SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Start integrated, stay integrated

Addressing your communication problems in engineering

Executive summary

Avoiding unexpected problems during the product development process is the overriding challenge in complex product development. Issues are most often caused by uncommunicated design assumptions, decisions and changes. The cost of late discovery and design adjustments can be cost-prohibitive. Model-based systems engineering (MBSE) focuses on creating and exploiting domain models as the primary means of information exchange between engineers, rather than on document-based information exchange. Integrated MBSE – bringing systems engineering (SE) inside of product lifecycle management (PLM) – enables your entire engineering team to start integrated and stay integrated, heading off any potential problems before they can surface.

Contents

Abstract

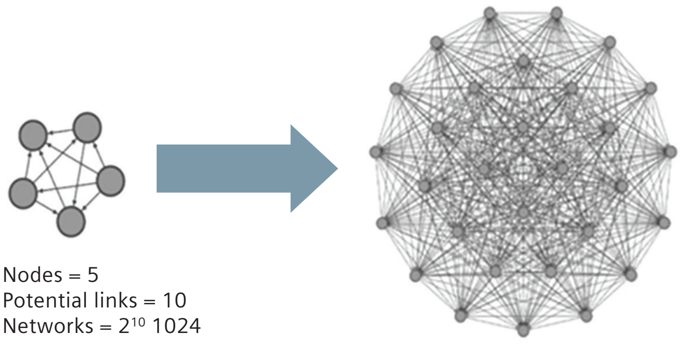
"What we have here is a failure to communicate."

Warden from the movie Cool Hand Luke

A failure to communicate is at the root of many engineering false starts and failures and can have catastrophic consequences.

As product complexity increases it requires more engineers dealing with mind-boggling complexity that are required to:

- Do more with greater constraints and less time
- Deal with very demanding customers
- Interact with more and more people



Nodes = 5
Potential links = 10
Networks = 2^{10} 1024

Nodes = 30, potential links = 435, unique configurations = 2^{435}
Number of atoms in the universe est. between 2^{158} and 2^{246}

The result is you no longer have an engineering problem, you have a communication and information management problem. This is exacerbated by each discipline speaking a different language: mathematics, electronics, software, construction, etc.

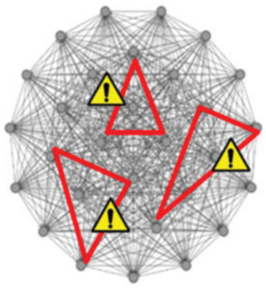
In this paper, we'll examine how systems engineering is supposed to be managing the cross-domain interactions between product domains, but it speaks yet a another language delivered in Microsoft® Excel®, Word and PowerPoint®/Visio diagrams which don't scale to the now millions of lines of code, thousands 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 and can't communicate across domains. We can't expect the Systems Modeling Language (SysML), the first attempt at a standard system modeling language, to be learned by all the disciplines, thus the growing confusion at the product/discipline boundaries where all the really bad product miscommunication/mistakes/failures/recalls happen.

So how do you solve this communication problem?

With today's products, an increasing combination of mechanics, electronics and software that include complex mechanical systems, hundreds of electronic control units (ECUs) and millions of lines of code exponentially increases the potential for error.

Since all the bad things in a product's life happen at these interfaces/combinations, it is a near certainty that problems will occur. The goal 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; let's call these "failure to communicate" issues.



The U.S. automotive industry is a good example as the National Highway Traffic Safety Administration (NHTSA) says there were 47 million vehicle recalls in 2018 affecting 17 percent of U.S. vehicles (figure 4). The NHTSA says each recall costs $100 per vehicle, resulting in $4.7 billion in direct-to-manufacturer costs, not including indirect costs such as extra inventory, overnight shipping parts, tarnished reputations, etc.

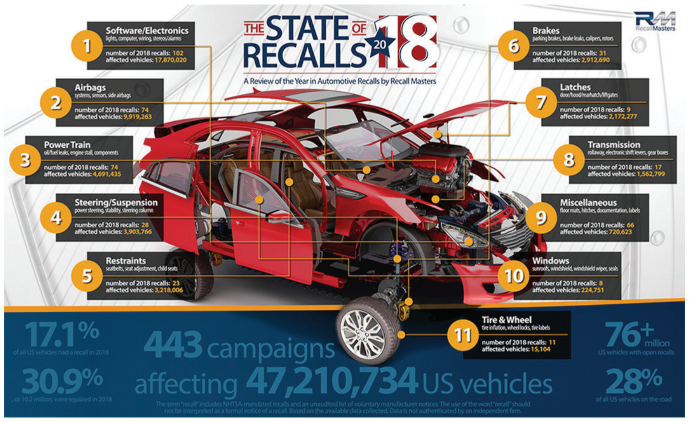


Figure 4: US Automotive Recalls summary by RecallMasters.com.

