APPLIED CAx

Integrated MBSE

Addressing organization-wide communication problems

Table of contents

1.0 A failure to communicate

When cross-product problems surface, we have a tendency to blame engineering. The warden on Cool Hand Luke[i] was right, we don't have an engineering problem, we have a communication problem. Product design decisions in one domain can affect many other domains. Failure to communicate those decisions to others results in late-discovered problems that consume half or more of a program's schedule to correct.

> **"What we have here is a failure to communicate..."**
> - Warden, Cool Hand Luke

WHY?

As product complexity increases, we bring more and more engineering talent on the job, adding to the complexity:

- Doing more with more constraints and less time
- Dealing with very demanding customers
- Interacting with more people

This means you are not only dealing with engineering problems—you have communication and information management problems. This is made worse by each discipline speaking a different language: mathematics, electronics, software, manufacturing, product support and more.

Systems engineering (aka cross-domain engineering) is supposed to be managing the cross-domain interactions between product domains, but it speaks yet another language delivered in Excel spreadsheets, Word documents, and PowerPoint/Visio diagrams, which don't scale to the now millions of lines of code, thousands of electronic control units, complex mechanical interactions, and hundreds of thousands of requirements.

Discipline-specific models don't scale as they are built for specific disciplines. We can't expect SysML (the first attempt at a standard system modelling language) to be learned by the other disciplines, thus the growing confusion at the product/discipline boundaries are where all the really bad product mistakes/failures/recalls happen.

So, what do we do about this communication problem?

First some background...

2.0 Why systems engineering?

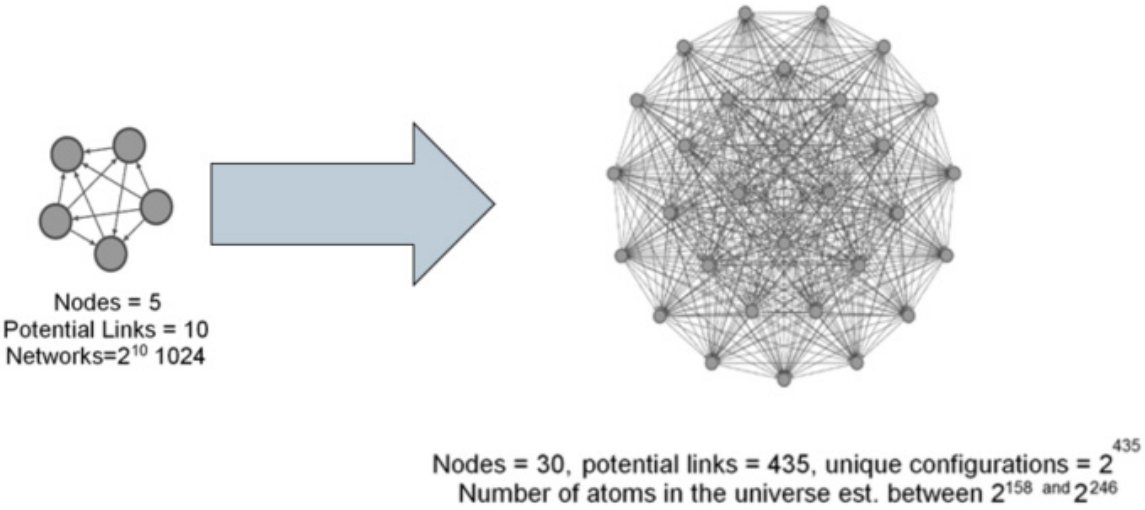
Murphy was an optimist.

MURPHY'S LAW

You're familiar with Murphy's Law. "Whatever can go wrong, will go wrong," along with some of its derivatives:

- Any line you stand in will be the slowest
- Toast always falls butter side down
- Anything you drop, will end up in the worst possible spot

Although initially approached as a lark, universities are beginning to take Murphy's Law seriously. For instance, Axton University in the U.K. has begun quantifying these laws and their outcomes, such as the long-standing question: Does toast always land butter-side down? In collaboration with The Daily Telegraph[ii], they recruited 1,000 students to conduct 20 trials each, dropping toast from the edge of a plate. The result? They found that 62.3% of the time, the toast falls butter-side down—this is not random, so you're not paranoid to believe that even toast is out to get you!



In today's products, which are increasingly complex combinations of mechanics, electronics, and software—including hundreds of ECUs, millions of lines of code, and hundreds of thousands of evolving requirements and regulations—the opportunities for Murphy's Law to strike are growing exponentially, reaching near certainty.

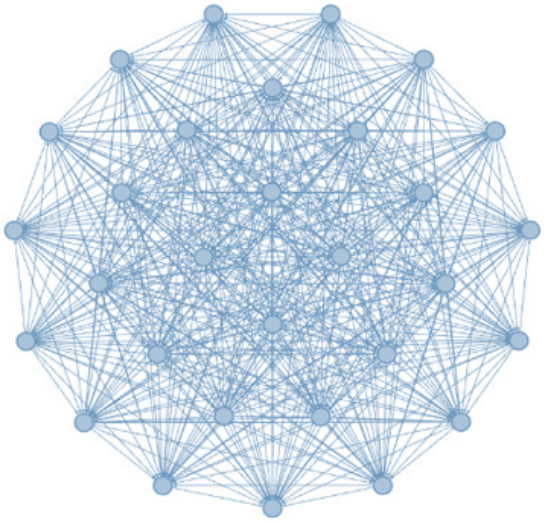
Doing the math, even 5 nodes/things interacting with each other with 10 possible links produces 2^{10} possible network combinations.

2.0 Why systems engineering?

As the number of nodes increases it rapidly grows beyond comprehension with more combinations possible than the estimated number of atoms in the known universe!

