Technical Debt in Automated Production Systems

CONFERENCE PAPER · OCTOBER 2015

READS
60

4 AUTHORS, INCLUDING:

Birgit Vogel-Heuser
Technische Universität München
195 PUBLICATIONS **454** CITATIONS
SEE PROFILE

Susanne Rösch
Technische Universität München
9 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Antonio Martini
Chalmers University of Technology
11 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Technical Debt in Automated Production Systems

Birgit Vogel-Heuser, Susanne Rösch
Institute AIS
Technische Universität München
[vogel-heuser, roesch]@ais.mw.tum.de

Antonio Martini
Software Engineering Division
Chalmers | University of Gothenburg
Gothenburg, Sweden
antonio.martini@chalmers.se

Matthias Tichy
Institute of Software Engineering and Compiler Construction
University of Ulm
matthias.tichy@uni-ulm.de

***Abstract*—The term technical debt borrowed from financial debt describes the long-term negative effects of sub-optimal solutions to achieve short-term benefits. It has been widely studied so far in pure software systems. However, there is a lack of studies on technical debt in technical systems, which contain mechanical, electrical and software parts. Automated Production Systems are such technical systems. In this position paper, we introduce technical debt for Automated Production Systems and give examples from the different disciplines. Based on that description, we outline future research directions on technical debt in this field.**

***Index Terms*—technical debt; automated production systems, software intensive systems**

I. Introduction

Today's goods are produced by highly automated production systems (APS) including a variety of industry sectors such as automated packaging, pharmaceutical and medical production, food processing, automotive (e.g. production of mechanical parts), etc. These production systems are typically highly specialized and designed and built on a case-by-case basis. Furthermore, they are designed by engineers from different disciplines. Mechanical engineers design the mechanical parts of the system, e.g., robots, material handling (e.g. using pneumatics) and material transport systems. Electrical engineers are responsible for the electrical systems, e.g., distributing power as well as connecting sensors and actuators and the programmable logic controller (PLC) that contains the software controlling the production process. Finally, software engineers develop the software.

The development process of APS is influenced by the fact that they are jointly developed by the disciplines and therefore design and development happens in parallel in the different disciplines. Additionally, due to the size of the systems and the case-by-case development, the development process and the overall life cycle have additional phases compared to usual processes for technical systems, e.g., assembly (combined with system integration with the other disciplines) and startup of the system on site (see Figure 1). The long usage of APS, up to 50 years in the chemical industry, further results in a high relevance of the maintenance phase.

Technical debt (TD) has been introduced more than a decade ago [1] as a metaphor from the field of economics to describe that decisions in developing software are often trade-offs between short-term benefits and long-term problems. A short term benefit ("taking a debt") can for example result in the violation of architectural guidelines, which continually hamper the development of new features, and therefore result in the continuous paying of costs ("paying the interest"). Li et al. [2] give an overview of TD which can affect all software development phases, e.g. requirements, design, testing, etc.

So far TD has been intensively studied over the last decade in the field of software (see [2]). However, to the best of our knowledge there exist no studies of TD in mechanical and electrical parts. However, in our experience, mechanical and electrical engineers face similar challenges and are often pressed to use sub-optimal solutions for a short-term gain. Thus, we believe that the concept of TD is applicable to those systems as well [3].

In our ongoing work, we explore the application of the concept of TD to APS and try to identify commonalities with and differences to pure software systems. Currently, we are collecting architectural TD items based on previous works of a manufacturer of APS as well as other collaborations with manufacturers. In contrast to existing work in TD, we also focus on TD items in the mechanical and electrical engineering disciplines.

The contribution of this paper is a presentation of representative examples of TD in APS. We present the item itself and the discipline it originated from. Furthermore, we discuss its causes as well as effects also taking the development phases into account. Furthermore, we describe further research directions for managing TD in APS.

In the next section, we introduce the development process of APS in more detail focusing on differences to pure software systems. We systematically present the examples of TD in Section III. After a discussion of related work in Section IV, we conclude and discuss open questions for research in Section V.