Here is an example of a combination-type recall/communication problem from a large truck manufacturer:

The picture below shows an ECU with a zinc back plate directly mounted on a steel crossmember. This resulted in galvanic corrosion that eventually destroyed the ECU, causing the fuel pump to shut down. The resulting recall affected 86,000 vehicles and cost about $8.6 million (figure 5).

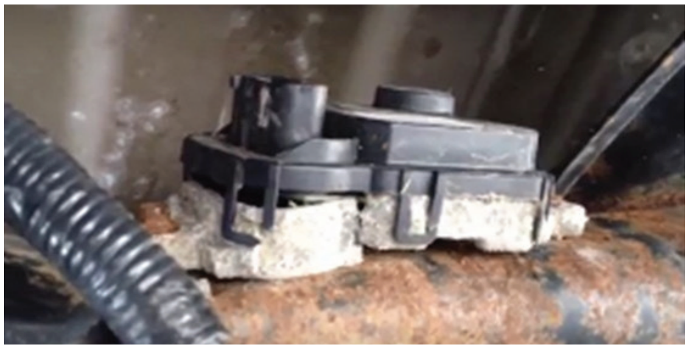
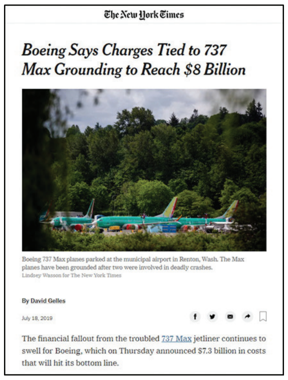


Figure 5: Recall of bad fuel pump control module.

The ability to see these types of cross-domain interactions requires a higher altitude cross-domain product architecture captured before starting product development to detect and prevent these problems. Like a building's architecture, the product architecture is the blueprint for the various downstream development disciplines describing WHAT needs to be done and HOW WELL to do it (the means of communicating). No one can imagine starting a construction project without a set of blueprints, but many product development organizations today don't have a product architecture (or integrated product architecture) and thus are experiencing these types of costly product recalls as their product complexity increases (often repeating the same mistakes).

Further investigation into our recall shows this wasn't an engineering mistake but rather a purchasing error. Purchasing decided to change suppliers in an attempt to reduce costs and neglected to communicate material and other requirements to the supplier, thus a new, less expensive zinc back-plated ECU went smoothly into the manufacturing chain without knowing it created a galvanic corrosion problem until fuel pumps stopped working months later, meaning that nearly everyone in an organization that directly or indirectly affect the product need to be thinking cross-domain systems (even purchasing).

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.



The New York Times

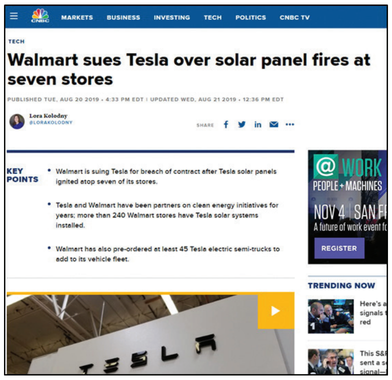
Boeing Says Charges Tied to 737 Max Grounding to Reach $8 Billion

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 that will hit its bottom line.



TECH

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

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.

The goal is to integrate product architecture where architects establish WHAT needs to be done and HOW WELL to do it with the product lifecycle. This is where a set of relationships (often called the digital thread) that run throughout the product are created. Those relationships form issues that could surface in the future. To avoid these problems, our goal is to integrate that architected cross-domain communication digital thread with the product lifecycle so we can be alerted to potential problems and avoid costly mistakes.

The systems engineering process

Although the pyramid builders were systems engineers, SE wasn't formalized until the 1950s with complex aerospace and defense projects such as the Polaris Missile and early warning radar. The principals and processes were developed into standards by various standards organizations: International Organization for Standardization (ISO)15288, Electronics Industry Association (EIA) 632, Institute of Electrical and Electronics Engineers (IEEE)1220 and others.

