Non-Technical Sources of Technical Debt and the Software Maintenance Readiness Framework (SMRF)

Barry Boehm, Pooyan Behnanghader
Submitted to ICSME 2018

ABSTRACT

At a workshop involving industry and government personnel on the sources of high software maintenance cost and technical debt, a number of sources were identified, discussed, and voted on, resulting in a Top-10 list of non-technical sources of technical debt. A resulting effort was made to categorize the sources as shortfalls in software maintainability planning, staffing, and technology (models, methods, processes, and tools). Subsequently, an effort was made to create for software maintenance preparation a counterpart of the 9-level Technology Readiness Level framework, to assess whether a software development project was satisfactorily progressing in its maintainability planning, preparation, and evolution. This paper summarizes the resulting Top-10 list of non-technical sources of technical debt, and a Software Maintenance Readiness Framework (SMRF) for use in project decision reviews to determine whether the project's progress in addressing its maintainability planning, staffing, and technology preparation is satisfactory. It summarizes feedback from early applications of the SMRF and from a workshop tutorial involving industry, government, and academic personnel, including an exercise in applying it to a representative project. It also summarizes current plans and progress in evolving the SMRF.

1. INTRODUCTION

The inexorable increase in software demand and the pace of changes in software technology, competition, interdependencies, and sources of vulnerability, will seriously strain the capacity of the available software maintenance labor force. Recently. a US industry-government workshop was held to address this challenge and what to do about it. A good deal of the discussion was focused on the high cost of software technical debt [1], and on ways to identify it and reduce it more quickly. During the discussions, several observations were made that a good many sources of technical debt are non-technical, and that addressing these would likely be cost-effective. In response, a portion of the workshop was devoted to identifying and prioritizing these non-technical sources of technical debt.

Section 2 presents the workshop's Top-10 list of non-technical sources of technical debt, plus a few others worth considering. Section 3 presents the resulting Software Maintenance Readiness Framework (SMRF), that identifies needed maintenance readiness levels in the three management, personnel, and technology categories at the 9 life cycle decision points from feasibility analysis through mature fielded system upgrades, and summarizes early experiences in applying it. Section 4 presents the resulting conclusions.

2. ADDRESSING NON-TECHNICAL SOURCES OF TECHNICAL DEBT

The working group addressing the non-technical sources of technical debt identified 17 non-trivial, relatively non-overlapping sources. The 12 group members were then given 20 points each to distribute across the 17 sources, and the points added up to determine the order of the Top-10. Most of the participant distributions of points were relatively flat, but some gave 5-7 points for sources they felt were particularly critical.

Here is the resulting Top-10 list of the primary sources of software process foresight shortfalls causing significant levels of technical debt.

1) Separate organizations and budgets for software acquisition and maintenance. The acquisition organization will tend to over-optimize on acquisition cost-effectiveness, leaving the maintenance organization unprepared for cost-effective maintenance.

2) Overconcern with the Voice of the Customer, as in Quality Function Deployment. [2] Delighting customers with attractive features often leads to commitments to incompatible and hard-to-maintain capabilities, which could be detected by listening to the Voice of the Maintainer.

3) The Conspiracy of Optimism. The project sponsors are competing for resources with sponsors of other projects, and will tend to be optimistic about what the project will deliver and how much it will cost. They will often try to get well by outsourcing the development to the lowest cost, technically acceptable bidder. Contractors bidding to perform the project will also tend to be optimistic about what the project will deliver and how much it will cost. A good example is the US Air Force F-22 aircraft. It proposed to deliver 750 aircraft for $26.2 billion, and when cancelled had delivered 187 aircraft for $79 billion [3].

4) Inadequate system engineering resources. The first organization to be impacted by inadequate budgets and schedules will be system engineering. The result will be exponentially-large amounts of technical debt due to poorly-defined interfaces, unaddressed rainy-day use cases and risks, and premature commitments to hopefully-compatible but actually-incompatible COTS products, cloud services, open-source capabilities, and hopefully-reusable components. The inadequate resources provide no opportunity to develop and review evidence of the feasibility (scalability, compatibility, performance, dependability, etc.) of the commitments.

5) Hasty contracting that focuses on fixed operational requirements. If budgets and schedules are tight, and the contract does not require delivery of test and debugging support, architectural descriptions, development support and configuration management capabilities, and latest release COTS products, these will not be made available to the maintainers. Even if contracted-for, these may be dropped or minimized as lower-priority needs if budgets and schedules are tight. Having a fixed-requirements contract is a source of significant delays, particularly as the pace of change in software-intensive systems continues to accelerate.

6) CAIV-limited system requirements. Often, customers' desired capabilities exceed the available conspiracy-of-optimism budgets, and a Cost As Independent Variable (CAIV) exercise is performed to prioritize the capabilities, and to include only the above-the-line capabilities in the Request for Proposals or Statement of Work. This throws away valuable information on the likely sources of future maintenance activity.

7) Brittle, point-solution architectures. The lowest-cost, technically-acceptable winning bidder will generally commit to a brittle, point-solution architecture addressing only the capabilities in the Statement of Work, minimizing development costs but again escalating maintenance costs.

