insufficiently documented process. A process to be spread among and assimilated by the process stakeholders shall be clearly and completely defined and documented. Finally, the Unsuitable Process debt is related to a process defined and applied but not aligned with the organization's business needs.

The investigation reported in [18] includes a set of main PD accumulation patterns: Sub-optimal process design (a process is not fully adequate for all the stakeholders or for some specific activities); Process divergence (the process is suitable for the business needs of the organization, but some stakeholders do not comply with it); Infrastructure deficiencies (the infrastructures do not provide adequate support to the process performance).

The study in [19] analyzes the topics discussed in software development teams' agile retrospectives and shows that several PDs are addressed in such retrospective meetings. This evidence confirms the relevance of PD in software development projects.

Finally, some studies report that the causes of TD are insufficient processes [20] and that effective TD management cannot be achieved if process debt is not tackled [21].

3 Process Weakness vs. Process Debt

Process weaknesses and PDs are two similar but not equivalent concepts. A process weakness can be defined as a practice (or sequence of practices) that is not adequately or inefficiently deployed with respect to the achievement of the defined organization’s objectives or a process divergence [6] with respect to a defined reference process or a regulatory standard. As process weaknesses and PDs are both based on a sub-optimally deployed practice, it is necessary to provide a deeper argumentation to clarify the difference between these two concepts. The literature doesn’t provide a conclusive direct contribution to clarify this point. Thus, we provide a key for understanding and clarify-

ing the differences between PD and process weaknesses. From a conceptual perspective, what distinguishes a PD from a process weakness is the scope, timing, purpose, and involved roles for its resolution. The scope of a process weakness is the overall organization process. In other words, a process weakness represents a bias for the overall organization, and its resolution is a way to improve the performance of the organization in terms of costs, time, and quality of the final products. The scope of the PD is instead the specific project where it is detected. A PD, for its nature, is project-specific and cannot be generalized. In fact, a PD is a sub-optimal activity that may cause harm in a specific project, but the same may be justified in another one as it doesn't cause any harm or can even give a benefit. The way a PD is treated depends on project-specific considerations and aims at leading the project to an acceptable conclusion under the actual circumstances. Consequently, the lifecycle of a PD usually spans within the project timeline. Conversely, the lifecycle of a process weakness can overcome the specific project lifetime. The roles involved in the management of a process weakness and a PD are different as well. In the first case, as a general organizational perspective is required, the key roles are process manager and top engineering managers; in the latter, the project manager is the key actor, as a deep knowledge of the characteristics and dynamics of the project is necessary. In fact, what drives the two types of management is the scope of the evaluation of the impact, the overall organization in the first case, and the specific project in the latter.

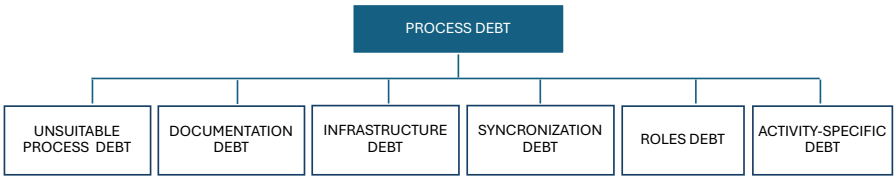

Fig. 1. Taxonomy of types of PD

4 A Methodology to Support PD Identification

TD Identification and Measurement are essential activities to determine the type of TD, where it is located, and how to estimate its impact on the software [7]. These phases received the most attention in the literature [17], [22]. Moreover, mainly for TDs that are directly related to code, they are supported by several tools (such as code analyzers), that can detect sub-optimal coding and calculate metrics. Nevertheless, despite being deeply studied and technically supported, they are considered the phases in which there is greater difficulty in achieving [23], and, according to practitioners, most of the effort spent managing TD is devoted to understanding and measuring it [24].

