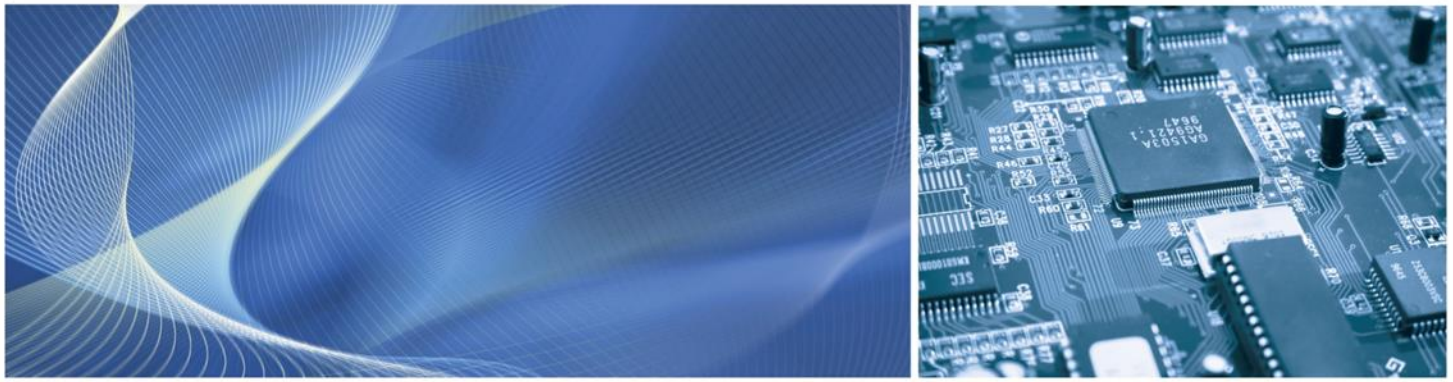
EmbeddedMarket
forecasters

WHITE PAPER



www.embeddedforecast.com

How Product Development Organizations can Achieve Long-Term Cost Savings Using Model-Based Systems Engineering (MBSE)

How financial managers can achieve lower costs of development, faster deployment of new products, and lower ongoing maintenance costs

Jerry Krasner, Ph.D., MBA

October 2015

Regarding the Data in this report

The data that is referred to in this report is *statistically accurate* and *authentic* and is based on:

- A statistically generated comprehensive and detailed survey of embedded developers and managers who reported on their design results (number of developers per project, vertical market of their design, time to market, percent of designs completed behind schedule or cancelled, closeness of final design outcomes to pre-design expectations, testing outcomes, etc.), the tools they used (development, modeling, Java, Eclipse, and other development tools), their choice of OS, IDE, communication middleware, processors used as well as where they go to learn about new products, tools and concepts.
- An EMF Dashboard – a unique tool that allows the user to simultaneously compare similar products (vendors can do competitive comparative analysis); that marketing executives can use for sales promo and strategic planning; that allows developers beginning a project to compare the experiences of hundreds of fellow developers that undertook similar projects to gain insights before making a commitment; and that allows CFOs and senior managers to look at what tools and processes resulted in the greatest cost savings.
- Data presented herein is based on 6 years of EMF Survey data totaling more than 4000 developer responses. The 2015 data, for example is based on the responses of 1334 embedded developers and managers (95% +/- 3% statistical accuracy).

For the interested reader, the following link demonstrates the power of the Dashboard and how we used it in developing the data that is presented herein:

http://www.embeddedforecast.com/emfmip_videos.php

1. Executive Summary

How can technology organizations reduce risk across the product development lifecycle? Today's technology organizations face a myriad of challenges in reducing project risk and improving project outcomes. Many companies are finding success with an approach called Model-Based Systems Engineering (MBSE). Model-based systems engineering is the application of a visual and semantically rich modeling language to the core systems engineering challenge: how to tame complexity while aligning people, process and technology around a common product vision.

When compared to typical systems engineering endeavors, the addition of model-based systems engineering delivers a 55% reduction in total development cost. MBSE approaches also streamline and enable product-line engineering. When compared to typical systems engineering endeavors, the addition of product-line engineering delivers a 41.6% reduction in total development cost.