8) The Vicious Circle. Even when acquisition organizations wish to include the Voice of the Maintainer and invite them to participate in defining a new system, they will often be met with apologies that the maintainers are too busy compensating for the maintainability shortfalls in their current systems to be able to participate.

9) Stovepipe systems. Having different organizations implement increasingly-interdependent systems leads to numerous clashes in coordinating changes across systems with incompatible internal assumptions, infrastructure

commitments, user interfaces, and data structures. Regional and national healthcare systems are just one of many examples.

10) Over-extreme forms of agile development, such as rejecting architectural descriptions as Big Design Up Front (BDUF) and saying You Aren't Going to Need It (YAGNI). This may be true on small projects where the developers continue into maintenance, but will be disastrous if provided to a different maintenance organization, or when the small project grows into a 50-person team coping with evolving a 500K source lines of code (KSLOC) project [4].

The 7 additional non-technical sources of technical debt received relatively small numbers of votes, but each received at least 3 votes:

- Choosing lowest-cost, technically acceptable maintenance contractor;
- Delivering systems with no-longer-supported COTS products;
- Easiest-first development problem-report closure;
- High development personnel turnover;
- High requirements volatility;
- Neglecting interoperability challenges;
- Over-optimizing performance via tightly-coupled, speed-optimized components.

3. A PROPOSED SOFTWARE MAINTENANCE READINESS FRAMEWORK (SMRF)

Several classes of organizations are fortunate enough to avoid serious problems with high maintenance cost for their more diverse and dynamic software-intensive systems. Some have developers who continue to effectively evolve the system through its life cycle. Some whose business or mission depends critically on high levels of service prepare thoroughly for maintenance; choose and support highly-capable maintenance organizations; and use results-based contracting such as Vested Outsourcing [5] rather than transaction-based or level-of-effort-based maintenance contracting.

Classes of organizations most needing to reduce high maintenance costs for their more diverse and dynamic software-intensive systems are those in which Research and Development (R&D) and Operations and Maintenance (O&M) are separately funded and managed; organizations which outsource software maintenance to lowest-cost, technically acceptable external companies, using transaction–based vs. results-based payment structures; and organizations with in-house software maintenance organizations unfamiliar with the applications area.

The concepts of Technology Readiness Levels (TRLs) [6] and Manufacturing Readiness Levels (MRLs) [7] have been highly useful in improving the readiness of systems to be fielded and operated. Table 1 provides a proposed Software Maintenance Readiness Framework (SMRF) for complementary monitoring a system's preparation to be cost-effectively maintained. Its columns are organized around the three major maintainability readiness shortfall categories of Life Cycle Management, Maintenance Personnel, and Maintenance Technology: Models, Methods, Processes, and Tools (MMPTs). In general, one would expect a major development project to be at SMRF 4 at its Project Initiation Review; at SMRF 5 at its Alternatives Analysis and Selection Review; SMRF 6 at its Preliminary Design Review; and SMRF 7 at its completion of Operational Test and Evaluation. Note that the SMRF framework emphasizes outcome-based maintenance incentives such as with Performance-Based Logistics or Vested

Outsourcing [5] at SMRF 7, and maintainability data collection and analysis (DC&A) and maintenance personnel career development at SMRF 8.

3.1 Early Evaluation Results

The SMRF was initially tested via several tutorials including an exercise to use the SMRF on a representative case study and suggest improvements, which led to several revisions. Subsequently, it has been used on over 10 milestone reviews, generally resulting in improvements in maintainability planning, maintainer participation in project activities and reviews, and identification of MMPTs needed by maintainers such as for requirements traceability, architecture definition and evolution, configuration management, problem diagnostics, technical debt analysis, and regression testing. SMRF improvements focused on strengthening levels 1-3 and 8-9. At this point, a major US Defense Department (DoD) support organization is preparing to apply it to DoD acquisition projects.