In the case of PD, the identification and measurement phase are even more difficult than TD. In fact, the process is generally harder to analyze and measure than a technical artifact due to its immaterial nature. To our understanding, there is a lack of specific techniques, tools, and methods for the identification of PDs.

In the following, we describe a methodology aimed at contributing to moving from an informal and accidental to a more systematic and methodologically sound way to identify PDs. As the identification of TD derives from a technical analysis of software artifacts (code, architecture, requirements, …), similarly, the identification of PDs needs to be derived from structured and reliable process assessments able to detect sub-optimal processes. Gathering information on process sub-optimal deployment through informal meetings or even interviews with stakeholders cannot be considered fully suitable as it lacks repeatability, is subjective, and is not based on quantitative evaluation. Thus, we propose to apply existing and mature process assessment models based on standards to detect process weaknesses that might determine PDs.

To address the research question defined in Section 1, we describe, in the following sub-section, a methodology to identify PD based on the performance of process assessment compliant with the ISO/IEC 33020 standard [25].

4.1 Methodology Description

The whole methodology relies on performing an ISO/IEC 33000-compliant process assessment. To perform such assessments, selecting or defining a Process Assessment Model (PAM) according to the ISO/IEC 33004 clauses is necessary.

As mentioned in Section 2, the process assessment is a bottom-up activity starting from the rating of process indicators, each associated with a specific process attribute. In process assessments, indicator ratings are combined according to the measurement framework to assign a rating to process attributes. In turn, the process attribute ratings are combined to establish the overall process rating. Notice that the process assessment results shall include not only the ratings of the process indicators but also the objective justification of any possible downrated process indicators (i.e., process indicators that don't reach the top score). Thus, using the assessment results, it is possible not only to identify process weaknesses through down-rated process indicators precisely but also to gather essential information to analyze the detected process weaknesses.

Once the assessment results are available, the process weaknesses corresponding to downrated process indicators are analyzed to understand whether they are candidates to become PDs. Eventual identified PDs are then collected and reported.

Fig. 2 represents the flowchart of the methodology described above. Notice that the PD identification can be performed both after the process assessment (based on the availability of all the assessment results) and during the assessment itself (as the rating of single process attributes is made).

One of the key phases for the application of the proposed methodology in practice is the identification of candidate PDs, starting from detected process weaknesses. The pivotal aspect of that is the availability of clear criteria to determine whether a process weakness can trigger a PD or not. In the following, we address such an aspect.

Based on the considerations provided in Section 3, the criteria to identify candidate PDs from detected process weaknesses are schematically reported below:

- Can the detected process weakness determine a significant delay in the project schedule?

- Can the detected process weakness determine significant extra costs with respect to the project budget and thus affect the expected profit margin?
- Can the quality of the final product of the project be affected if the process weakness is not resolved?

If the answer to one or more of the following questions is yes, then the process weakness is a candidate to be a PD for the project. Otherwise, the process weakness can be managed in the context of a process improvement campaign that usually follows the process assessment. The identification of the candidate PD is followed by the analysis phase, in which the principal and the interests are determined or estimated.