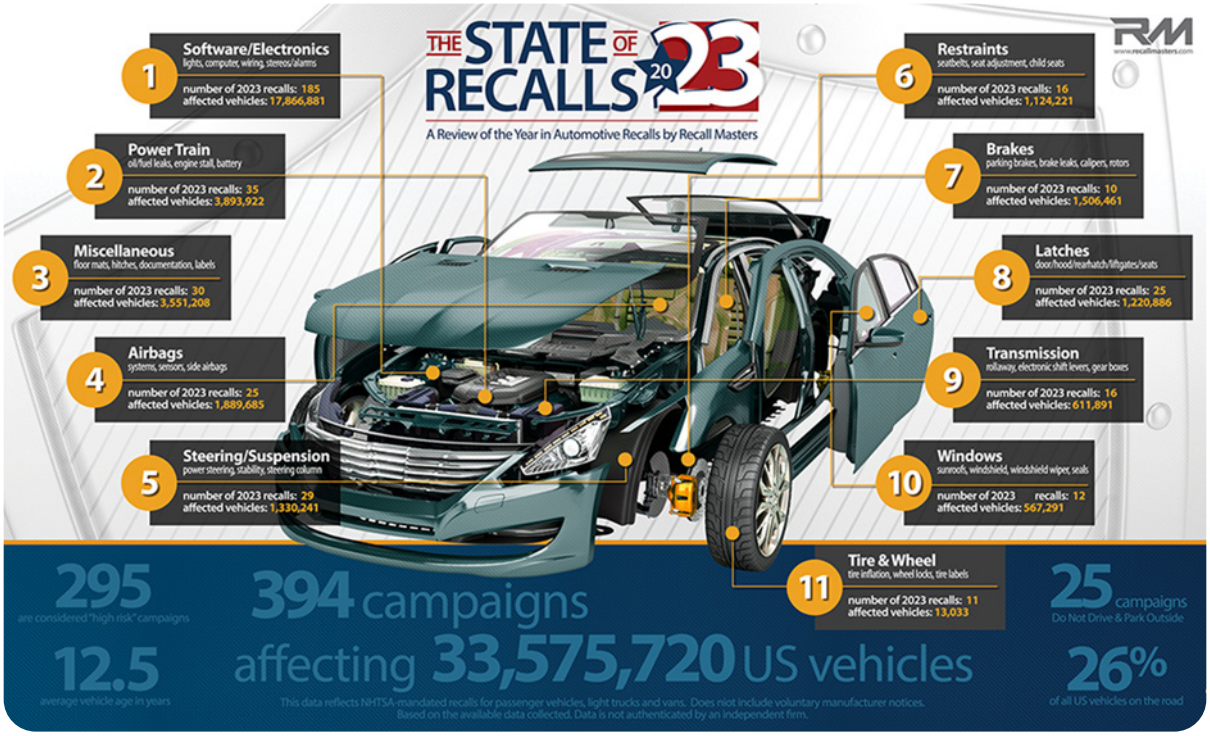
PREVENTING FAILURES

Catching cross-product issues before they cost millions



Since all the bad things in a product's life happen at these interfaces/combinations, we have a near certainty that catastrophic product problems will happen. The goal then is catching these cross-product/inter-domain problems early in the lifecycle before they escape and are discovered by customers and result in expensive product recalls—these are "failures to communicate" caused issues.

The U.S. automotive industry is a good example. The National Highway Traffic Administration (NHTSA.Gov) says there were some 33 million vehicle recalls in the U.S. in 2023 (see summary infographic by RecallMasters.com below).



2.0 A failure to communicate

THE TRUE COST OF RECALLS

Why cross-domain collaboration is essential

According to AlixPartners Automotive Consultancy[iii], each vehicle recall costs approximately $500 per car. In 2023 alone, this added up to $16.5 billion in direct costs to fix the problems—excluding indirect costs like excess inventory, overnight shipping, and damaged brand reputations.

Many of these issues occur at the boundaries or combinations between disciplines, where traditional discipline-specific tools fail to identify them. To prevent these costly failures, a cross-domain perspective and stronger communication are crucial.

DOMAIN DISCONNECTS

When communication breakdowns lead to costly recalls

To illustrate how issues arise at domain boundaries, consider this recall from a large truck manufacturer: An electronic control unit (ECU) with a zinc backplate was directly mounted onto a steel crossmember. This setup led to galvanic corrosion, which eventually destroyed the ECU. The failure caused fuel pumps to shut down, leading to a recall that affected 86,000 vehicles and cost the company approximately $43 million. Further analysis shows this wasn't an engineering problem at all but a purchasing problem. Purchasing added a new ECU supplier, not communicating the mounting location and material requirements. The supplier chose a less-expensive zinc back-plate which was then mounted directly on a steel cross-member in assembly with the resulting corrosion.

BLUEPRINTS FOR SUCCESS

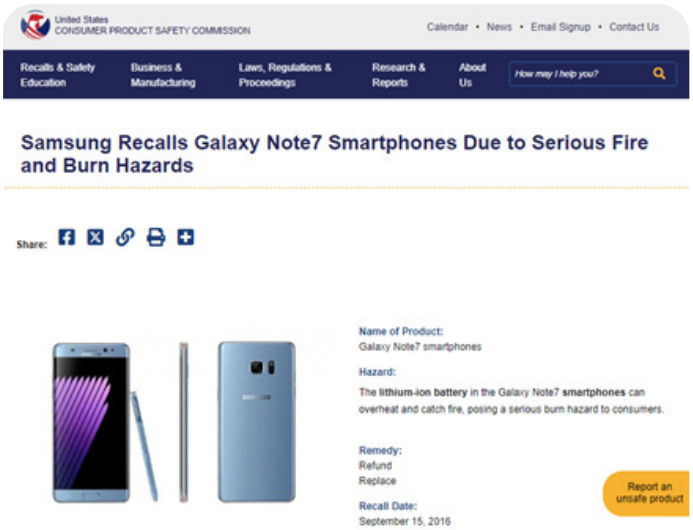
The critical role of product architecture

Detecting and preventing cross-domain issues requires a high-level, cross-domain product architecture established before product development begins. Like a building's architecture, a product's architecture serves as the blueprint, providing guidance to downstream development teams by clearly defining what needs to be done and how to do it. This architecture is the essential means of communication between disciplines.

No one would start a building or construction project without blueprints. However, many product development organizations today lack a comprehensive product architecture—or, more critically, an integrated one (to be discussed later). As product complexity grows, this absence leads to costly recalls and repeated mistakes with each domain doing their own thing and worrying about bringing it all together later.

2.0 A failure to communicate

Of course, it's not just vehicles that have these issues. These types of communication problems happen in nearly all complex, cross-domain development organizations—like these examples from recent headlines:


United States
CONSUMER PRODUCT SAFETY COMMISSION

Calendar • News • Email Signup • Contact Us

Recalls & Safety Education | Business & Manufacturing | Laws, Regulations & Proceedings | Research & Reports | About Us | How may I help you?

Samsung Recalls Galaxy Note7 Smartphones Due to Serious Fire and Burn Hazards

Share:

Name of Product:
Galaxy Note7 smartphones

Hazard:
The lithium-ion battery in the Galaxy Note7 smartphones can overheat and catch fire, posing a serious burn hazard to consumers.

Remedy:
Refund
Replace

Recall Date:
September 15, 2016

Report an unsafe product


MARKETS | BUSINESS | INVESTING | TECH | POLITICS | CNBC TV

Walmart sues Tesla over solar panel fires at seven stores

PUBLISHED TUE, AUG 20 2019 • 4:33 PM EDT | UPDATED WED, AUG 21 2019 • 12:36 PM EDT

Lora Kolodny
@LORAKOLODNY

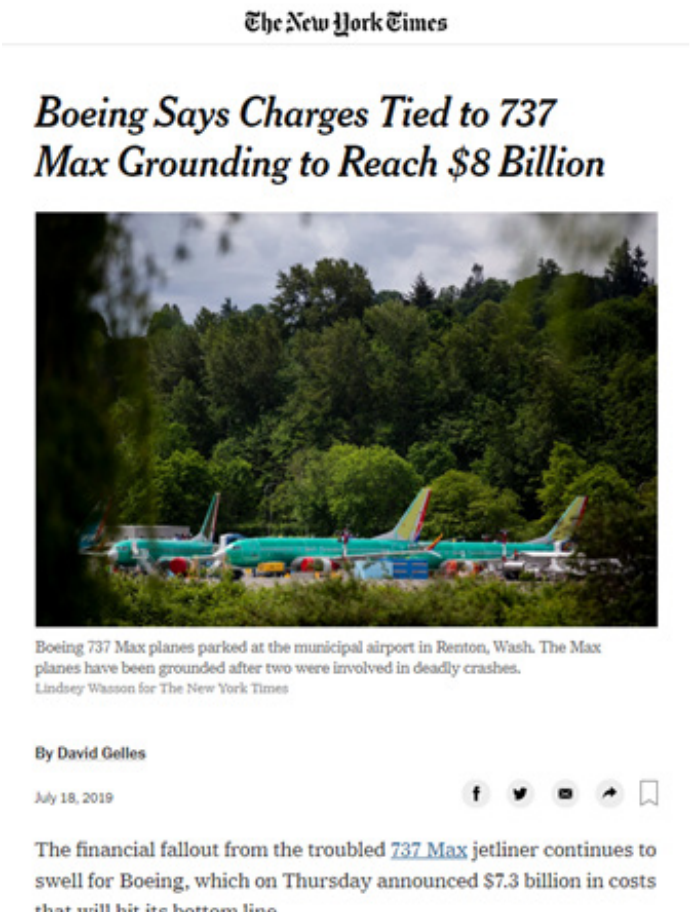
SHARE

KEY POINTS

- Walmart is suing Tesla for breach of contract after Tesla solar panels ignited atop seven of its stores.
- Tesla and Walmart have been partners on clean energy initiatives for years; more than 240 Walmart stores have Tesla solar systems installed.
- Walmart has also pre-ordered at least 45 Tesla electric semi-trucks to add to its vehicle fleet.

