Information and Software Technology 61 (2015) 71–92

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Comparing development approaches and reuse strategies: An empirical evaluation of developer views from the aerospace industry

Julia Varnell-Sarjeant [a,*], Anneliese Amschler Andrews [a], Joe Lucente [a], Andreas Stefik [b]

[a] Department of Computer Science, University of Denver, Denver, CO 80208, United States
[b] Department of Computer Science, University of Nevada, Las Vegas, Las Vegas, NV 89154-4019, United States

ARTICLE INFO

Article history:
Received 16 June 2014
Received in revised form 10 November 2014
Accepted 2 January 2015
Available online 28 January 2015

Keywords:
Empirical study
Software reuse
Embedded systems
Nonembedded systems

ABSTRACT

Context: There is a debate in the aerospace industry whether lessons from reuse successes and failures in nonembedded software can be applied to embedded software. Reuse supposedly reduces development time and errors. The aerospace industry was an early advocate of reuse, but in Aerospace, not all reuse experiences have been as successful as expected. Some major projects experienced large overruns in time, budget, as well as inferior performance, at least in part, due to the gap between reuse expectations and reuse outcomes. This seemed to be especially the case for embedded systems.
Objective: Our goal is to discover software reuse practices in the aerospace industry. In particular, we wish to learn whether practitioners who develop embedded systems use the same development approaches and artifacts as software practitioners who develop nonembedded systems. We wish to learn whether reuse influences selection of development approaches and artifacts and whether outcomes are impacted.
Method: We developed a survey given to software practitioners in a major Aerospace Corporation developing either embedded or nonembedded systems. The survey probed to identify development methods used, artifacts reused and outcomes resulting from the reuse. We used qualitative and quantitative methods such as descriptive statistics, MANOVA, Principle Component Analysis and an analysis of freeform comments to compare reuse practices between embedded systems and nonembedded systems development.
Results: We found that embedded systems were more likely to include component based development, product line development and model based development in their development approach, whereas nonembedded systems were more likely to include Ad Hoc and COTS/GOTS in their development approach. Embedded systems developers tended to reuse more and different reuse artifacts.
Conclusion: We found that, while outcomes were nearly identical, the development approaches and artifacts used did, in fact, differ. In particular, the tight coupling between code and the platform in embedded systems often dictated the development approach and reuse artifacts and identified some of the reasons.

1. Introduction

Reuse is claimed to reduce effort, cost and to increase quality [2], improve maintainability, reduce risk, shorten life cycle time, lower training costs, and achieve better software interoperability [51]. The US aerospace industry was an early advocate of reuse. Chapter 8 of [2] covers reuse in aerospace, discussing potential savings in quality, cost and productivity. The US Government has invested heavily in reuse, e.g. the Control Channel Toolkit (CCT) in 1997 and Global Broadcasting Service (GBS) beginning in 1998. Hence, one would expect large scale planned reuse. Many US government requests for proposal contain a reuse requirement including quantifying expected savings from reuse. Yet root cause analyses of Nunn-McCurdy cost breaches often cite reuse as a factor [63,7]. Further, there are those who question whether successful reuse strategies work equally well for all types of systems. Those supporting reuse cited the existence of working, already developed assets that were performing similar capabilities. Those supporting new development made the claim that when you ask a product to do what it was not designed to do, the costs of modification and maintenance exceeds the savings of reuse. Further, some claim that embedded systems are fundamentally different

* Corresponding author.
E-mail addresses: jfvarnellsarj@me.com (J. Varnell-Sarjeant), andreas.stefik@unlv.edu (A. Amschler Andrews), joelucente@du.edu (J. Lucente), andrews@cs.du.edu (A. Stefik).

http://dx.doi.org/10.1016/j.infsof.2015.01.002
0950-5849/

from nonembedded systems, and hence, successful reuse needs different strategies. Reasons include:

- When embedded systems are written and optimized directly to the target processors, and the reused products are to be run on a different platform, much of the code will have to be modified to accommodate the standard of the different platform.
- Throughput, processing, and timing of embedded systems are critical to system performance, and thus to mission success.
- Much software available for reuse is old, sometimes obsolete. It may have been written to processors that are no longer supported by the vendors. In order to make that software work, it has to be reoptimized to newer processors.
- Platforms running the software are changing faster than the software.
- When embedded are systems developed for the Department of Defense, they are not allowed to deploy with code that is not needed [14, p. 25]. This complicates reuse, as many reuse assets include more general solutions.