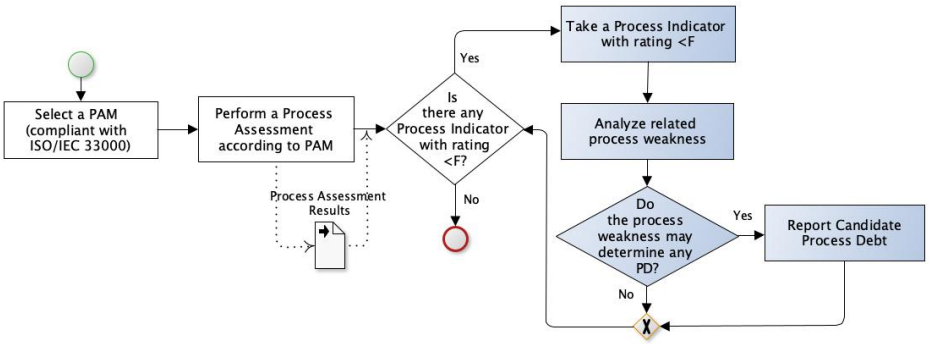

Fig. 2. PD Identification methodology flow chart

5 An Exemplar Application in Automotive

In this section, we provide an exemplary application of the methodology defined in Section 4. For this purpose, we use the data collected during an Automotive SPICE assessment performed recently. This way, we show how the process indicators provided by the applied process assessment standard can be used as a reference to identify, in a disciplined manner, under-optimal process solutions adopted.

In the following, we briefly introduce Automotive SPICE as an ISO/IEC 33004 compliant model for performing conformant assessments of the process capability in the development of embedded automotive systems. Then, we present an excerpt of significant exemplar cases demonstrating the ability of the methodology defined in Section 4 to identify candidate PDs starting from the evaluation of process indicators. Finally, we will discuss the results and the effectiveness of the proposed methodology.

5.1 Automotive SPICE: The Reference Standard for Process Assessment and Improvement in Automotive

Automotive SPICE [9] (ASPICE) provides a process framework that disciplines the software development activities at a high level of abstraction and allows their process capability assessment to match predefined sets of numerous process requirements.

ASPICE, as a de-facto process standard, is used by car manufacturers to push software process improvement among suppliers of software-intensive systems [26]. Process capability is defined as a characterization of the ability of a process to meet current or projected business goals. ASPICE is compliant with the ISO/IEC 33004 and ISO/IEC 33020. Many car manufacturers worldwide use this standard to qualify suppliers by requiring them to achieve defined capability levels on specific processes [26]. Due to space limitations, we do not provide further details on ASPICE, for further information on the structure and usage of ASPICE we invite readers to refer [27].

Although each ASPICE assessment can define its own assessment scope, in practice, the reference ASPICE process scope is the one identified by the VDA (Verband der Automobilindustrie e.V.) [27]. It is composed of a subset of the ASPICE Process Reference Model processes, each with an expected Capability Level (CL) 2 or more. The VDA Scope is the ASPICE benchmark in automotive and the reference scope many automotive OEMs use to qualify suppliers of software-intensive car components.

We resumed the information of a recent assessment to perform the exemplary application of the methodology. Such information has been analyzed to select meaningful examples of process indicator ratings to show the methodology's application.

5.2 Exemplary application of the methodology

This sub-section provides an exemplary application of the methodology defined previously. We used the information collected during an actual recently performed ASPICE assessment to show how the ratings of the process indicators and the related information can be used to identify candidate PDs. One of the authors of this paper is the certified ASPICE Principal Assessor who led the assessment. For confidentiality reasons, assessors cannot disclose information about the organizational unit that undertook the assessment nor information about the project used as the source of evidence for the assessment. Thus, the example provided in this section will be anonymous, making any connection with the original data and information impossible. The assessment was conducted using data from a project aimed at developing a software-intensive system in the field of infotainment for mid-level automotive vehicles. The assessment took a total of five working days (one for the assessment preparation, three for information gathering through interviews and work product analysis, and one for ratings and reporting). The project used for the assessment was composed of three main iterations (called Proto A, Proto B, and Proto C) corresponding to three major releases of software. Proto A corresponds to 70% of implemented functionalities, Proto B corresponds to 100% of functionalities, and Proto C is devoted to the integration of the software with the rest of the system and final validation. The approach was fully iterative (i.e., all the software engineering processes were performed for each Proto). The assessment was performed when the Proto A release was completed, and the Proto B activities had just begun.

For space limitations, we do not discuss all the candidate PDs derived from the assessment results. On the contrary, we provide one example derived from process weaknesses associated with downrated ASPICE process indicators for each type of PD mentioned in 2.3 and shown in Fig. 1.