They describe principals such as defining the problem before you solve it, understanding operational concepts, managing system interactions and then move through a V-like process as described in figure 8.

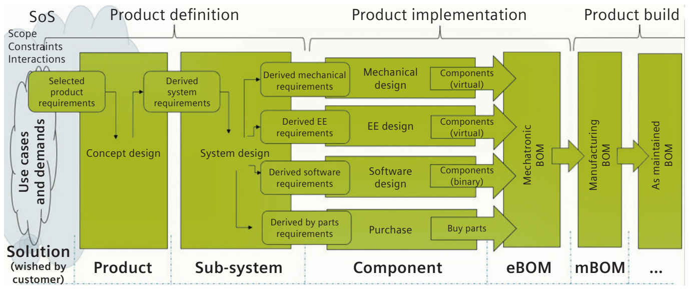


Figure 8: Standard ISO15288-type left-to-right development process.

The SE process starts at the far left with the early product development stages and works to the right. At the very beginning of a product development process we start by understanding how our product will work in the context of other products and systems. For example, an airplane interacts with many other systems, referred to as a system of systems (SoS) such as air traffic control systems, navigation systems, airports, maintenance, logistics, environment, and more. Understanding these establishes the scope, constraints, interactions, etc., for the product describing how the product will interact within its operational environment. This process generates a series of scenarios, use cases, demands and customer wishes that creates the initial set of high-level requirements that are the starting point for conceptual design.

Concept design is where we start looking for alternative ways of accomplishing the requirements, which in turn leads to subsystem design, eventually allocating WHAT to do to various downstream development disciplines (electrical, software, mechanical) and into the development specific design tools for electronic computer-aided design (ECAD) and mechanical computer-aided design (MCAD). As these disciplines make design decisions it turns into a bill-of-materials (BOM) that needs to be manufactured, then maintained, and eventually retired.

Organizations with low development process maturity typically start at the right side and work their way backward through the process as cross-domain issues are discovered (for example, starting where the problems surface and money is hemorraging). As they work into the development disciplines, the disciplines optimize around their development silos, creating more issues with the strongest domain winning at the cost of others, ending up with disintegrated, unoptimized, unbalanced solutions that don't meet the overall requirements. IEEE put it this way: "We are good at component design with some 90 percent of components working as designed. However, 50 percent of them fail when you plug them into the system they were designed for."

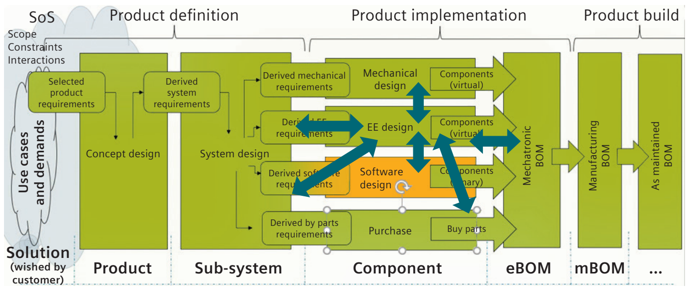


Figure 9: Silo optimized design on process.

This points out one of the interesting things about the SE process: if the early product definition process isn't 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 SE decreases the later it is applied in the project as the value of the architecture's direction becomes a costly burden as the late-design problem correction edicts come down (or even start over on the design). This is why American engineer Simon Ramo (that's the R in TRW) said, "All the really big mistakes are made the very first day." You already know this instinctively as you know where the most leverage on a playground teeter-totter/see-saw is (figure 10).

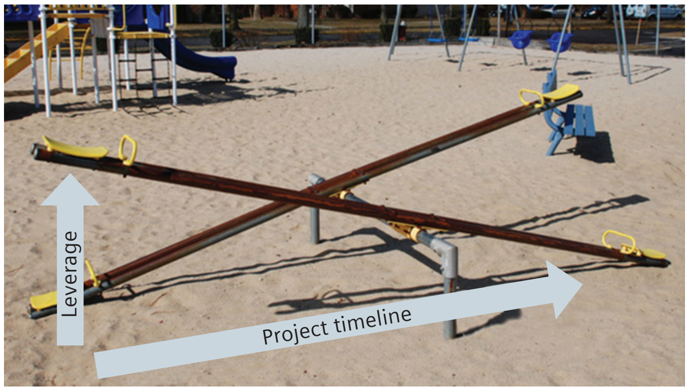


Figure 10: What a see-saw can teach us about project leverage.