II. Development Process of APS

APS consist of a technical process running on a technical system (mechanics) and additional platform components (electronics/electrics) for communication and the acquisition and processing of information. These platform components include automation computers, sensors, actuators and directly wired components to interact with the technical system. Typical communication networks used in modern APS include for instance Profibus and Ethernet (Profinet, EtherCAT and Powerlink) [4]. The operating systems of automation computers usually provide hardware management, I/O-linkage and task management. The automation functionality is mainly realized by cyclically executed control software running on automation computers under

hard real-time constraints [4]. Although there is no universally accepted engineering process or model of the life cycle, there are several suggestions for such models such as PI-Step [5], V-Model XT [6] or the Namur process model [7]. As a result of the interdisciplinary character, the life cycle of APS includes different and furthermore additional steps (see Fig. 1). When comparing Fig. 1 to software engineering process models, it becomes immediately apparent, that the three disciplines mechanical, electrical and software engineering must be regarded in parallel during the different steps. Design decisions in one discipline can affect other disciplines on a large scale. These challenges become even more complex during the different life cycles and evolution frequencies of the different disciplines [8].

An additional step that is included in the life cycle of APS is the assembly of an APS. During assembly not only the integration of one discipline, but the integration of several disciplines is achieved, making integration testing a challenging task. Other additional process steps are the commissioning and start-up of APS, which often do not happen within the APS developing company but on-site (due to the large dimensions of some APS). Furthermore, APS are usually in operation for decades before they are finally decommissioned and demolished, resulting in extended operation and maintenance phases. During operation, APS are aging as a result of physical effects such as wear out and corrosion. The result is a limited life expectancy of mechanical components, which have to be replaced after a couple of years. Usually, they are not replaced by identical spare parts, because they are no longer available or the customers prefer up to date solutions, e.g. with higher energy efficiency. The same holds for the platform components (electronics), including automation hardware and communication components, only with a shorter time horizon [4].

III. Technical Debt Items in APS

In the following, we present different TD items as a first step for identifying open research questions on managing TD of APS. For each discipline, one representative and real industrial case of an occurrence of TD in the life cycle of APS is presented. We structure the cases by presenting the context of the TD item, then the TD item and its cause followed by the principal as well as the interest (particularly also on the quality) referring to definitions by Li et al [2].

A. Mechanical Hardware

Context: During operation the isolation of a part (the coil) of an electrical motor melted due to obstructed ventilation slots of a motor (mechanic) combined with bridging of the temperature alarm beforehand or the absence of an overheat sensor in the motor. The appropriate spare part (motor) is not available in stock or in the next couple of days.

Technical Debt: The producing company hires a local workshop repair that re-coils the electrical motor. However, the repair results in lower quality as only few workshops are able to recoil certain motors with the appropriate quality. In case of APS, local workshops often lack the experience and manual dexterity in handling motor coiling and, on top of that, are missing the appropriate equipment and fixtures for the job. This potential lack in quality, is being accepted to decrease downtime making a technical compromise. Another aspect related to that, is the original TD of the design having possibly been rushed to meet the delivery date and the decision of not including a temperature sensor or bridging alarms in order to avoid production stops.

Principal: Removing the TD requires changing to a newly ordered motor of the same type.

Interest: Because the APS is running again the replacement of the poorly coiled motor by a newly ordered spare part is often neglected. The potential effect is less accurate motor behavior, overheating and, in the worst-case, fire due to overheating during the continued operation. Related to quality, the reliability is reduced and functional correctness may decrease. Depending on the severance of the stated effects it might be even required to replace the motor and the drive or a part of a machine in case of fire, which will cause a longer downtime up to several weeks.

B. Electrical Platform

Context: During detailed design of the electrical equipment the electrical engineer is often not aware of the best possible spatial positioning and mounting of the devices in the control cabinet, because this is designed by the cabinet manufacturing staff. The manufacturing staff is better aware of the mechanical dimensions and the cable conduit as well as the wiring due to long term experience. The knowledge of the design department and cabinet manufacturing department is distributed and sometimes the mounting is inappropriate. This distribution of knowledge is often cause of TD, especially due to missing procedures and a lack of (tool) support in updating engineering documents in order for the design department or the cabinet manufacturing department to detect wrong decisions in the other department.