Table 1 contains an excerpt of the process weaknesses that caused a downrating of process indicators (i.e., process indicators that have achieved a rating lower than the maximum score). The first column contains the ID of the process weaknesses; the second column contains the ASPICE process indicator [9], and the third column contains its achieved rating in the assessment. The fourth column contains the description of the process weakness that caused the downrate.

Table 1. Examples of Process Weaknesses from Downrated Process Indicators

#	**ASPICE Process Indicator Definition**	**Rate***	**Process Weakness Description**
1	SWE.1.BP3: *Analyze software requirements.*	L	Missing evidence of specific verifiability analysis performance
2	SUP.1.BP1: *Develop a project quality assurance strategy.*	P	The independence of QA is not guaranteed, as the project manager is the person responsible for performing QA on the same project. A conflict of interest exists.
3	SWE.4.BP3: *Perform static verification of software units.*	P	Full compliance with MISRA C rules is not defined as mandatory and, thus, not achieved
4	SWE.6.BP1: *Develop software qualification test strategy, including regression test strategy.*	P	Project-specific software regression test strategy (i.e., criteria to select regression tests, responsibilities allocation, target release, …) is documented with an insufficient level of granularity.
5	MAN.3.BP2: *Define project life cycle.*	P	The project life cycle and the customer's development process are not fully consistent. In particular, the validation phase of the Proto C is not fully consistent with the release plan agreed with the customer (a delay should be expected).
6	MAN.3.BP4: *Define, monitor, and adjust project activities.*	L	The task management tool used is a Gantt Chart (MS Project), and the granularity of the activities on the chart is too large to be effectively managed. The detailed task management (effort, timing, responsibilities) is high-level and too static to represent the actual situation. Task management is performed informally through oral communication without registration.

*Rating scale: F: fully (86%-100% achievement); L: largely (85%-51% achievement); P: partially (50%-16% achievement); N: not (15%-0% achievement)

Table 2 contains the description of the candidate PDs identified from the analysis of downrated process indicators reported in Table 1. The first column contains the ID of the candidate PD corresponding to the process weakness, which has the same ID as in Table II. The second and third columns indicate the type and a synthetic description of the candidate PD. The principal of the candidate PD is briefly described in the fourth column. A preliminary generic identification of the related interests is provided in the fifth column. After the identification of candidate PDs, the analysis of interests and the principal and the related quantification of costs was made by the quality manager, project manager, and software engineering program manager. The details of such an analysis are out of the scope of this paper.

Table 2. PD Candidate Identification from Process Weaknesses

#	Type*	Candidate PD	Cost of Principal	Identified Interests
1	A	As verifiability analysis is not performed, the software requirements specification may contain not quantifiable and vague information (e.g., vague sentences or no quantitative reference values).	- an additional analysis of software requirements aimed at identifying problems in terms of verifiability involving both software engineers and software testers - corrective changes to software requirements - consistency check, through traceability, of test specifications - review of software test specifications for consistency with software requirements	Some tests may be ineffective due to the low quality of related requirements. Thus, additional effort may be needed at testing time due to difficulties in defining software test cases. Difficult traceability between software requirements and software qualification tests causes over costs at regression testing.
2	R	As the independence of QA is not guaranteed, the effectiveness of the QA itself can be weakened due to conflicts of interest. For example, some QA issues may not be duly reported or appropriately managed as considered, from the PM perspective, sources of delays for project tasks.	- appointment of an independent quality engineer for the project - review of all the closed Quality Issues by the quality engineer - re-open some quality issues and identify corrective actions to do - manage re-opened quality issues till their closure	Unresolved QA issues related to process work products determine extra costs to face ex-post inconsistencies in terms of project activities performance (responsibility, scheduling, dependencies) and in terms of work product correctness (approval flow, completeness of documents, review performance).
3	U	The quality of code may be low as several state-of-the-art coding rules may not be fulfilled.	- review of the whole code developed so far and correct violations of MISRA C rules - acquire an additional license of the QA-C tool to perform MISRA C rules check automatically - training for software engineers for the use of the tool - require MISRA C rules compliance for the next software releases in the project	Low-quality code is prone to run-time errors, with additional effort needed at the software verification stage. Additional costs in terms of code maintainability may occur. MISRA C coding rules are widely applied in automotive and sometimes are explicitly required by OEMs. Missing compliance assurance with them is a gap with respect to competitors.
4	D	Lack of information about procedural steps, targets, tools to use, or actors involved in software regression testing at the software qualification level may lead to loss of effectiveness and confusion.	- review and modify software regression strategy to increase its level of granularity in terms of criteria to select regression tests, expected documental outcomes, timing of performance of regression tests - check the software release plan for consistency with the software regression testing policy - Apply revised software regression strategy to next software releases	Software qualification regression testing is not performed appropriately, which may lead to additional costs due to defects found on the customer side or in vehicle operation
5	S	Possible inability to respond to customer needs in terms of timing or completeness of releases.	Re-plan the ongoing and future project activities (with a specific focus on validation) in terms of schedule and resources to make releases consistent with the customer milestones.	Loss of reputation and customer satisfaction. Possible penalties to pay in the case of late releases.