@WORK
PEOPLE + MACHINES
NOV 4 | SAN F
A future of work event fo
REGISTER

TRENDING NOW


The New York Times

Boeing Says Charges Tied to 737 Max Grounding to Reach $8 Billion

Boeing 737 Max planes parked at the municipal airport in Renton, Wash. The Max planes have been grounded after two were involved in deadly crashes.
Lindsey Wasson for The New York Times

By David Gelles

July 18, 2019

The financial fallout from the troubled 737 Max jetliner continues to swell for Boeing, which on Thursday announced $7.3 billion in costs

FUTURE-PROOFING DESIGN

Why integrated product architecture is essential

The value of a cross-domain integrated product architecture, created through systems engineering, lies in the problems and costs it avoids in the future rather than immediate savings. Boeing, Tesla, and Samsung serve as examples of companies that could have saved billions of dollars if they had identified cross-domain impacts and issues during product development instead of discovering them post-launch.

Integrated product architecture defines what needs to be done and how to do it. During the architectural design process, architects establish relationships—often referred to as the digital thread—that weave through the entire product lifecycle. These relationships identify potential future risks and issues, providing an opening for Murphy's Law to cause problems.

Our goal is to integrate this architected cross-domain communication digital thread with the product lifecycle, enabling us to foresee and prevent costly mistakes before they happen.

3.0 Systems engineering

THE ORIGINS OF SYSTEMS ENGINEERING

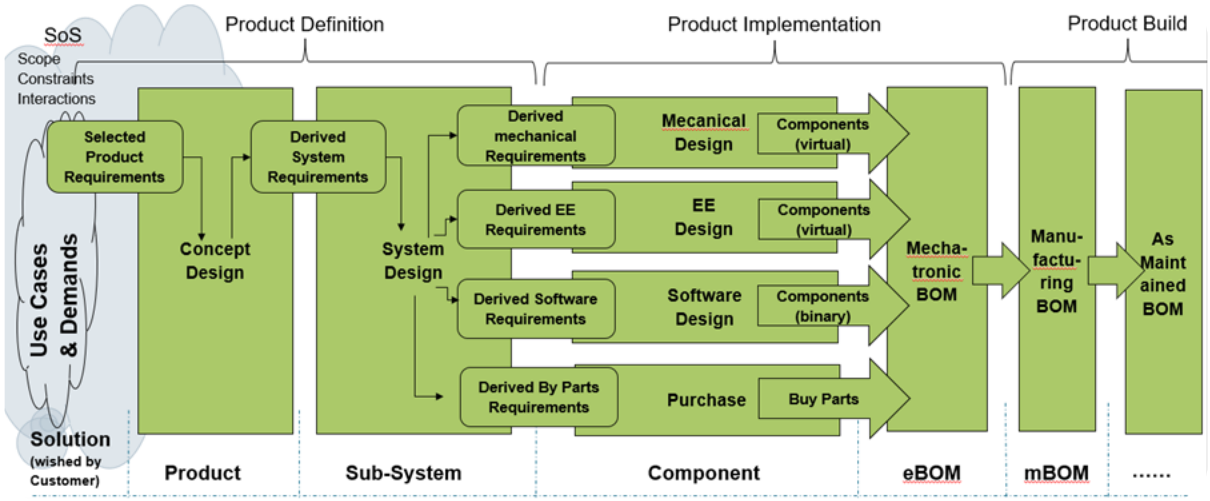
From pyramid builders to aerospace

Although the pyramid builders were systems engineers, Systems Engineering (SE) wasn't formalized until the 1950s with complex aerospace and defense projects (Polaris Missile, Early Warning Radar).

SYSTEMS ENGINEERING STANDARDS

Key principles in systems engineering

The principals and processes were developed into standards by various standards organizations: ISO15288, EIA 632, IEEE1220, and others. They describe principles such as defining the problem before you solve it, understanding operational concepts, managing system interactions, and then moving through a V-like process as described here:



THE SE PROCESS OVERVIEW

From concept to production

The SE process starts at the left side with early product development, focusing on how the product interacts with other systems, like an airplane's integration with air traffic control, navigation, and airports. This understanding defines the scope, constraints, and interactions, shaping the product's role in its operational environment.

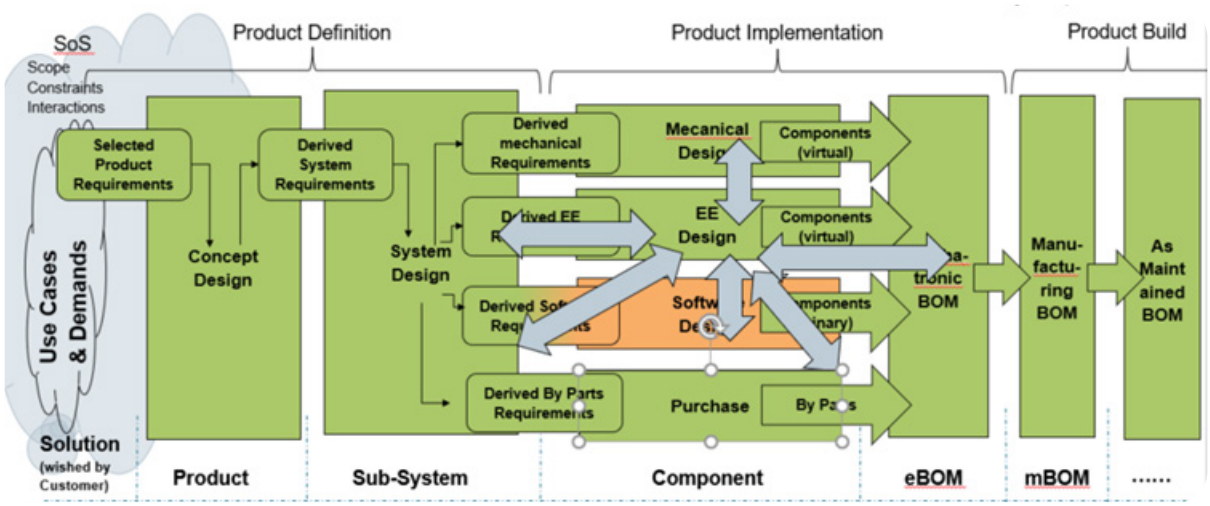
Moving to the right, we generate scenarios, use cases, and customer needs to create high-level requirements for conceptual design. We then explore alternative approaches, moving to sub-system design that is then managed, manufactured and task allocated across disciplines (electrical, software, mechanical). These decisions form a Bill of Materials (BOM) that is managed, manufactured, maintained, and eventually retired.