2. Identifying, Assessing and Controlling Costs across the Life Cycle

For many technology organizations, the journey to risk reduction starts with project cost accounting. Actual project costs are measured in total development expenditures, including direct engineering costs and the cost of tooling and training. These costs, which have traditionally been driven by project complexity and engineering productivity, must also be augmented by indirect costs arising from assorted project risks, such as the risk of project failure or cancellation, the cost of incremental service, support or recalls due to poor quality, and lost revenues caused by missed windows of opportunity. Even if known risks do not come to pass, the risk must still be recognized as a cost and accounted for – perhaps in allowing extra time, contingency planning or QA. Most risks can be quantified and objectively determined if the cost of the solution is less than the cost of the potential impact of the risk. Some risks are too dramatic to quantify, such as reputational risk due to failure.

In previous publications EMF data has clearly alerted and shown CFOs and financially responsible managers that the total cost of development can be orders of magnitude greater than tool acquisition costs – particularly when dealing with complex systems. The following cost considerations are important to developers and managers alike.

- On time shipment of product – the cost of delay can be effective. In this paper we will show that MBSE developments ship faster than comparable developments not using MBSE.
- Number of engineers needed to complete project on-time. MBSE developments require fewer developers to complete developments. Also MBSE developments have fewer behind schedule completions thereby reducing the cost of development in terms of man months of effort.
- Maintaining the expected performance, systems functionality and features and schedule of the development – final design results are closer to pre-design expectations for MBSE developments.
- Achieving market windows of opportunity – MBSE developments not only get to market faster, but they can be easily upgraded to meet new market opportunities by integrating legacy software into new designs and automatically generating and deploying new code.

- Cost of software and systems re-design necessitated by changes in available hardware can be prohibitively expensive and time consuming. This cost is minimal with MBSE.
- Cost of in-field support is more effective because support personnel can more clearly understand design models and code versus just source code. MBSE minimizes the possibility of product recalls.

EMF data has shown that:

- The use of MBSE has reduced design delays and cancellations.
- The use of MBSE has significantly improved the relationship between pre-design expectations and final designs.
- SysML is the most popular graphical representation for MBSE for discrete embedded system designs.
- UML is the most popular graphical representation for MBSE for discrete embedded software designs.
- Model simulation and code generation enables faster design iterations that produce desired performance, functionality and capabilities.
- Using model simulation and code generation, design cycles are more predictable and result in faster product shipments with lower project risk.
- Using model simulation and code generation contributes significantly to a reduction in design, development and implementation costs.

EMF data has clearly shown that the total cost of development can be orders of magnitude greater than acquisition costs in appropriate circumstances when MBSE is not employed:

- On time shipment of product – MBSE developments ship faster than comparable developments not using MBSE.
- Maintaining the expected performance, systems functionality and features and schedule of the development – final design results are closer to pre-design expectations for MBSE developments.
- Achieving market windows of opportunity – MBSE developments not only get to market faster, but they can be easily upgraded to meet new market opportunities by integrating legacy software into new designs and automatically generating and deploying new code.
- Cost of re-design necessitated by changes in available hardware is minimal with MBSE.
- Cost of in-field support is more effective because support personnel can more clearly understand design models and code versus just source code. MBSE minimizes the possibility of product recalls.

3. What is Model-Based Systems Engineering?

Systems engineering is a multi-disciplinary engineering approach that ensures successful system specification, design, validation, and verification of complex products. Model-based systems engineering is the application of a visual and semantically rich modeling language to the core systems engineering challenge: how to tame complexity while aligning people, process and technology around a common product vision.

A model-based systems engineering approach allows organizations to design before they build – allowing stakeholders to visualize, simulate and fine-tune product specifications with a greater degree of precision far earlier in the product life-cycle -- ideally, before project resources have been committed and project schedules established. This enables organizations to explore multiple solutions or ideas concurrently with minimal risk; manage size and complexity with rich abstractions, and detect errors and omissions earlier in the product development lifecycle, when they are significantly less costly to find and fix.

A key advantage of a visual and standards-based modeling approach is that it aligns electrical, mechanical, software engineers and architects – all of whom use differing technology language and toolsets – with a common, whole-systems view of the product under development. It replaces ambiguity with clarity that transcends both engineering discipline and geographic location. It provides a common language that is easily understood by all engineers, whether their native language is English, Chinese or German, and whether they specialize in electrical, mechanical or software engineering.

Systems engineering can be practiced at varying levels of maturity. In early stages, organizations model, design and validate individual products. As organizations mature their systems engineering practices, they begin to design, specify and validate complete product lines. It is at this second stage, known as model-based product line engineering, that companies realize the greatest value from their MBSE investments. Table 3.1 presents a cost comparison between comparable systems developments based on MBSE and non-MBSE.