Laying a project timeline under the see-saw, you know instinctively where the leverage is. It's not in the middle of the project where we are working on mechanical, electrical or other designs. The project leverage is early in the lifecycle where the most important WHAT to do and HOW WELL 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® software 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). Teamcenter is part of Xcelerator, a comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software.

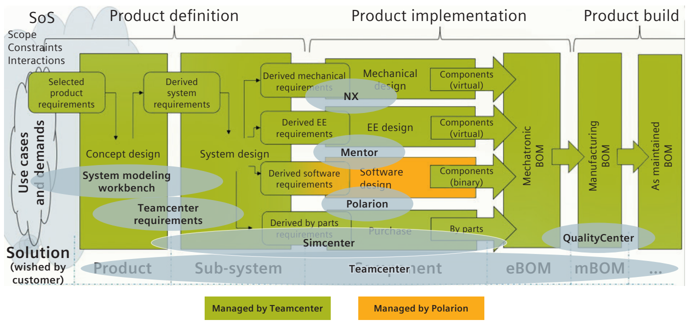


Figure 11: Where Siemens solutions fit into the process.

Our integrated MBSE goal is to integrate SE with the product lifecycle starting at the beginning of the project where the most important decisions are made and carry 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.

Systems Engineering is about defining the problem before solving it.

The value proposition

SE and its model-based implementation (MBSE) is about creating/communicating a product architecture that considers all aspects of a product (requirements, cost, materials, manufacturing, competition, reliability, safety, ...) and then drives the downstream development process to deliver against that vision; it's the scaffolding that makes cross-domain communication possible. We can't create surprise-free products unless everyone is on the same page, working towards a common mission, and understanding their part in the overall development process so they comprehend how their part of the product fits with other domains (for example, interfaces).

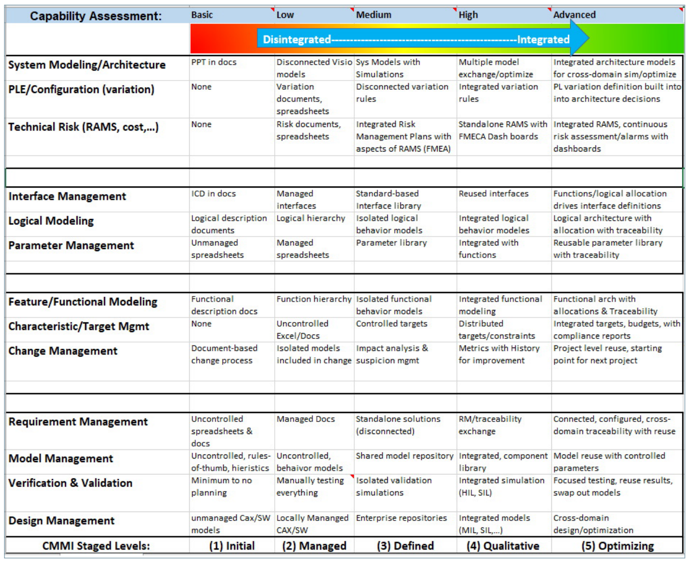
Since it's impossible to predict/consider everything that might happen during development (late delivery, unexpected change in customer requirements, unavailable resource, etc.), systems engineers will switch gears and perform the important role of executing the daily development process to work alternatives, adjust plans, etc., as development surprises occur to arbitrate how to handle the multiple cross-domain impacts of the unexpected. This is much like a building architect who creates the original set of plans for the construction crews but then shifts to spending time at the construction site to deal with surprises. If the architect isn't there virtually or otherwise, the trades make their decisions in isolation without understanding the consequences, potentially resulting in costly setbacks.

Because of their background and experience, there are fewer of these types of architects/engineers/leaders than discipline specific engineers, but because of their leverage they are among the project's most critical people. These architects need tools to help them capture the product architecture (accelerate capture and integrate product architecture) and keep them ahead of the construction crews. When we shift to design engineering, manufacturing, planning, purchasing, support, etc., the rest of the architecture's lifecycle becomes democratized, meaning that everyone involved in a project needs a systems perspective in their aspect of product support.

So, how bad is our communication problem?

As described earlier, different domains speak different languages, making it difficult to communicate directly between disciplines. This means that SE and the architecture it creates establishes the scaffolding needed for cross-discipline communications. If that scaffolding is disconnected (in documents, isolated tools, etc.) the disciplines can't communicate effectively leading to additional inter-discipline/inter-model friction that ultimately leads to communication failure.

