2014 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) TECHNICAL SESSION
AUGUST 12-14, 2014 – NOVI, MICHIGAN

The Business Case for Model-Based Software Development

Ron Lannan
LHP Software
Columbus, IN

ABSTRACT

Use of Model-Based Design (MBD) processes for embedded controls software Development has been purported for nearly the last decade to result in cost, quality, and delivery improvements. Initially the business case for MBD was rather vague and qualitative in nature, but more data is now becoming available to support the premise for this development methodology. Many times the implementation of MBD in an organization is bundled with other software process improvements such as CMMI or industry safety standards compliance, so trying to unbundle the contributions from MBD has been problematic. This paper addresses the dominant factors for MBD cost savings and the business benefits that have been realized by companies in various industries engaged in MBD development. It also summarizes some key management best practices and success factors that have helped organizations achieve success in MBD deployment.

INTRODUCTION

Model-Based Design (MBD) is a software development process in which the implementation, verification, and documentation of software features flow directly from a single graphical model of the software behavior. While the use of MBD has grown dramatically over the last 10 years, quantitative return on investment (ROI) data is nearly non-existent in the literature. While some cost savings data does exist, it is predominantly based on case studies submitted by companies to support the marketing efforts of MBD tool suppliers. The case studies typically address the cost and cycle time savings in the development process, but fall short in capturing the full software life cycle cost benefit of MBD. Arguably, the cost of quality savings due to lower in-use product defect rates, enabled by MBD, could easily dwarf the cost savings in development. This point is largely ignored in most studies. As a result, the total savings due to using the more robust MBD process are likely understated.

As we look at the MBD cost savings data and studies in the literature there is much variability in the projected savings. This is understandable for several reasons. First, the starting point for software process maturity can vary widely across companies deploying MBD. As a result, savings are likely to be much greater for companies starting from a lower CMMI capability level because MBD can inherently improve the software process and enable more continuous and seamless design, code, and testing. Second, the existence of human resources experienced in model-centric development will vary widely among companies. And lastly, there is much variability in the choice of implementations, with some being more suited to MBD than others. In general, system implementations that can be described mathematically and are physics-based are best suited for MBD. Most often, ordinary differential equations and algebraic expressions are used to define the input to output relationships of the system outputs and components.

THE MBD COST SAVINGS PREMISE

The dominant premise for MBD cost savings is two-fold: the elimination of errors early in the design stage and the efficiency and effectiveness of auto-code generation and model-driven testing. While there are many other collateral benefits of MBD, these two are the primary drivers of the business case argument for MBD.

The relative cost to fix an error in any system development is shown in Figure 1. [3]

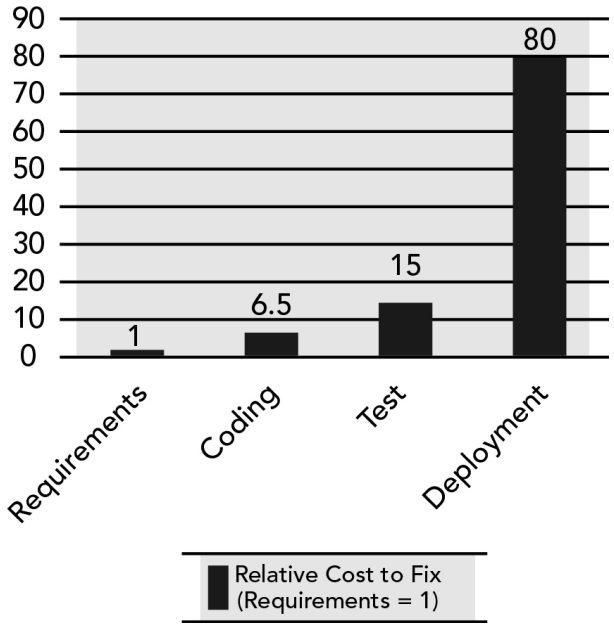

Figure 1: Defect cost penalty.

This chart shows the later in the process defects are caught, the more expensive it will be to fix them. Because MBD provides an executable model that can be tested early in the design cycle, more requirements errors can be avoided and the software model can also be used to provide more effective, intermediate, and end-to-end testing of the software system. Note that the cost of finding a problem once it is deployed is almost two orders of magnitude higher than if the problem was caught early in the design phase. Also, catching problems in the requirements phase versus the test phase has an order of magnitude cost advantage during software and system development.

The relative shift in where problems are found for MBD is shown in Figure 2. [5] Note that when we compare the MBD process to Software Engineering Institute (SEI) industry averages, most defects are caught in the requirements and design phases as opposed to the software unit test and integration phases of the non-MBD process. This positive shift in early defect discovery results in less rework and also shortens the development cycle time.