3.0 Systems engineering

CHALLENGES IN LOW MATURITY PROCESSES

When development starts backwards

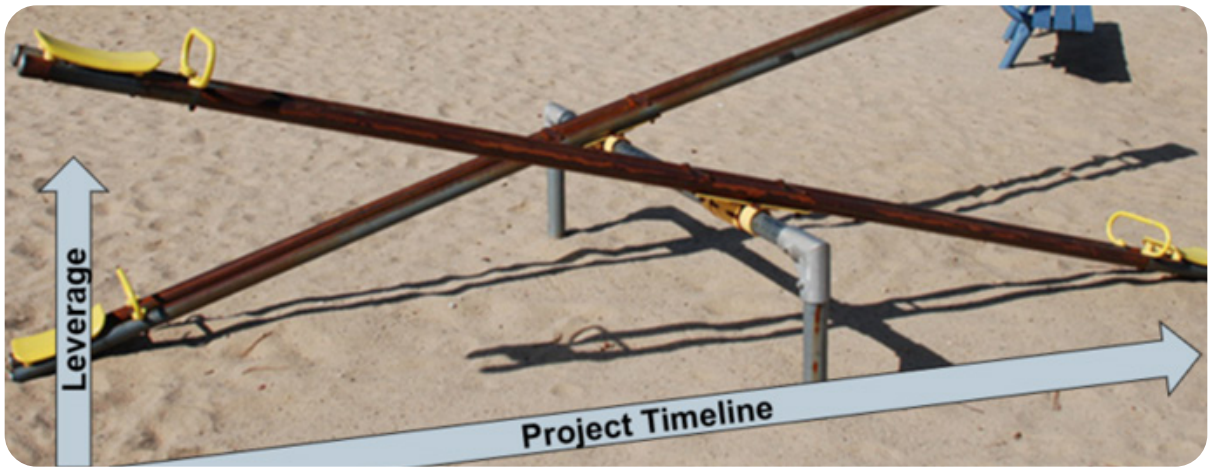
Organizations with low development process maturity often start on the right side and work backward through the process, addressing cross-domain issues as they arise—usually after problems surface and costs escalate. As they move left into development disciplines, these teams optimize within their silos, creating more issues. The dominant domain often prevails, leading to disintegrated, unoptimized solutions that fail to meet overall requirements. As IEEE states, “We are good at component design, 90% work as designed. However, 50% fail when integrated into the system they were designed for.”



THE COST OF DELAYED SE

The critical timing of systems engineering

This points out one of the interesting things about this SE process (and systems engineering generally): if the early product definition process isn’t done—or done fast enough (for whatever reason)—downstream product development continues with or without the blueprints, resulting in the late discovery of problems. Thus, the value of systems engineering decreases the later it is applied in the project, as the value of the architecture direction is missing from design decisions becomes a costly burden when late-design problem correction edicts come down (or even require starting over on the design). This is why Simon Ramo (the “R” in TRW) put it this way: “All the really big mistakes are made the very first day.”[iv] You already know this instinctively, as you know where the most leverage is on a playground teeter-totter/seesaw.

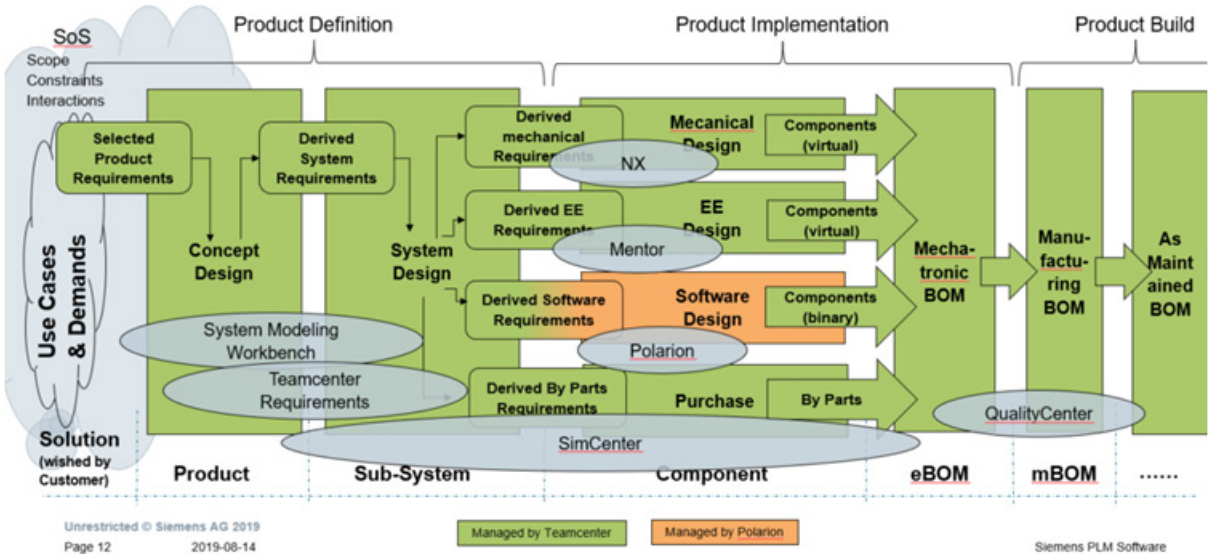


3.0 Systems engineering

Systems Engineering is about defining the problem before we start solving it.

Laying a project timeline under the seesaw, you know instinctively where the big leverage is. It's not in the middle of the project where we are working on mechanical, electrical or other design. So, the product/project leverage is early in the lifecycle where the most important WHAT to do and HOW to do it decisions are made. So, it makes sense to get the architecture right from day one and manage it with the rest of the product information.

Teamcenter is the keeper of product knowledge, including product architecture knowledge (remember the architecture is what established the cross-domain relationships that enable balanced, recall-free development).



ACHIEVING INTEGRATED MBSE

Connecting systems engineering to the product lifecycle

Our integrated MBSE goal is to integrate systems engineering with the product lifecycle starting at the beginning of the project where the most important decisions are made and carrying those decisions throughout the product lifecycle. Those what/how decisions establish the relationships that carry the digital thread across the lifecycle into downstream development tools for guided/connected product development communication creating recall-free products.

4.0 The value proposition

> **“What we have here is a failure to communicate...”**
> - Warden, Cool Hand Luke

4.1 A failure to communicate...

Back to our communication problem: systems engineering (and its model-based implementation—MBSE) is about creating and communicating a product architecture that considers all aspects of a product (requirements, cost, materials, manufacturing, competition, reliability and safety) and driving the downstream development process to deliver against that vision. It serves as the scaffolding that enables cross-domain communication. We cannot create surprise-free products unless everyone is on the same page, working toward a common mission, and understanding their role in the overall development process. This ensures they know how their part of the product fits with other domains (i.e., interfaces).

Since it is impossible to predict or account for every scenario during development—such as late supplier deliveries, unexpected changes in customer requirements, or unavailable resources—systems engineers play a vital role in adapting to these surprises. They manage alternatives, adjust plans, and arbitrate how to handle the multiple cross-domain impacts of the unexpected. This is akin to a building architect who designs the original plans but also spends time at the construction site to address surprises. If the architect isn’t present—virtually or otherwise—the trades make decisions in isolation without understanding the broader consequences, leading to serious and costly setbacks when these issues surface.

Due to their generalized background and experience, systems architects or engineers are far fewer in number than discipline-specific engineers (e.g., electrical, mechanical, software). However, their leverage (remember the playground seesaw analogy) makes them among the most critical contributors to a project. These architects need tools to help them capture, accelerate, and integrate the product architecture, staying ahead of the “construction crews.” Want proof? Forensic accounting at Texas Instruments in the 1990s predicted a 70% probability of a “runaway” project—out of control, behind schedule, and over budget—unless the architects remained ahead of the development teams.[v]

Later, as the project transitions to design engineering, manufacturing, planning, purchasing, support, and other lifecycle stages, the architecture becomes democratized. This means that everyone involved in the project must adopt a systems perspective for their aspect of product support. For example, a purchasing agent’s mistake (as seen in the ECU recall case above) highlights the importance of understanding cross-domain impacts. Democratized architecture models create opportunities for many more team members to consume and apply systems engineering and product architecture information effectively.