You can understand/measure your communication friction, by using a MBSE Maturity Assessment Matrix based on standard SE development process and detailed in the International Council on Systems Engineering (INCOSE) handbook (figure 12).



Capability Assessment:	Basic	Low	Medium	High	Advanced
	Disintegrated---Integrated				
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
PLE/Configuration (variation)	None	Variation documents, spreadsheets	Disconnected variation rules	Integrated variation rules	PL variation definition built into into architecture decisions
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS (FMEA)	Standalone RAMS with FMECA Dash boards	Integrated RAMS, continuous risk assessment/alarms with dashboards
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional modeling	Functional arch with allocations & Traceability
Characteristic/Target Mgmt	None	Uncontrolled Excel/Docs	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports
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
Requirement Management	Uncontrolled spreadsheets & docs	Managed Docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behaivor models	Shared model repository	Integrated, component library	Model reuse with controlled parameters
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL, SIL)	Focused testing, reuse results, swap out models
Design Management	unmanaged Cax/SW models	Locally Mananged CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization
CMMI Staged Levels:	**(1) Initial**	**(2) Managed**	**(3) Defined**	**(4) Qualitative**	**(5) Optimizing**

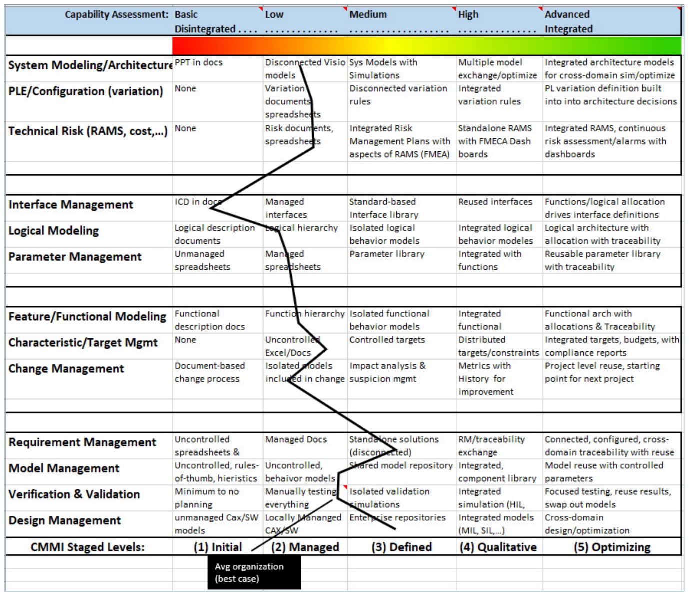
Figure 12: MBSE Maturity Assessment Matrix based on standard SE development process.

The Capability Assessment column (column A) documents what SEs do (per the process); the columns from left to right describe various ways of doing those SE tasks in disconnected up to integrated model-based ways starting with disintegrated on the left to integrated approaches on the right, creating a fast way to assess where an organization is in their integrated, continuously communicating MBSE journey. For example, looking at the requirement management row from left to right, organizations can manage requirements using disconnected, uncontrolled spreadsheets and documents, maintain requirements in managed documents, and in standalone requirement solutions such as DOORs or Jama. Organizations can also move to individual requirements that appear in other places (such as a requirement showing up in a MATLAB or CAD model), with the ultimate individual requirements integrated, managed, and or configured with the rest of the product data, related through to other domains creating cross-domain impact visibility/traceability.

Describing another one, interface management, remember, we care about interfaces in our products because that's where all the bad/unexpected things happen in a product that cause product failures. Moving from left-to-right on the interface management row, we can manage interfaces in interface control documents (ICDs). Or we can move up in maturity to managing individual interfaces, interface libraries, grouped in ways that support re-use (such as all the interfaces of a transmission), or the best integrated functional modeling that defines interfaces when you allocate a function to things performing those functions. This allows us to understand where functions go no matter where they are performed. For example, a wire in a harness can tell you what functional interface it is carrying.