Others think that software is software, and the same reuse strategies can be used on either type of system. It is, hence, unclear whether reuse is different for embedded versus nonembedded systems.

The aerospace industry develops software for both embedded and nonembedded systems. Is reuse really successful currently, particularly in the aerospace domain which deals with embedded and real time systems? Specifically, does reuse effectiveness vary between embedded and nonembedded software? Do embedded systems projects employ the same development and reuse strategies? Are they reusing the same types of software and hardware artifacts? To shed light on these questions, a survey was designed to collect information on reuse success and challenges for embedded vs nonembedded systems, and to compare reuse outcomes for different development strategies and their related reuse artifacts.

The paper is organized as follows: Section 2 defines concepts such as development approaches and artifacts and what we mean by embedded and nonembedded systems. Section 3 summarizes related work. Section 4 describes the design of the study. Section 5 presents the results in the form of descriptive statistics, hypothesis testing, Principle Components Analysis and qualitative analysis. Section 6 presents a discussion of the results. Section 7 discusses threats to validity. Section 8 presents our conclusions.

2. System types, development approaches, and artifacts

Mohagheghi et al. define software reuse as "the systematic use of existing software assets to construct new or modified ones or products. Software assets in this view may be source code or executables, design templates, free standing Commercial-Off-The-Shelf (COTS) or Open Source Software (OSS) components, or entire software architectures and their components forming a product line or product family. Knowledge may also be reused and knowledge reuse is partly reflected in the reuse of architectures, templates or processes [45]."

2.1. System types

2.1.1. Embedded software systems

The term embedded systems includes cyber-physical systems. ISO defines embedded systems as "a program which functions as part of a device. Often the software is burned into firmware instead of loaded from a storage device. It is usually a freestanding implementation rather than a hosted one with an operating system [30]." They are further defined as "...integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa [40]." Examples of embedded software include device drivers, avionics, propulsion systems and robotics.

2.1.2. Non-embedded software systems

For lack of another definition, non-embedded software is defined as software not tied to the processors or inherently integrated with the physical system. Examples of non-embedded software include web applications, desktop applications, video games, and other networking applications.

2.2. Development approaches

This study examines several development approaches used in industry that use existing products. Table 1 summarizes the development approaches. In practice, project teams may use either a single development approach or a combination. The latter can happen for a variety of reasons. A later release of a product may use a more recent development method for additional functionality, for example.

2.3. Artifacts

This study also examines several artifacts often reused for developing a system. Table 2 describes the artifacts and their definitions for purposes of this paper. Note that because these are bespoke systems, sales materials, often reused in commercial applications, are not reused. Also note that because reuse of documentation was not reported, it was not included in this analysis.

3. Related work

We performed an extensive survey of literature dating back to 1992 [3] to determine what research existed comparing reuse in embedded systems to nonembedded systems, and to see whether development approach as related to reuse had been addressed. The survey of literature included quasi-experiments (there were no experiments), case studies, surveys, reviews of industry practice, metaanalyses, experience reports, and expert opinions. After an initial filtering process, we found 84 empirical studies focusing on software reuse.

The papers were classified by study type, system type and development approach. Once the papers were classified, it became clear that many reported on a particular reuse strategy or development approach, but were not discussing the value of reuse per se, nor were they performing a comparison against other development approaches or between types of systems. Because these did not add to the analysis, a threshold was established requiring that at least 20% of the paper be devoted to a discussion of the merits of reuse itself or comparison with other development approaches. While [67] find that only papers that devote at least 30% of their content to empirical results contain adequate experimentation, we determined that setting the threshold this high would exclude important data. This criterion resulted in the removal of sixteen papers since they discussed (similar to [26]) empirical results only peripherally. The removed papers, [24,29,38,53,68,32,39,75,13,35,74,27,69,6,23,28] were from the experience report and expert opinion categories. We also excluded textbooks (e.g. [36]), since their major purpose is to teach a methodology rather than to evaluate reuse success.

Finally, in 24 papers we could not determine whether the systems were embedded or nonembedded. This was disappointing, because highly regarded papers about reuse did not identify the