Table 1: *Software-Intensive Systems Maintainability Readiness Framework (SMRF)*

Software-Intensive Systems Maintainability Readiness Framework (SMRF)			
SMR Level	**OpCon, Contracting: Missions, Scenarios, Resources, Incentives**	**Personnel Capabilities and Participation**	**Enabling Models, Methods, Processes, and Tools (MMTPs)**
9	5 years of successful maintenance oper- ations, including outcome based incen- tives, adaptation to new technologies, missions, and stakeholders.	In addition, creating incentives for con- tinuing effective maintainability. Perfor- mance on long-duration projects.	Evidence of improvements in innovative O&M MPTs based on ongoing O&M ex- perience.
8	One year of successful maintenance op- erations, including outcome based incen- tives, refinements of OpCon. Initial in- sights from maintenance data collection and analysis (DC&A).	Stimulating and applying People CMM Level 5 maintainability practices in con- tinuous improvement and innovation such as smart systems, use of multicore pro- cessors, and 3-D printing.	Evidence of MPT improvements based on maintenance DC&A based ongoing refinement, and extensions of ongoing evaluation, initial O&M MPTs.
7	System passes Maintainability Readiness Review with evidence of viable OpCon, Contracting, Logistics, Resources, Incen- tives, personnel capabilities, enabling MPTs, outcome-based incentives.	Achieving advanced People CMM Level 4 maintainability capabilities such as em- powered work groups, mentoring, quan- titative performance management and competency based assets.	Advanced, integrated, tested, and exer- cised full-LC MBS&SE MMPTs and Maintainability other-SQ tradespace analyses.
6	Mostly-elaborated maintainability Op- Con, with roles, responsibilities, work- flows, logistics management plans with budgets, schedules, resources, staffing, in- frastructure and enabling MMPT choices, V&V and review procedures.	Achieving basic People CMM levels 2 and 3 maintainability practices such as maintainability work environment, com- petency and career development, and performance management especially in such key areas such as V&V, identifica- tion & reduction of technical debt.	Advanced, integrated, tested full-LC Model-Based Software & Systems (MBS&SE) MPTs and Maintainability- other-SQ tradespace analysis tools identified for use, and being individually used and integrated.
5	Convergence, involvement of main main- tainability success-critical stakeholders. Some maintainability use cases defined. Rough maintainability OpCon, other SC- SHs, staffing, resource estimates. Preferred maintenance organization option, incentive structures determined.	In addition, independent maintainability experts participate in project evidence-based decision reviews, identify poten- tial maintainability conflicts with other SQs. Selected developers and maintainers work out skills mixes, collaboration options.	Advanced full-lifecycle (full-LC) O&M MMPTs and SW/SE MMPTs identified for use. Basic MPTs for tradespace analysis among maintainability & other SQs, including TCO being used.

4	Artifacts focused on missions. Primary maintenance options determined, Early involvement of maintainability SCSHs in elaborating and evaluating maintenance- organization options.	Critical mass of maintainability SysEs with mission SysE capability, coverage of full Maintainability SysE skills areas, representation of maintainability SCSH organizations.	Advanced O&M MMPT capabilities identified for use: Model-Based SW/SE, TOC analysis support. Basic O&M MMPT capabilities for modification, repair and V&V: some initial use.
3	Elaboration of mission Operational Con- cept (OpCon), Architectural views, life- cycle cost estimation. Key mission, O&M, success-critical stakeholders (SCSHs) identified, some maintainability options explored.	O&M success-critical stakeholders pro- vide critical mass of maintainability- capable SysEs. Identification of additional. Maintainability-critical stakeholders.	Basic O&M MMPT capabilities identified for use, particularly for OpCon, Architecture, and Total Ownership Cost (TOC) analysis. Some exploratory initial use.
2	Mission evolution directions and main- tainability implications explored. Some mission use cases defined, some O&M options explored.	Highly maintainability-capable Systems Engineers (SysEs) included in Early SysE team.	Initial exploration of O&M MPT options
1	Focus on mission opportunities, needs. Maintainability not yet considered.	Awareness of needs for early expertise for maintainability. concurrent engineering, O&M integration, Life Cycle cost estimation	Focus on O&M MPT options considered

4 CONCLUSIONS

The increasing complexity of software-intensive systems and the rapid pace of change in technology are driving organizations' software total ownership costs more and more toward software maintenance. Examples are more, larger, and more complex software systems such as Internets of Things and self-driving vehicles; increasing needs for software dependability and interoperability; increasing software autonomy; increasing data capture and data analytics; increasing legacy software, and mounting technical debt.

However, much greater savings can be achieved by addressing three non-technical sources of technical debt due to overemphasis on initial acquisition cost-effectiveness. These include shortfalls in Life Cycle Management aspects; Personnel capabilities and participation aspects, and Maintenance Models, Methods, Processes, and Tools (MMPTs). Drawing on the successful use of the Technology Readiness Level and Manufacturing Readiness Level frameworks, this paper provides a similar Software Maintenance Readiness Framework (SMRF). Its early successful applications provide indications for enabling acquisition projects to better prepare for reduced maintenance costs and system Total Ownership Costs.

5. REFERENCES

[1] P. Kruchten, R. Nord, and I. Ozkaya. Technical debt> From metaphor to theory and practice, *IEEE Software,* 29(6):18-21, Nov 2012.

[2] Yoji Akao. 1994. Development history of quality function deployment. *The Customer Driven Approach to Quality Planning and Deployment* 339 (1994).

[3] R. Haffa and A. Datls, Learning from acquisition history, *Aerospace America,* 54 (1): 30-33, 2016.

[4] Amr Elssamadisy and Gregory Schalliol. 2002. Recognizing and responding to bad smells in extreme programming. In *Proceedings of the 24th International conference on Software Engineering*. ACM, 617–622.

[5] Kate Vitasek, Mike Ledyard, and Karl Manrodt. 2013. *Vested outsourcing: five rules that will transform outsourcing*. Palgrave Macmillan.

[6] US Department of Defense, "Technology Readiness Assessment (TRA) Guidance, April 2011.

[7] D. Cundiff. 2003. Manufacturing readiness levels (MRL). *Unpublished white paper* (2003).