Technical debt: An example of such a miscommunication is that the commissioners need longer cables because of an inappropriate position of the device on the mounting plate than

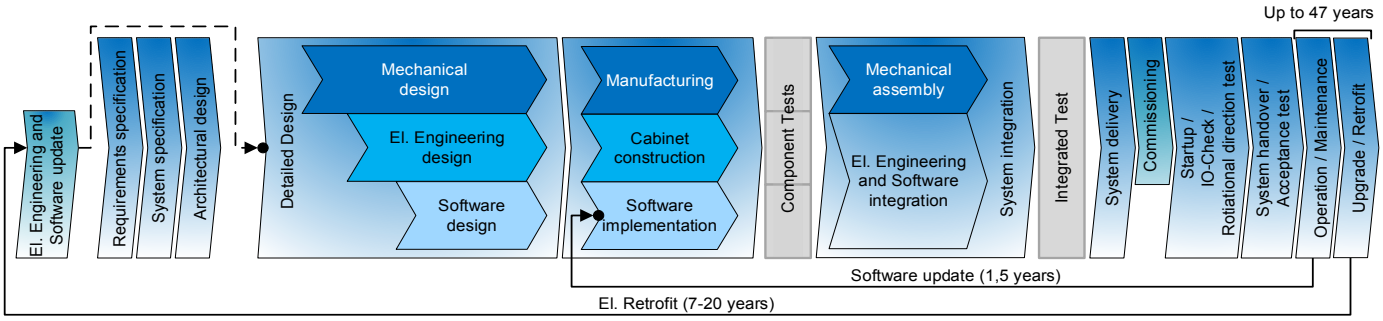

Figure 1: Life Cycle phases of APS

planned by the engineer and specified for a communication interface, e.g., cable length of RS 232 limited to 25m. The commissioners then simply take the longer cables unaware of the length restrictions.

Principal: To remove the TD, the problem must first be correctly identified and an appropriate solution for wiring and positioning must be found and applied.

Interest: The TD results in sporadic signal faults and therefore wrong positioning during manufacturing in case of a drive or wrong visualization in case of a serial interface (RS 232). It may also lead to mechanical damage during operation (overrunning an end position of a transportation device of the product). The negative effects of the decisions in one domain (electrical) on another domain (signal faults in the software) should especially be noted in this case. Functional correctness may not be guaranteed when this TD occurs. Furthermore, there may be penalties due to a later start-up as communication is not working or missing income due to frequent standstills or inappropriate products. If the wrong signals arrive in the software, even replacing entire parts of the APS due to mechanical damages might be required.

C. Software

Context: In the commissioning phase, changes in electrical and mechanical construction are more time-consuming and expensive than making changes in the software. Therefore, mistakes made in earlier life cycle phases are compensated by changing the software if possible.

Technical debt: An example is that it becomes apparent, that an error handling routine for a pneumatic cylinder is missing. The routine should observe the time from setting the actuator of the pneumatic cylinder until reaching the extended end position. If this takes too long, an alarm must be given. In the optimal case, a change request to the programmer, who developed the library function for controlling pneumatic cylinders, should be made. However, as the time pressure to start the APS is high, the commissioner implements the error handling on the next higher software architecture level that he may access, i.e., directly in the application, since the library function is not accessible for him. This results in the violation of the architectural concept for the APS (all pneumatic cylinder functions should have been encapsulated in the library). This conscious decision of avoiding proper change management and violating the architectural concept is often accompanied by a lack of documentation. As the commissioner works as fast as possible to start up the APS, she/he does not document the changes made in the software resulting in many software versions in the field.

Principal: To remove this TD, the proper change management needs to be triggered and the change properly documented.

Interest: Maintainability regarding several aspects such as modularity, reusability and modifiability is decreased. Even if the principal is properly removed in that specific plant, if the same type of plant is built multiple times, they diverge and, consequently, support and maintenance cost may become very large.

IV. Related Work

TD has recently been studied in the scientific community and, according to a recent mapping study [2], only a few cases have been conducted in industry.