4.2 So, how bad is your communication problem?

As described earlier, different domains speak different “languages,” making direct communication between disciplines challenging. This is where systems engineering, and the architecture it creates, plays a critical role. It establishes the scaffolding needed for effective cross-discipline communication. These decisions, made at the cross-domain level, ensure products are optimized across all areas.

4.0 The value proposition

4.2 So, how bad is our communication problem?

However, if this scaffolding is fragmented—spread across disconnected documents, isolated tools, or siloed processes—disciplines cannot communicate effectively. This disconnection creates inter-discipline and inter-model friction, which ultimately leads to communication breakdowns and product issues.

ASSESSING COMMUNICATION EFFECTIVENESS

MBSE maturity matrix: Measuring friction across disciplines

To understand and measure this communication friction, we can use an MBSE Maturity Assessment Matrix. This framework is grounded in systems development processes outlined in standards like ISO15288, EIA632, and IEEE1220, as detailed in the INCOSE SE Handbook.

The matrix summarizes the things that systems engineers are supposed to do (per the standards) and asks what technologies are being used to perform those jobs.

Things Systems Engineers do...

CMMI-type Mapping:	(1) Initial	(2) Managed	(3) Defined	(4) Qualitative	(5) Optimizing
How they do them...	**Uncontrolled**	**Controlled Documents**	**Isolated models**	**Enterprise Integration**	**Continous Engineering**
System Architecture Modeling *Product architecture definition*	...in PPT or docs	...in Visio diagrams (managed in GIT, etc.)	...in standard SysML modeling tools	...integrated system architecture inside PLM	Continuous integration via in-PLM architecture
Planned Product Variability *PLE/Configuration/Variation*	None	...in variation documents & spreadsheets	...thru disconnected variation rules	...integrated PLM variation rules	PLM variation drive architecture decisions
Reliability & System Safety Analysis *Technical Risk (RAMS)*	...in risk documents & spreadsheets	...in managedl RAMS artifacts (FMEA,...)	...in disconnected RAMS tools (FMECA ...)	...PLM-integrated RAMS analysis tied into arch	Continous RAMS risk assessment & alarms
Cross domain services					
System Definition & Design Integration *Logical modeling & Interface mgmt*	..in ICD & logical description documents	...in PLM Managed interfaces & logical structures	...in interface libraries tied to SE artifacts	...interfaces using PLM managed para's, reqs,...	PLM arch across domains for continuous I/F mgmt
Integrated services					
Feature Engineering *Feature/Functional Modeling*	...in Feature/Functional description docs	...in a PLM-based functional hierarchy	...are func models tied to controlled functions?	...PLM func modeling using std para's, reqs, ...	PLM func arch allocations for func visability
Parameter/Target Mgmt *Characteristic/Targets/TPM*	...in many uncontrolled spreadsheets & docs	...in controlled spreadsheets & docs	...in project Parameter/Target libraries	...in Enterprise PLM parameter/target mgmt	PLM para's, targets for continuous compliance
Change management	...in document-based change process (CCB's, etc.)	...in PDM change practice including some models	...in PLM with change impact & suspicion mgmt	...inside PLM with models, para's, history,...	PLM change history is next starting point
Content Management					
Requirements Analysis *Requirements engineering & mgmt*	...in uncontrolled spreadsheets & docs	...in managed requirements docs (sharepoint,...)	...in standalone RM tools with exchange	...integrated with other domain models (CAD,...)	PLM-enabled continouous compliance
Behavior Model Management System, performance, et al simulation	...in uncontrolled models on desktops	...in GIT version controlled model respository	...are SE artifacts linked into models?	...in PLM model & product config with simulation	Continuous simulation & multi-domain optimizing
Verification Management & Governance *Product Test/V&V*	...in uncontrolled documented test procedures	...in managed test cases	...are SE artifacts linked into test?	...in Devops-like V&V HIL/SIL simulation	Continuous focused testing
Physcial Design Management *CAD, CAE,.. control/mgmt*	...in unmanaged storage (desktops, etc)	...in your PDM system	...are SE artifacts linked into CAD?	...are SE artifacts linked across domains	Continuously verfied physical Digital Twin
Problems/Value realized at each MBSE Maturity level:	**Disconnected Silos: Model chaos**	**Disconnected models operating in silos**	**Some models exchanging info with some insights**	**Fine-grained cross-domain situational awareness**	**Virtual development environment matches reality**