Worldwide	2015		2013		2010	
Using MBSE	✓	×	✓	×	✓	×
Development time months	13.2	12.7	8.5	13.4	12.9	11.7
% Behind schedule	31.5%	38.6%	38.7%	38.8%	45.6%	56.5%
Months behind	5.2	4.9	5.9	4.9	4.2	3.9
% Cancelled	8.9%	16.3%	11.1%	12.7%	11.4%	14.3%
Months lost to cancellation	4.1	4.3	6.0	5.4	5.4	4.3
SW developers/project	4.8	10.4	8.5	13.4	8.9	12.4
Average developer months/project	63.4	132.1	72.3	179.6	114.8	145.1
Dev. months lost to schedule	7.9	19.7	19.4	25.5	17.0	27.3
Dev. months lost to cancellation	1.8	7.3	5.7	9.2	5.5	7.6
Total developer months/project	73.0	159.0	97.3	214.2	137.3	180.0
At $10,000/developer month						
Average developer cost/project	$633,600	$1,320,800	$722,500	$1,795,600	$1,148,100	$1,450,800
Average cost to delay	$96,139	$269,599	$194,081	$254,761	$170,453	$273,234
Average cost to cancellation+	$17,515	$72,894	$56,610	$91,897	$54,788	$76,248
Total developer cost/project	**$747,254**	**$1,663,293**	**$973,191**	**$2,142,258**	**$1,373,341**	**$1,800,282**

Table 3.1: Comparing Cost of MBSE and non-MBSE Systems Developments (2010- 2015)

In order to present a comparable cost analysis we assume that the cost per developer man month is $10,000. That assumption is based on a fully loaded cost/developer. The reader can also look to "Total Developer Months/Project" as another comparative view.

The data presented in Table 3.1 clearly shows:

- Between 2010 and 2015, MBSE developments show a distinct advantage over similar developments that don't use MBSE.
- Between 2010 and 2015 the average cost per MBSE development has dropped 83.8%. This can be attributed by realizing that with experience come gains in productivity.
- Between 2013 and 2015 the average cost of non-MBSE developments dropped by one-third, but the comparison of the 2010 and the 2015 average cost of non-MBSE developments was 20%.
- The data supports the idea that while the average cost of all systems developments have improved over the period 2010-2015, MBSE developments have not only proved to be less costly (roughly less than half as costly) but have continued to cost less as experience with MBSE has increased.

4. What is Product Line Engineering?

The goal of PLE (product line engineering) is to reduce the time, cost and effort required to create, deploy and maintain similar products. To achieve this goal, the ultimate solution must minimize duplicate effort, maximize commonality among design and implementation assets, and optimize reuse of effort across similar products within each of its product lines.

Essentially, PLE offers a fundamental shift in perspective thereby creating a simpler solution to a traditionally difficult problem. It is simpler to take a broader perspective that views product line engineering as creating a single system instead of viewing product line engineering as creating a multitude of products. If a company can adopt the viewpoint of addressing all of their developments as a single system using the same system and software assets, rather than multiple individual developments, then the capabilities and advantages described herein (including effective code reuse) can be achieved with considerable savings.

PLE allows you to have a common design where you vary only the specific pieces that are different. PLE also allows you to have a strong mapping between the features you are designing and the common and variable pieces without having to make copies which must be separately maintained. A PLE lifecycle framework offers a consistent and integrated solution for all system and software tools and for the reuse of all systems and software assets, across the entire product development and maintenance lifecycle. This is much more effective than traditional development approaches.

It should be mentioned that the PLE concept is not new – but it has been difficult to effectively utilize in the past. However, by utilizing Model-Based PLE which incorporates the UML, SysML, variant modeling, simulation, and asset reuse capabilities of PTC Model-Based Systems Engineering, the new PLE concept is far easier to integrate into ongoing processes, thereby avoiding a disruption in the development process. It is also worth noting that this approach has also been recently standardized in the ISO 26550:2013 Software & Systems Engineering – Reference Model for Product Line Engineering & Management, showing its wide acceptance and the desire for standardized tooling.

Having stated benefits of PLE as a basic software solution, the next question is how do you start working in a PLE driven environment?

The fact is many companies practice code re-use, which is one component to a larger PLE process. The issue of code reuse depends on how you do it. Some approaches create limitations.