6	I	Lack of a unique, complete, and shared reference environment to control and manage task scheduling and synchronization may lead to a loss of control of the project activities and ineffective estimation, report, and trend analysis of effort and time.	- Acquire a task management tool (e.g., Redmine or Jira) and perform training for users - Move ongoing and planned tasks in the new environment - Use the new environment for the rest of the project	Project costs and scheduling are out of control, which leads to over costs and delays in project milestones.

*Type of PD: A: Activity-specific debt, R: Roles debt, S: Synchronization debt, D: documentation debt, I: infrastructure debt, U: unsuitable process debt

We report briefly the decisions taken about these three candidate PDs:

- The principals of PD candidates 1 and 3 have been paid through immediate improvement actions. In those cases, the principal has been considered sustainable, and the related process weaknesses with high priority. So, no PD has been taken.
- The PD candidate 4 has been taken as a PD, and the change to the software regression strategy has been started and completed in PROTO B. The repayment of the PD (i.e., the application of the revised strategy) has been planned to impact the testing activities performed in PROTO C (final qualification software testing) and not for those performed in PROTO B.
- The PD candidate 2 has been taken as a PD. After the analysis of interests, no repayment of that PD has been planned for the specific project. The repayment has been moved to the following projects after the recruitment of new quality assurance staff.
- The principal of PD candidate 5 has been paid by an immediate improvement action consisting of a replanning of the PROTO C activities to make them consistent with the release plan agreed with the customer. So, no PD has been taken.
- Finally, PD candidate 6 has been taken as a PD. The principal has been evaluated as too high to pay in the current project, as having a mixed management environment in the same project may introduce confusion and inconsistencies in task management. Thus, no repayment of that PD has been planned for the specific project. The repayment has been moved to the next projects after acquiring a state-of-the-art task management environment and completing the related training for the staff.

Specific project risks have been created and managed for PD candidates 2 and 6.

5.3 Case Study Results Discussion

In this sub-section, we briefly discuss the outcomes of the exemplary application of the methodology by highlighting the main strengths and limitations.

Completeness: The ASPICE PRM (as any other PRM compliant with ISO/IEC 33004) guarantees completeness both in terms of relevant processes for the development of software-intensive automotive systems and in terms of process indicators. In fact, as the purpose of a process reference model is to define a set of processes that collectively can support the primary aims of a community of interest [10], this leads to a consistent and comprehensive set of processes available. Moreover, each process is defined so that the set of process outcomes shall be necessary and sufficient to achieve

the purpose of the process. In other words, the process indicators address all the key elements of the processes of interest for the reference community (in this case, the manufacturers of software-intensive automotive systems). That doesn't mean ASPICE can detect all the possible candidate PDs in a project. In fact, it is possible that some PDs can be taken even if the rating of related ASPICE practices is satisfactory. Nevertheless, ASPICE provides support for the identification of PDs in all the relevant processes a project is composed of.