Disintegrated development---------- Continuously Integrated development

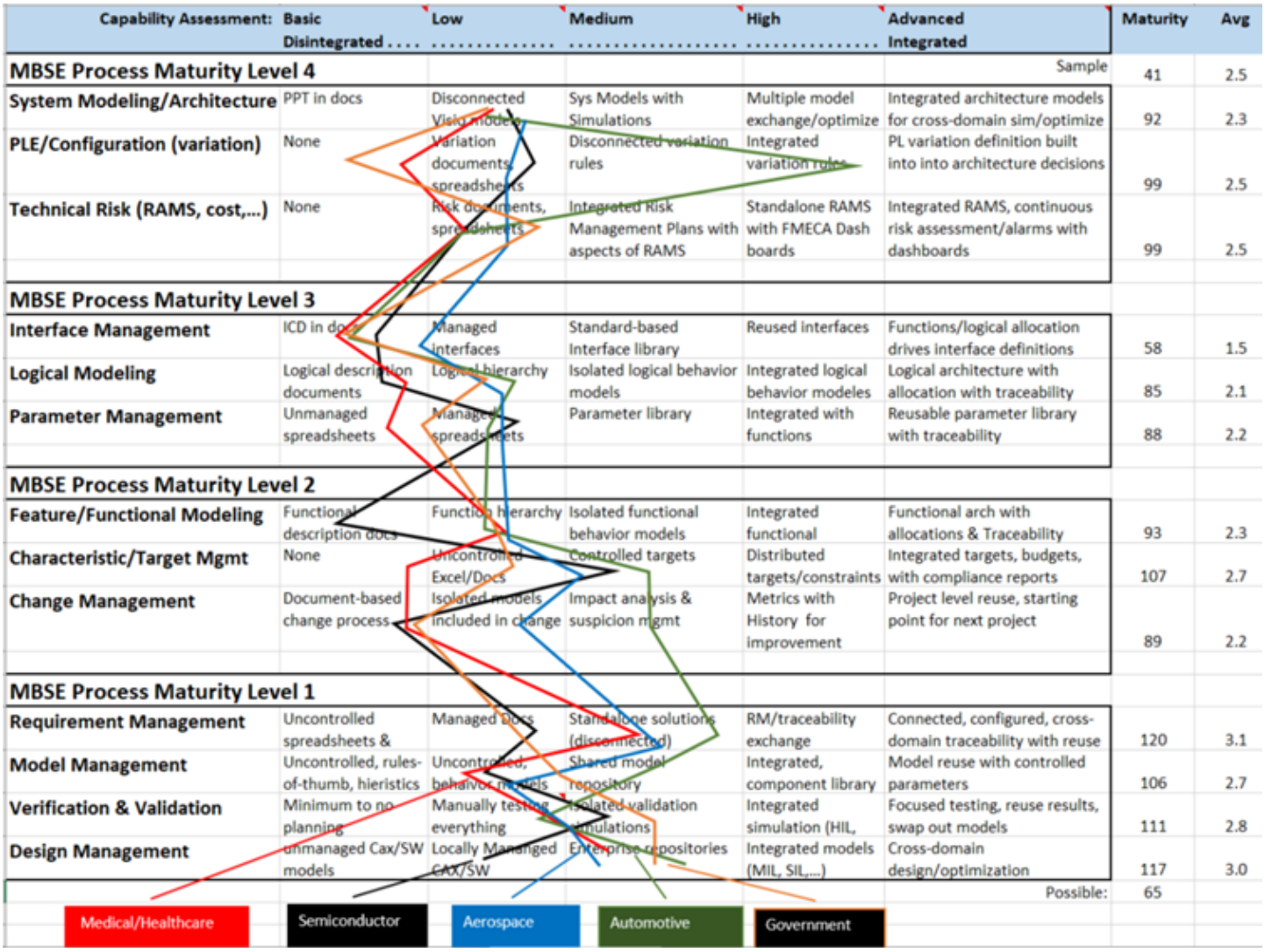
4.0 The value proposition

4.2 So, how bad is our communication problem?

This assessment matrix asks what technologies you are using to do the jobs that systems engineering are supposed to do:

- Where do you manage requirements?
- Where do you manage test cases?
- Where do you manage change? ... and so forth.

We have done many of these surveys in many different industries and found that everyone is MBSE communication challenged.



Capability Assessment:	Basic Disintegrated	Low	Medium	High	Advanced Integrated	Maturity	Avg
MBSE Process Maturity Level 4					Sample	41	2.5
System Modeling/Architecture	PPT in docs	Disconnected Visio models	Sys Models with Simulations	Multiple model exchange/optimize	Integrated architecture models for cross-domain sim/optimize	92	2.3
PLE/Configuration (variation)	None	Variation documents, spreadsheets	Disconnected variation rules	Integrated variation rules	PL variation definition built into into architecture decisions	99	2.5
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS	Standalone RAMS with FMECA Dash boards	Integrated RAMS, continuous risk assessment/alarms with dashboards	99	2.5
MBSE Process Maturity Level 3							
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions	58	1.5
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability	85	2.1
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability	88	2.2
MBSE Process Maturity Level 2							
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional	Functional arch with allocations & Traceability	93	2.3
Characteristic/Target Mgmt	None	Uncontrolled Excel/Docs	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports	107	2.7
Change Management	Document-based change process	Isolated models included in change	Impact analysis & suspicion mgmt	Metrics with History for improvement	Project level reuse, starting point for next project	89	2.2
MBSE Process Maturity Level 1							
Requirement Management	Uncontrolled spreadsheets &	Managed docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse	120	3.1
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behavior models	Shared model repository	Integrated, component library	Model reuse with controlled parameters	106	2.7
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL,	Focused testing, reuse results, swap out models	111	2.8
Design Management	unmanaged Cax/SW models	Locally Managed CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization	117	3.0
					Possible:	65	

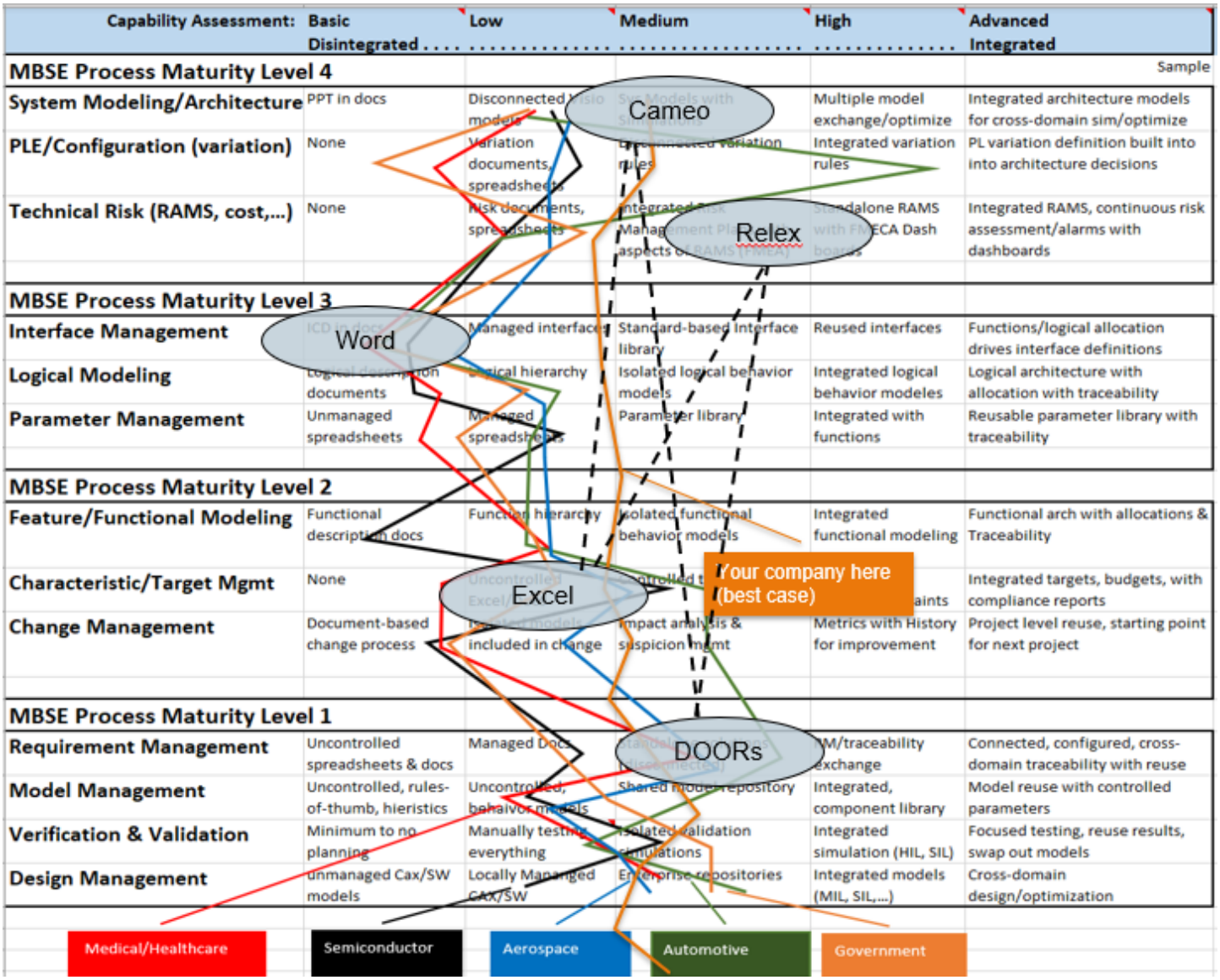
Medical/Healthcare | Semiconductor | Aerospace | Automotive | Government

We should point out that this is an “optimistic” model. We ask respondents to think about the best experience they can remember in their organization. So, these lines represent the best projects they’ve seen. Imagine what the problem programs look like.