The primary ways to get reuse are:

- Clone and own – make a copy of what you are working on
- Inheritance
- Common services

These all work to some extent but each has limitations.

Clone and own has a major limitation in the fact that maintenance becomes a headache and time consuming. Inheritance is a great practice but it causes users to rewrite pieces that go into each specific child. Common services have an issue of ownership. Which product owns each service?

There are different but equally challenging issues for systems engineers with sub-systems physically used within systems, then improvements and replacements over time. When building new systems, how can you step up to 'product lines' an identify commonality, existing assets and valid sub-system variations.

By incorporating the advantages of PLE with Model Driven Development (MBSE) for software and Model-based Systems Engineering (MBSE), the introduction of previously developed and tested code or systems can be easily re-incorporated into the new product version irrespective of the operating system or the microprocessor selected, which may be different than those used in earlier product variations.

To focus on a software example, by being able to add new features to existing and previously tested code – even if a new OS or microprocessor has been selected – it is obvious that huge savings can be realized and development time (time-to-market – or windows of opportunity) and risks can be significantly reduced.

Table 4.1 presents a three year comparison between PLE and comparable developments. As part of the year over year EMF Survey of Embedded Developers, respondents were asked to identify which (if any) code reuse technologies of which PLE was one of he options.

These survey results encompass all product-line engineering approaches, both model-based and non-modeled based. They therefore represent a dataset that overlaps our previous inquiry on the relative value of an MBSE approach. The key take-away is that MBSE and product-line engineering are each significant, positive efficiency and cost reduction drivers.

Worldwide	**2015**		**2014**		**2013**	
Using PLE Developments	✓	×	✓	×	✓	×
Development time months	11.1	12.9	13.3	14.4	12.5	13.5
% Behind schedule	36.9%	37.9%	35.6%	39.7%	34.1%	39.1%
Months behind	3.3	4.2	4.7	4.6	3.3	4.4
% Cancelled	11.0%	11.1%	10.6%	10.8%	7.5%	10.1%
Months lost to cancellation	3.5	4.3	4.3	5.1	4.1	4.6
SW developers/project	5.8	8.1	6.1	7.7	7.4	9.4
HW developers/project	3.4	5.3	3.6	4.6	3.9	6.0
Total project developers	9.2	13.4	9.7	12.3	11.3	15.4
Average developer months/project	102.1	172.9	129.0	177.1	141.3	207.9
Dev. months lost to schedule	11.2	21.3	16.2	22.5	12.7	26.5
Dev. months lost to cancellation	3.5	6.4	4.4	6.8	3.5	7.2
Total developer months/project	116.9	200.6	149.7	206.4	157.4	241.5
At $10,000/developer month						
Average developer cost/project	$1,021,200	$1,728,600	$1,290,100	$1,771,200	$1,412,500	$2,079,000
Average cost to delay	$112,028	$213,301	$162,300	$224,623	$127,159	$264,942
Total developer cost/project	**$1,133,228**	**$1,941,901**	**$1,452,400**	**$1,995,823**	**$1,539,659**	**$2,343,942**

Table 4.1: Comparing Cost of PLE and Comparable non-PLE Developments

From Table 4.1 the following is observed:

- The margin of cost differential between PLE and comparable non-PLE developments has increased (it costs less to develop) between 2013 and 2015.
- The average cost of PLE developments has dropped in each of the three years, whereas the average cost of comparable non-PLE developments has remained basically the same.
- PLE has maintained a "Behind Schedule Completion advantage" and a "time-to-market" advantage as well.

It should be mentioned that the data presented in Tables 3.1 and 4.1 are based on different development environments and that the cost of hardware developers is included in the Table 4.1 calculations.

As part of the EMF Survey of Embedded Developers, respondents were asked to report on how close their final design came to their pre-design expectation. Developers were given the following choices:

- Within 10%
- Within 20%
- Within 30%
- Within 40%
- Within 50%
- Not within 50%

EMF considers designs completed within 20% of pre-design expectation to be "excellent design outcomes". Table 4.2 presents the percentage of excellent design outcomes compared with total designs undertaken. Design outcomes were determined for "Performance" and for "Systems Functionality".

Designs Completed within 20% of Pre-Design Expectations	2015	
Using PLE	✓	×
Performance	69.2%	61.1%
Systems Functionality	68.3%	61.2%

Table 4.2: Comparing PLE and non-PLE Design Outcomes

From Table 4.2, 69.2% of PLE designs are considered to be "excellent design outcomes". EMF considers design completed within 30% of pre-design expectations to be "satisfactory" outcomes. Here we wanted to show the percentage of excellent design outcomes. The difference between PLE and comparable developments to be significant.