Coverage of PD types: Process Indicators provided by any ISO/IEC 33000 compliant model address all the PD types shown in Fig. 1. In fact, each candidate PD provided in Table III refers to a different PD type, and every PD type is represented there.

Feasibility/Flexibility: The availability of data from a third-party assessment (as in the case described in 5.2) is not strictly necessary for applying the methodology. The ISO/IEC 33000 model can be applied in a smarter way by the organization's team itself, using the process indicators as a guideline. That makes the applicability of the methodology less rigorous but much easier and more flexible.

Process culture: the application of the methodology contributes to building or strengthening a process-oriented mindset among the members of the organization. Reasoning in terms of PD (principal, interests, debt) is an effective way to get people to manage process issues with a structured, mature, and sound approach.

Threat to Validity: the exemplary application presented in 5.2 has limited validity and cannot be used to derive general conclusions about the effectiveness of the methodology in detecting the PDs in a project. Further investigations and additional applications to real cases might lead to more general conclusions. The only conclusions that we can derive from the presented exemplary application of the methodology are related to its applicability in practice.

6 Conclusions and Future Directions

Process Debt (PD), conceptually derived from the Technical Debt (TD), is intended as a sub-optimal activity that can provide short-term benefits, but it is generally damaging to a software business in the long run. The identification of PD is a pivotal phase of PD management, but it is hard to perform due to the immaterial nature of the process. PD identification shall rely on the analysis of the weaknesses in the organizational *modus operandi*; such an analysis typically lacks tool support and systematicity. In this paper, we propose a methodology to strengthen PD identification through the support of existing process assessment standards. The methodology exploits the process indicators provided by the process assessment reference standard ISO/IEC 33000 family to provide a comprehensive set of process-related aspects for PD identification.

Not all the process weaknesses identified through the evaluation of process indicators provided by the ISO/IEC33000-compliant assessment frameworks can trigger a PD. Nevertheless, we argue that they can represent a way to perform a systematic and documented identification of candidate PDs.

We provided a case study showing an exemplary application of the methodology using ASPICE as an ISO/IEC33000-compliant process assessment framework for the

development of software-intensive automotive systems. The case study shows that the downrated process indicators can be an effective starting point for the systematic identification of candidate PDs. The experience of the exemplary application of the methodology showed that, although the applied methodology is far from being able to detect all the possible candidate PDs in a project, nevertheless it makes the process weaknesses analysis and the PD identification more systematic and disciplined, and it contributes to the growth of the process management culture in the organization.

In the future, we intend to evaluate, through empirical studies, the effectiveness of the methodology and explore the possibility of implementing a supporting tool.