4.0 The value proposition

4.2 So, how bad is our communication problem?

Adding tools for each row (although there are a few tools that claim to cover everything):

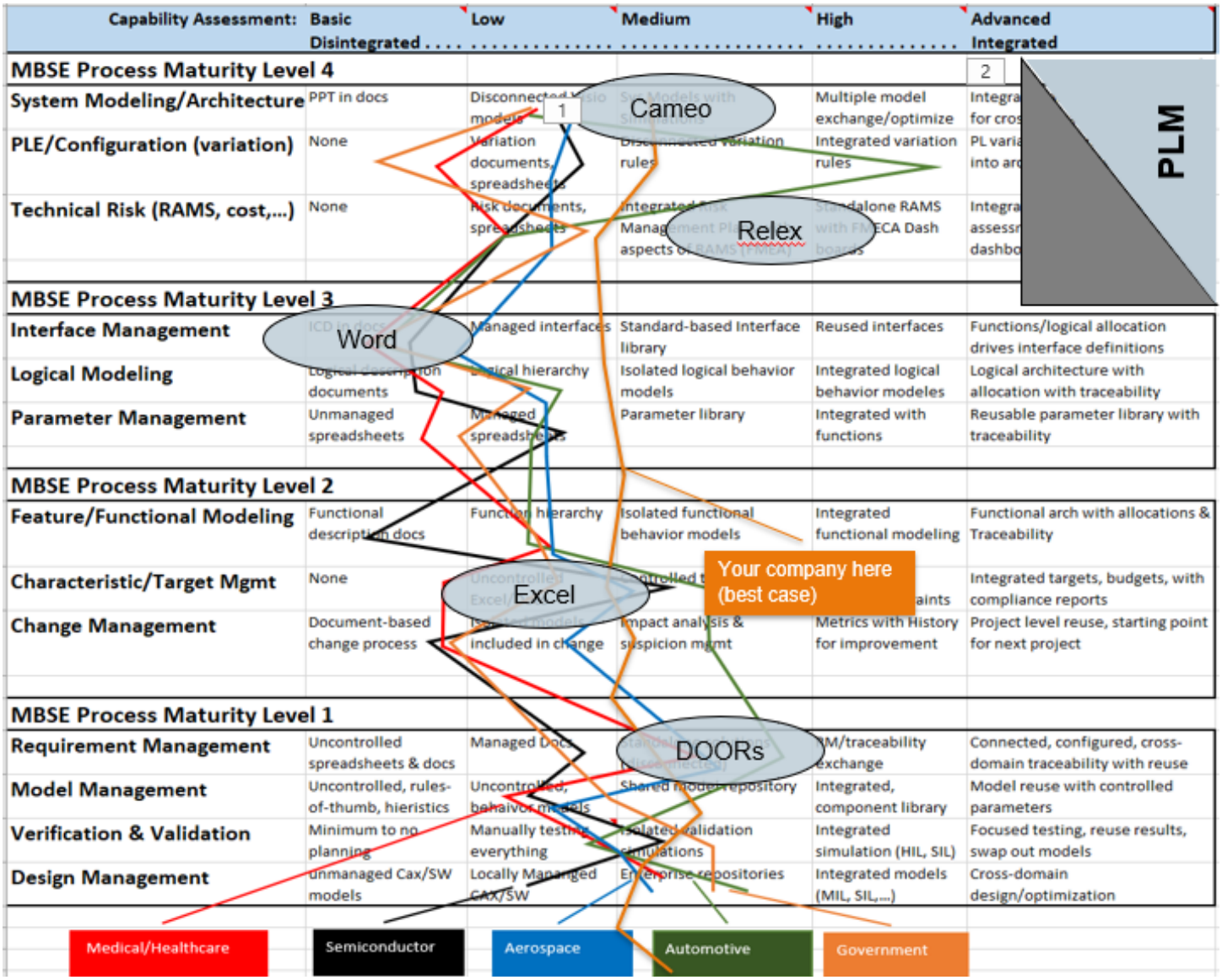


You can now see the communication problem between the various tools used to support a particular part of the systems development process. Drawing lines of communication/exchange of information between tools, such as DOORs to/from Cameo, doesn't solve the communication problem since neither tool understands product configuration, change, history, variation, etc. You can also see how these tools lock in an organization at a particular level of communication maturity rather than allowing an organization to continue its journey to more advanced communication. Remember, our complex 2^{10} math problem (NxN) means you can't keep up with the number of exchanges between tools. You must rely on the PLM System, which maintains the configured product structure with all its nuances, changes, versions, etc.

4.0 The value proposition

4.2 So, how bad is our communication problem?

These tools provide their domain specific perspective but it's your PLM system that provides the baseline configuration/source of truth that each of them needs to rely on to give trustworthy answers. The combination of the individual tools with current product configuration yields cross-domain understanding from our individual silo perspective–what's referred to as SysLM–elevating Product Lifecycle Management (PLM) to System Lifecycle Management (SysLM).



This process view changes our perspective from thinking about individual tools to how to enable a continuous communication model-based approach that leverages integrated systems engineering's ability to create surprise-free products.

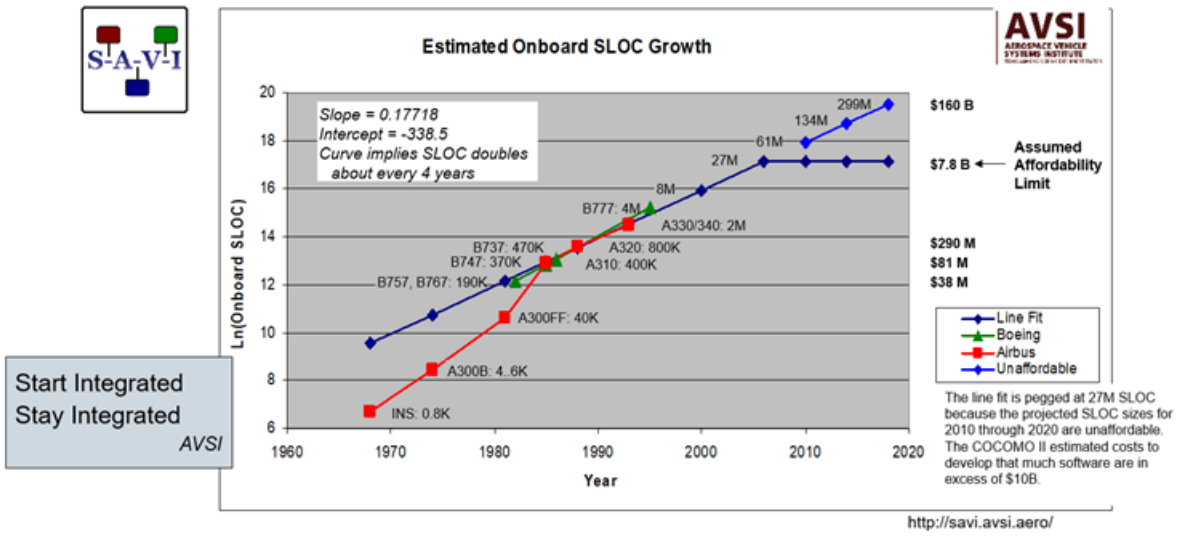
4.0 The Value Proposition

4.3 Extending MBSE into the supply/design chain

Once we understand the integrated MBSE process, we see our customers and suppliers as part of a broader system of systems (SoS). This product architecture enables communication across the entire design chain, creating a Model-Based Design Chain (MBDC) that connects customers, our process, and suppliers in a seamless development journey.

This means if we are only thinking about our organization, we're not getting all the potential value; i.e., the value extends into the suppliers that are contributing to the product development process (as much as 60%-70% of a product value in some industries). The product architecture defines WHAT will be done and HOW it will be done not only to inside development but also to the outside/purchased parts as well. Today, once it's decided what parts to purchase, an RFP/spec is sent to the supplier requesting quotes, etc. That spec describes the WHAT and HOW to the supplier so they can deliver something that integrates with the rest of the product to deliver the customer-required functionality. Today, this process is typically document-based, with periodic reviews (PDR, CDR) to check progress and make sure designs are aligned. Once delivered, "Systems Integration" starts to bring together all the component sub-systems into one system, hopefully cooperating to deliver the required functionality or sending it back for redesign/updates when integration problems are discovered.