5. Summary and Final Thoughts

Technology concepts and "market insights" in the embedded world are frequently based on "thought experiments" and "logic" which has led to remarkable surprises among early adopters and unfortunate vendors. Remember when Linux was "free"? Today it makes "sense" that if open source software is free then using it should significantly reduce the cost of development. Unfortunately the total cost of development includes not only the front end cost of acquisition, but also the number and cost of developers, the time to project completion, the frequency of behind schedule completions and cancellations as well as the cost of maintaining deployed systems.

The PLE concept has also relied on the assumption that its use "makes sense" and should cost less. I see frequently stated examples of one or more OEMs that report on their success. I can't remember the source of the following quote but it puts these assumptions in their proper perspective.

The Plural of ANECDOTE is NOT DATA

The data and results presented herein are based on thousands of developer responses over a period of 6 years. The 2015 data alone is based on 1334 developer responses. Although significant insights can be drawn from a single year's data, if statistical concepts are followed, the ability to look over many years of data to see trends and repeatable data are very significant. The analysis we presented herein was intended to be concise and a guide for the reader to examine the data for themselves.

EMF believes that the year over year data support the conclusions that were reached – both for MBSE and for PLE. What we didn't speak to he factors behind this data that would explain why the results were not really surprising. Here is a summary.

- MBSE modeling provides a clear understanding of the design intent and eases communication between team members and customers. In addition it makes team based design easier due to a graphical means of partitioning the design and defining interfaces.
- SysML and UML are a standard sets of graphical notations that enable Model-Driven Development (MDD) and MBSE. The graphical notations in SysML and UML for performing design architecture are based on OO techniques. When developing C applications the OO ability to define completely encapsulated elements enable developers to control complexity as the design scope increases.
- Some C developers prefer to follow a structured modeling paradigm instead of an OO method. By providing graphically the concepts of Blocks and Flows, which dominate the typical structured methodologies, C developers are quickly able to adopt a UML based MBSE environment without having to learn a new methodology. MBSE allows validation of the model at any point in development. This enables the C developer to find and eliminate errors early in the process when they are least expensive to fix.
- Constant validation of the system is key to producing high quality systems very quickly eliminating the need to wait until all the design and implementation is done and prior to testing the model. The closer the model execution resembles how the model will execute on the target system, the greater the value provided by MBSE.
- Production code generation gets one to final product quicker with fewer defects and lower risks. It frees up C developers to work at a higher level of abstraction in order to quickly create even very complex systems. A lot of systems today have become so complex that handwriting the entire system would require years as opposed to months using MBSE technology.

Appendix

About EMF

EMF is the premier market intelligence and advisory firm in the embedded technology industry. Embedded technology refers to the ubiquitous class of products which use some type of processor as a controller. These products include guided missiles, radars, and avionics as well as robots, automobiles, telecom gear, and medical electronics.

Embedded Market Forecasters (EMF) is the market research division of American Technology International, Inc. EMF clients range from startups to Global 100 companies worldwide. Founded by Dr. Jerry Krasner, a recognized authority on electronics markets, product development and channel distribution, EMF is headquartered in Ashland, Mass.

Website: www.embeddedforecast.com
Email: jerry@embeddedforecast.com
Phone: +1 508-881-1850

About the author

Jerry Krasner, Ph.D., MBA is Vice President of Embedded Market Forecasters and its parent company, American Technology International. A recognized authority with over 30 years of embedded industry experience, Dr. Krasner was formerly Chairman of Biomedical Engineering at Boston University, and Chairman of Electrical and Computer Engineering at Wentworth Institute of Technology and Bunker Hill Community College. In addition to his academic appointments, Dr. Krasner served as President of Biocybernetics, Inc. and CLINCO, Inc., Executive Vice President of Plasmedics, Inc. and Clinical Development Corporation, and Director of Medical Sciences for the Carnegie-Mellon Institute of Research. Earlier, he was Senior Engineer at the MIT Instrumentation Laboratory. Dr. Krasner earned BSEE and MSEE degrees from Washington University, a Ph.D. in Medical Physiology / Biophysics from Boston University and an MBA from Nichols College. He is a visiting professor at the Universidad de Las Palmas (Spain) where he is recognized for his work in neurosciences and computer technology.