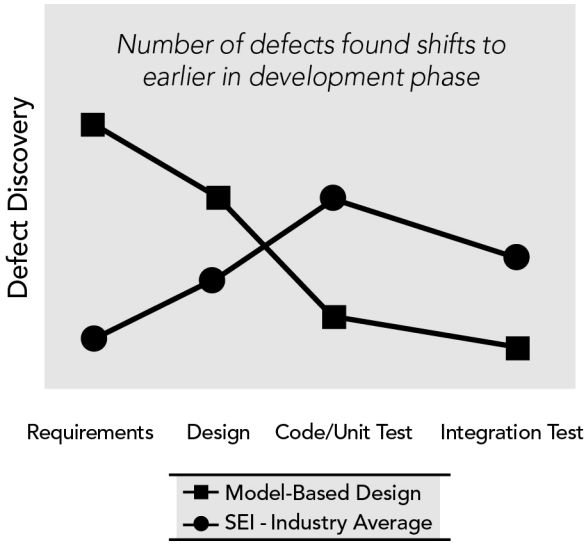

Figure 2: Defect shift pattern with MBD.

If we look at the process phases, as shown in Figure 3 [5], we can see that the MBD process results in over 50% savings in the test phase and over 30% savings in the requirements phase. There is roughly a 10% savings in coding due primarily to auto-code generation. Auto-code generation generally eliminates errors from hand coding and is able to be easily re-targeted to different hardware platforms.

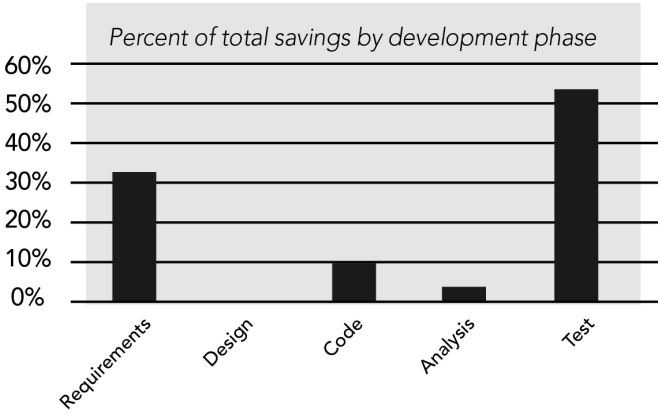

Figure 3: MBD savings by phase.

Other collateral benefits from the MBD process are identified thoroughly in another technical brief from LHP Software. [2]

MBD SAVINGS DATA

Research of the literature on MBD cost savings did not reveal any analysis that specifically showed the MBD cost advantage from a well-documented, bottom-up cost baseline. Most data was generalized and based on case studies associated with MBD tool suppliers. However, there was a study conducted [4] that was a macro-level, top down analysis of MBD savings across several embedded industry verticals. This analysis was based on responses from more than 500 embedded developers covering five market segment verticals: telecom/datacom, auto/transportation, industrial automation, medical, and military/aerospace. In his study, Krasner segmented responses and data between MBD and non-MBD developers. Interestingly, his findings showed that the relative development cost savings for MBD ranged from near 0 for industrial automation to as much as 95% for telecom/datacom. In the middle was auto/transportation with a 39% savings and medical with a 79% savings. His findings also cited the lack of cost savings for industrial automation was likely due to the continued heavy reliance on physical prototyping and the lack of complexity in the applications. The data on military/aerospace wasn't statistically valid and it was noted that the military spends less on development tools than does industry, which is one factor in the slow adoption of MBD by this segment.

A recent report coming from the automotive sector [1] explored the effects of MBD on cost, cycle time, and quality. Additionally, the report identified the key factors used by companies to optimize the economics of their MBD process. A summary of the cost and time savings analysis from this study is shown in Figure 4.

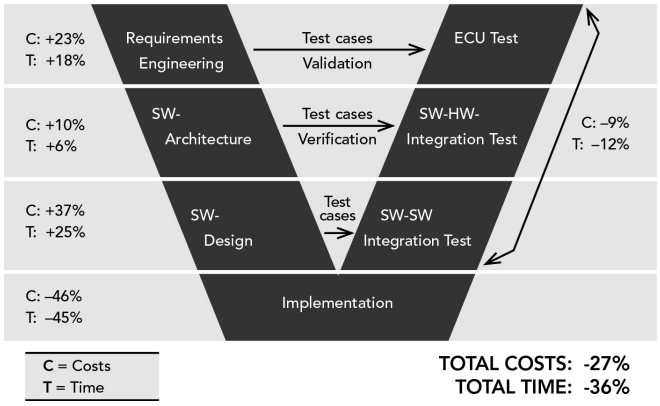

Figure 4: MBD cost and time savings by phase.