Among these, most of the studies have been focusing on pure software development or at least include only the software aspect: for example [9] is an industrial case study aiming at understanding how TD was managed in agile software development teams. The study does not tackle a single TD item, but rather focuses on the overall process of managing TD and understanding what is considered interest. The study by Guo et al. [10] was also aimed at tracking TD, but also in this case the scope was limited to a pure software application (MS Exchange). Nord et al. [11] have conducted a case study in a system integrating several infrastructure disaster simulators in order to find a dependency metrics with the aim of capturing the best decision between two paths (short- and long-term focused) based on the cost of refactoring. The empirical model proposed by [12] is also based on a software system developed in Java. Other studies such as [13], [14] or [15] are also taking prioritization and management of TD into consideration.

However, all the previously mentioned studies take the sole perspective of software development: none of them is specifically related to APS, besides [3] highlighting selected challenges of TD concerning their management in relation to other disciplines (e.g. mechanical and electrical engineering).

Two recent studies [16, 17] have been performed at 7 companies developing embedded systems, including several concrete cases of TD. The results provide a taxonomy of the possible dangerous items at the architecture level, showing how the growth of TD, due to various vicious circles and the resulting interest, can become critical for the productivity and the continuous delivery of value to the customer. Although the studied systems are closer to the APS domains, the mentioned TD items do not refer to specific embedded software issues regarding, for example, the hardware aspects of the systems. Further investigation of similar cases would shed light on how the found TD items can be related to other disciplines such as electrical engineering (for example, regarding the causes or impacts of the TD items).

In summary, the current body of knowledge does not include studies in which TD occurrence and management is related to disciplines other than pure software development. This further motivates the need for a more systematic investigation of TD in APS. As shown in the mentioned examples, TD items can be found in disciplines such as electrical and mechanical engineering, and their causes and interests could have an impact on software development and vice versa.

V. Summary and Open Questions

We presented that TD also exists in APS and gave examples of TD items from mechanical engineering as well as electrical and software engineering. The presented TD is introduced in development phases specific for APS and shows preliminary results that warrant further research into TD in APS. In more detail we see the following open questions and research directions

The presented TD items have been collected from personal experience from having worked in and working with manufacturers of APS. While the identified TD items show that TD is a problem in the APS domain as well, a systematic study on TD in that domain has to be undertaken. Particularly, a broader variety and more examples of TD have to be analyzed to have a

broader sample size to extend the classification for TD. Particularly, this includes collecting data in more companies. We have to systematically analyze types of APS with different properties such as domain (e.g., chemical industry, automotive industry), type of production (mass or small series production), type of goods (fluids, big work pieces, or small electronics), size of the APS, and planned lifetime. This data collection will give a broader view on TD and differences in subgroups of them.

Based on that data collection, we can also validate whether the existing taxonomies (e.g., types and causes for TD) (e.g., in [2]) apply also in the case of APS or whether they need to be extended. Furthermore, APS are specific in various ways. First, they consist of mechanical, electrical and software elements and are therefore developed by engineers from these disciplines. An open question is how much interaction in terms of causes and effects exist between the different disciplines, i.e., how often a TD in one discipline causes an interest in another disciplines. One aspect specific to APS is for example the long times of operation of APS (up to 47 years, see Fig. 1). As some of the software is specific to the hardware in operation, software updates are often prevented due to these complex software dependencies. This causes the existence of many different software versions and variants in the field. Further investigation will enable us to understand whether TD should be mostly managed inside a single discipline or whether management of TD needs to be performed across disciplines (and often also department) borders. Second, APS have additional life cycle phases, e.g., the mechanical and electrical assembly of the system on site, where the system is built as well as the start-up phase, where the required technical function is firstly validated, both under high time pressure, or maintenance on site, where non-expert staff often performs workarounds to keep the system in operation. It could be that these specific phases are phases where much TD occurs.

Contagious debt as a new concept has been recently studied [16] and spreads through systems causing very high interests. Similarly, the borders between disciplines are interesting areas and should be analyzed to see how contagious debts spreads across them [18].