Per the Aerospace Vehicle Systems Institute (AVSI, a consortium of the major air-framers—see http://savi.avsi.aero/), this cycle/approach is no longer affordable. The system integration problem (which consumes almost half of the product development cycle–often taking longer than the original design) makes building airplanes from scratch no longer affordable!

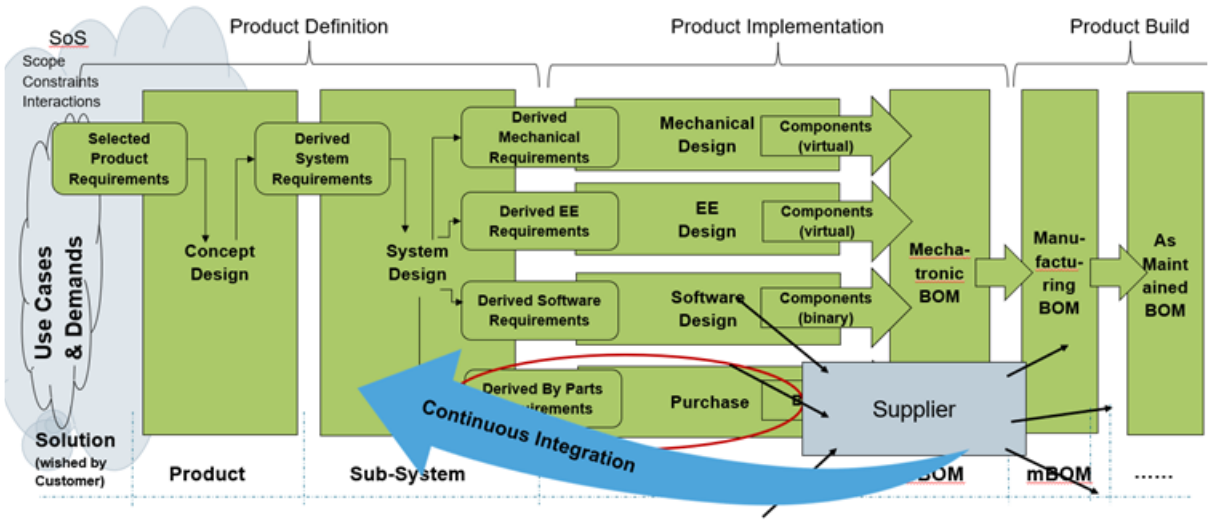


Of course, it's not just aerospace that has this problem. Almost all multi-domain industries pad their schedules for system integration risk/failure, and no one is ever surprised by a half-program schedule padding (for example; in the OEM-semiconductor business, it's not uncommon for OEM's to wait for prototypes to show up before starting design, delaying their Time to Market (TTM) and delaying semiconductor suppliers Time to Revenue (TTR)).

With a communication-enabling integrated product architecture in hand, we can solve this problem by passing the portion of the system (complete with functions, inputs/outputs, requirements, etc., with IP protection) to our suppliers, giving them a known good integrated model to work from in developing their sub-system. With the help of SysLM, periodically, they 'check-in' models so OEM level integration can start earlier and continuously, essentially eliminating the long, multiple, expensive serial system integration cycles and potentially saving as much as half of the product development cycle because the product is continuously integrated—what we mean by start integrated, stay integrated.

4.0 The value proposition

4.3 Extending MBSE into the supply/design chain



Most of the Murphy-type problems related above are integration failures–undiscovered integration failures. If we could look at your development problem histories, we are sure that uncommunicated changes were made either on the OEM or Supplier side that were not communicated/managed. This results in system integration problems (discovered late and very expensive), which makes the value of the integrated product architecture even greater when integrated with Teamcenter to keep track of who has what version and ensure everyone (including suppliers) is on the same page when a change is processed.

> **"There is no greater waste than doing efficiently something that shouldn't be done at all."**
> -Peter Drucker

4.4 Problem in a nutshell

Avoiding product development surprises is what keeps you up at night. We help you implement an integrated MBSE solution that enables continuous communication across domains through an integrated SoS product architecture that prevents product development surprises by ensuring we are always integrated across the entire design chain (from customer to OEM to suppliers and back)—thus the tagline: Start Integrated, Stay Integrated.

4.0 The Value Proposition

4.4 Teamcenter advantage

1. By integrating SE/MBSE/Product Architecture with Teamcenter PLM, we create a SysLM (System Lifecycle Management) solution that is much more than just managing product data. An integrated product architecture enables continuous communications across organizations to drive product development from a systems perspective that ensures the WHAT and HOW are implemented by the downstream development disciplines (including suppliers) in an integrated way. This ensures built-in compliance to requirements and architecture alignment, enabling organizations to “Start Integrated, Stay Integrated” (versus start design and integrate/fix later). This turns out to be a big deal. Without cross-domain communication, we end up discovering problems during system integration. You already know this, and in fact, plan for it by scheduling/planning half or more of your program schedules for system integration. Imagine the value if you had confidence in your shared architecture to eliminate that program schedule padding.
2. Integrating MBSE with the product lifecycle also integrates the product architecture with standard product lifecycle services like change, variation, workflow, and more; i.e. configuring the product also configures the product architecture, requirements, targets, test cases, and interfaces across the entire Design Chain. Imagine the value of knowing where a change goes now vs. later.
3. Finally, a scalable Teamcenter infrastructure enables global complex product development by ensuring everyone participating in product development, including suppliers, is on the same page/continuous communication. This ensures everyone, no matter what time zone, is guided by the product architecture and working with current information, which creates integrated-by-design products.

5.0 Summary

The problem in almost all complex product development is keeping everyone on the same page, with many headline-grabbing problems discovered late in the product development process (or by customers) caused by uncommunicated design assumptions, decisions, and changes. The cost of late discovery and design redo can be daunting, resulting in 'workaround' discussions that leave monuments behind (i.e., the tourist attraction Leaning Tower of Pisa is a failed architecture). There is some value in these monuments for future generations, like this 'Column of Shame' at a major engineering college that serves as an object lesson/warning to others when an architecture mistake (or no architecture at all) is discovered too late to correct–thus the integrated Systems Engineering/MBSE mantra of "Start Integrated, Stay Integrated."

To summarize:

To create the cross-product digital thread to guide your development and enable continuous communication you begin with product architecture that starts at the top (System of Systems).

- Like building architecture, System Modeling defines WHAT the product will do and HOW to do it
- Product architecture must be integrated with the product lifecycle to drive the development processes (including configuration, change, workflow, variance)
- A streamlined/simplified standard system modeling language with entwined standard methodology integrated with Teamcenter guides/ captures product architecture (SysLM)
- Unlike disconnected MBSE tools, the integrated architecture establishes the cross-domain dependencies needed to understand/manage complex cross-domain products
- So, you can discover/communicate issues now vs later, allowing organizations and their supply/ design chains to start integrated, stay integrated

Column of Shame

i Movie "Cool Hand Luke" starring Paul Newman released 1967

ii https://www.telegraph.co.uk/news/uknews/1331810/Breakfast-at-Murphys-or-why-the-toast-lands-butter-side-down.html

iii https://www.alixpartners.com/media/14438/ap_auto_industry_recall_problem_jan_2018.pdf

iv INCOSE 1997 International Symposium keynote address

v TI MBSE Landmark Study in authors possession