7 References

1. Münch, J., Armbrust, O., Kowalczyk, M., & Sotó, M. (2012). Software process definition and management. Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Ashrafi, N. (2003). The impact of software process improvement on quality: in theory and practice. Information & Management, 40(7), 677-690.
3. Software Engineering Institute: CMMI for Development Version 1.3, Carnegie Mellon University Pittsburgh Pennsylvania Technical Report CMU/SEI-2010-TR-033.
4. Schroeder, R. G., Linderman, K., Liedtke, C., & Choo, A. S. (2008). Six Sigma: Definition and underlying theory. Journal of Operations Management, 26(4), 536-554.
5. Cunningham, W. 1992. The Wy Cash portfolio management system. In: Proc. of the 7th Object-Oriented Programming Systems, Languages, and Applications. ACM, pp.29–30.
6. Ahmad, M. O., & Gustavsson, T. (2024). The Pandora's box of social, process, and people debts in software engineering. Journal of Software: Evolution and Process, 36(2), e2516.
7. Alves, N. S., et alt. (2016). Identification and management of technical debt: A systematic mapping study. Information and Software Technology, 70, 100-121.
8. Information Technology – Process Assessment, document ISO, ISO/IEC 33000, International Organization for Standardization, Geneva, Switzerland, 2015.
9. VDA QMC Working Group 13 / Automotive SIG. Automotive SPICE ver. 3.1, Process Assessment Model, November 2017.
10. ISO/IEC 33004 - Information technology - Process assessment – Requirements for process reference, process assessment and maturity models, 2015.
11. Lami, G., Fabbrini, F., Buglione, L. (2014). An ISO/IEC33000-compliant measurement framework for software process sustainability assessment. IWSM Mensura (pp. 50-59). IEEE.
12. ISO/IEC 33003 - Information technology — Process assessment — Requirements for process measurement frameworks, 2015.
13. Guo Y., Seaman C. 2011. A portfolio approach to technical debt. In Proc. of the 2nd International Workshop on Managing Technical Debt, ICSE 2011.
14. Klinger T., et al. 2011. An enterprise perspective on technical debt. In Proc. of the 2nd Int.l Workshop on Managing Technical Debt, ICSE 2011.
15. Lim E., Taksande N., Seaman C. 2012. A balancing act: what software practitioners have to say about technical debt. IEEE Software 29(6), 22–27.
16. Avgeriou, P. C., et alt. (2020). An overview and comparison of technical debt measurement tools. IEEE software, 38(3), 61-71.
17. Zengyang L., Avgerioua P, Liang Peng. 2015. A systematic mapping study on technical debt and its management. The Journal of Systems and Software 101 193–220. Elsevier.

18. Martini, A., Besker, T., & Bosch, J. (2020, December). Process debt: A first exploration. In 2020 27th Asia-Pacific Software Engineering Conference (APSEC) (pp. 316-325). IEEE.
19. Martini, A., Stray, V., & Moe, N. B. (2019). Technical-, social-and process debt in large-scale agile: an exploratory case-study. In Agile Processes in Software Engineering and Extreme Programming Workshops: XP 2019, (pp. 112-119). Springer International Publishing.
20. Malakuti, S., Ostroumov, S. (2020). The Quest for Introducing Technical Debt Management in a Large-Scale Industrial Company. In: Jansen, A., et alt. (eds) Software Architecture. ECSA 2020. Lecture Notes in Computer Science, vol 12292. Springer, Cham.
21. Holvitie, J., Leppänen, V., Hyrynsalmi, S.: Technical debt and the effect of agile software development practices on it - an industry practitioner survey. In: 2014 Sixth Int.l Workshop on Managing Technical Debt, pp. 35–42 (2014).
22. Melo, A., Fagundes, R., Lenarduzzi, V., & Santos, W. B. (2022). Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. Journal of Systems and Software, 194, 111483.
23. Besker, T., Martini, A., & Bosch, J. (2018, May). Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work. In Proceedings of the 2018 International Conference on Technical Debt (pp. 105-114).
24. Besker, T., Martini, A., & Bosch, J. (2019). Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. Journal of Systems and Software, 156, 41-61.
25. ISO/IEC 33020 - Information technology — Process assessment—Process measurement framework for assessment of process Capability, 2019.
26. Fabbrini, F., Fusani, M., Lami, G., Sivera, E. (2007). A SPICE-based software supplier qualification mechanism in automotive industry. Software Process: Improvement and Practice, 12(6), 523-528.
27. Messnarz, R., Ekert, D., Zehetner, T., & Aschbacher, L. (2019). Experiences with ASPICE 3.1 and the VDA automotive SPICE guidelines–using advanced assessment systems. In Proc. of 26th EuroSPI Conference 2019, (pp. 549-562). Springer International Publishing.