As seen from the chart, the cost and time are higher in the descending part of the V diagram (largely due to frontloading with activities like rapid controls prototyping) but lower in the implementation through verification phases. Overall this study showed an average cost savings of 27% and an average cycle time savings of 36% when considering the entire V diagram. However, there were some companies that showed an increase in cost when using MBD. This was largely attributed to companies with less than one year of experience.

The study also analyzed the top three factors that most influenced the cost savings. Companies that possessed high values in all three categories were able to achieve the highest cost savings – roughly 40%.

The factors are:

- A high degree of function modeling for the entire software system and, consequently, a high degree of generated code
- Intensive function model testing
- High capability of the employees engaged in MBD as well as software engineering.

MBD BUSINESS PRACTICES

In order to maximize success in deploying an MBD process there are several key practices that management should consider. This comes from the author's experience in leading a large controls organization in the deployment of MBD and product line architecture in a $20 billion corporation.

Identify the problems you are trying to solve. These could range from trying to improve software quality, increase productivity/capacity, improve reuse through a product line architecture approach, hit release dates more predictably, or enable worldwide distributed software. With a prioritized list of problems one can then identify which problems can best be addressed by MBD, process improvement, or other improvement based methods such as Six Sigma or product line architecture approaches.

Solicit the help of an experienced outside company to help benchmark your current process and determine how an MBD approach could be sculpted to fit within your current organizational systems. Keep in mind that companies that have no experience with MBD usually do not benefit from MBD within a year or two of deployment. This is due primarily to lack of experience and knowledge on how to avoid the pitfalls.

Ensure that the MBD improvement initiative is a major thrust of the Chief Technical Officer's strategic plan, and establish visible measures to show progress against your deployment plan. Also, ensure that you have developed a comparative cost study of the existing and new MBD process to identify where the savings will be realized.

Start with a pilot production project. The project should be selected so as to minimize financial risk in the event of schedule or quality issues with the pilot product launch. Lessons learned from the pilot project will prove invaluable and allow you to identify bottlenecks in the process, tools, or training that will be required before you move to more widespread applications. Also, it is important to select pilot project team members that have a passion for making MBD a success and have the requisite training.

Ensure that your MBD software deployment leader has the passion, energy, courage, and conviction to knock down barriers, effectively engage the organization, and communicate up and down to reinforce the strategic vision and tactical initiatives. This person is likely the difference between success and failure of the MBD initiative. It is also important that this person has a sound industry and technology perspective and participates in external forums that continually validate your company's technical direction.

Don't underestimate the 'fight or flight' risk with software engineers. The shift from hand-coding to auto-coding poses a threat to many software engineers. Some will resist and leave the organization, some will adapt and learn to become architects and software integrators, and others will embrace it and provide new ideas and system-level thinking given the higher level of abstraction afforded by MBD approaches. This MBD skill mix shift will certainly require you to re-evaluate your future mix of controls, systems, software, and test engineer needs.

THE LHP COMPETITIVE ADVANTAGE

LHP Software was a pioneer in the adoption of MBD in the automotive industry and continues to be heavily engaged with model based design and development for a variety of customers. We have employees experienced with senior management leadership in MBD deployment within multi-national companies, controls and systems engineers with both plant and controls modeling experience, software and test engineers skilled in the use of MBD processes and toolsets, and alliance partners such as National Instruments who possess products and capabilities to make modeling and function testing a powerful tool in your MBD arsenal. In examination of the three keys to achieving cost savings near 40%, LHP Software is uniquely positioned to work with companies to successfully implement these key MBD attributes throughout their organization. LHP Software provides expertise in function model creation and function model testing, as well as experienced technical and management consulting in MBD processes, which enables companies to optimize their investments in MBD.

CONCLUSIONS

The business case for MBD is coming to a clearer focus and more industry verticals are realizing the savings possible with MBD software development approaches. Some companies are successful in MBD deployment while others are not. The difference lies in being able to avoid the pitfalls, embracing a high degree of function modeling of the system, exploiting the back-end of the V model with intensive function model testing, and utilizing experienced technical and management resources to guide the MBD start-up experience.

REFERENCES

[1] Broy, M. K. (2011, March 03). Model-Based Software

[2] Development - Its Real Benefit. EE Times Europe.

Fraser, S., Fenstermacher, D., & Doyle, C. (2012). TheBenefits of Model Based Design. Columbus, IN: LHP Software, LLC.

[3] King, T., & Marasco, J. (2008). What is the Cost of a Requirement Error?

[4] Krasner, J. (2010). Comparing Embedded Design Outcomes With and Without Model-Based Design. Embedded Market Forecasters.

[5] Lin, J. (2011, May 25) Measuring Return on Investment of Model Based Design. EE Times.