In summary, our long-term aim is to develop a management framework for TD in APS. This framework has to contain approaches for (1) identifying TD items, (2) monitoring TD items, (3) estimation of interest of the TD, and (4) prioritization of TD. While a fully-automated approach is not feasible, the steps can be supported, e.g. by improving the methods and support for documentation during commissioning and changes made compared to the original design. In particular, static analysis of control software in the APS domain would help identify TD as mentioned above using the example of error handling of a pneumatic cylinder. Nevertheless, much work will be manual by the engineers and TD needs to be communicated to the decision makers. Here, visualizations of TD can help to understand and monitor the TD as well as estimate the effects and particularly the interest. A starting point may be the work of [19], where interdependencies and connections of the different disciplines involved in APS are modeled and visualized using SysML and SysML4Mechatronics. However, the concept of TD has not yet been introduced in the approach.

ACKNOWLEDGMENT

The authors thank the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) for funding parts of this work as part of the collaborative research center 'Sonderforschungsbereich 768 – Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products' (SFB 768). We thank the Software Center at Chalmers and University of Gothenburg for funding parts of this research.

REFERENCES

[1] W. Cunningham, "The WyCash portfolio management system." Proc. Object-oriented Progr. Sys., Lang., Applicat. (Addendum), Vancouver, British Columbia, Canada. ACM, 1992.

[2] Z. Li, P. Liang, P. Avgeriou, and N. Guelfi, "A systematic mapping study on technical debt," Submiss., vol. 101, pp. 193–220, 2014.

[3] B. Vogel-Heuser, "Applicability of Technical Debt as a Concept to Understand Obstacles for Evolution of Automated Production Systems," IEEE Int. Conf. on Systems, Man and Cybernetics, 2015, accepted paper.

[4] B. Vogel-Heuser et al., "Challenges for software engineering in automation," J. Softw. Eng. Appl., vol. 7, no. 5, pp. 1-12, 2014.

[5] B. Vogel-Heuser, Ed., Systems Software Engineering. Angewandte Methoden des Systementwurfs für Ingenieure, Oldenbourg Industrieverlag GmbH, Munich, 2003.

[6] V-Modell XT Part 1: Fundamentals of the V-Modell, http://www.vmodell-xt.de, [Jun. 11, 2015].

[7] NAMUR-Empfehlung NA 35: Handling PCT Projects, 2003.

[8] F. Li, G. Bayrak, K. Kernschmidt, and B. Vogel-Heuser, "Specification of the requirements to support information technology-cycles in the machine and plant manufacturing industry," Proc. IFAC Symp. Inform. Contr. Prob. Manufact., Bukarest, Romania, pp. 1077-1082, 2012.

[9] Z. Codabux and B. Williams, "Managing technical debt: An industrial case study," Int. Workshop Manag. Tech. Debt, pp. 8–15, 2013.

[10] Y. Guo et al., "Tracking technical debt #x2014; An exploratory case study," IEEE Int. Conf. Softw. Maint., pp. 528–531, 2011.

[11] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," Joint Working IEEE/IFIP Conf. Softw. Arch. and Europ. Conf. Softw. Arch., pp. 91–100, 2012.

[12] A. Nugroho, J. Visser, and T. Kuipers, "An empirical model of technical debt and interest," Proc. 2nd Workshop Manag. Tech. Debt, New York, NY, USA, pp. 1–8, 2011.

[13] D. I. K. Sjoberg, A. Yamashita, B. C. D. Anda, A. Mockus, and T. Dyba, "Quantifying the effect of code smells on maintenance effort," IEEE Trans. Softw. Eng., vol. 39, no. 8, pp. 1144–1156, 2013.

[14] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities," Proc. 2nd Workshop Manag. Tech. Debt, New York, NY, USA, pp. 39–42, 2011.

[15] N. Zazworka et al., "Comparing four approaches for technical debt identification," Softw. Qual. J., vol. 22, no. 3, pp. 403–426, 2014.

[16] A. Martini and J. Bosch, "The danger of architectural technical debt: Contagious debt and vicious circles," Working IEEE/IFIP Conf. Softw. Arch., 2015.

[17] A. Martini, J. Bosch, M. Chaudron, "Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study," Information and Softw. Technology, 2015.

[18] Vogel-Heuser, A. Fay, I. Schäfer and M. Tichy, "Evolution of software in automated production systems - Challenges and Research Directions," J. of Systems and Softw., accepted paper.

[19] S. Feldmann et al., "A comparison of inconsistency management approaches using a mechatronic manufacturing system design case study," IEEE Int. Conf. Automat. Sci. Eng., 2015, in press.