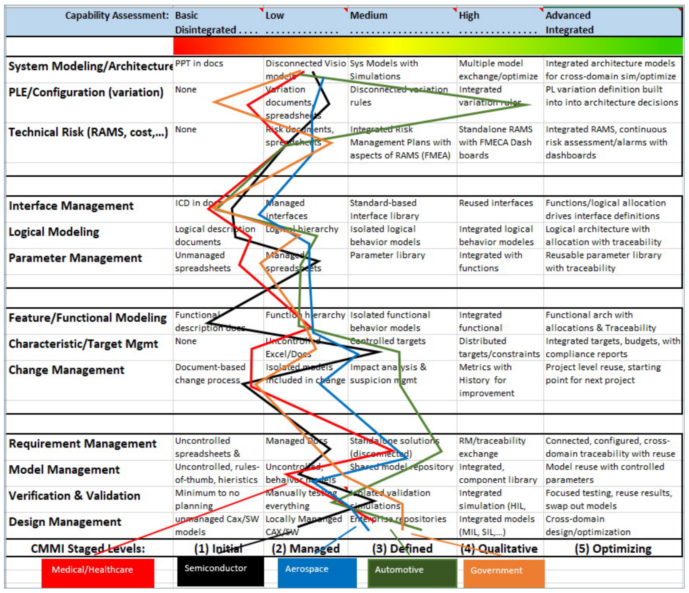
With those examples, we can quickly assess where we are and use that to drive an organizational MBSE process/journey towards model-based continuous communications.

We've been gathering samples from a variety of industries and have developed a measure of dysfunctional communication for an average organization (figure 13) or broken out by industry (figure 14).



Capability Assessment:	Basic Disintegrated	Low	Medium	High	Advanced Integrated
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
PLE/Configuration (variation)	None	Variation documents spreadsheets	Disconnected variation rules	Integrated variation rules	PL variation definition built into into architecture decisions
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS (FMEA)	Standalone RAMS with FMECA Dash boards	Integrated RAMS, continuous risk assessment/alarms with dashboards
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional	Functional arch with allocations & Traceability
Characteristic/Target Mgmt	None	Uncontrolled Excel/Docs	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports
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
Requirement Management	Uncontrolled spreadsheets &	Managed Docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behaivor models	Shared model repository	Integrated, component library	Model reuse with controlled parameters
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL,	Focused testing, reuse results, swap out models
Design Management	unmanaged Cax/SW models	Locally Mananged CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization
CMMI Staged Levels:	(1) Initial	(2) Managed	(3) Defined	(4) Qualitative	(5) Optimizing

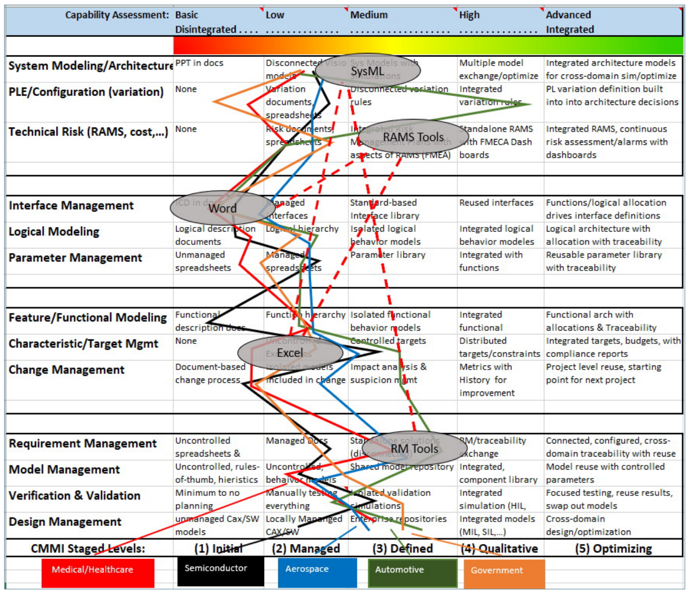
Figure 13: Average MBSE Assessment score for sampled organizations.



Capability Assessment:	Basic Disintegrated	Low	Medium	High	Advanced Integrated
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
PLE/Configuration (variation)	None	Variation documents spreadsheets	Disconnected variation rules	Integrated variation rules	PL variation definition built into into architecture decisions
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS (FMEA)	Standalone RAMS with FMECA Dash boards	Integrated RAMS, continuous risk assessment/alarms with dashboards
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional	Functional arch with allocations & Traceability
Characteristic/Target Mgmt	None	Uncontrolled Excel/Docs	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports
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
Requirement Management	Uncontrolled spreadsheets &	Managed Docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behaivor models	Shared model repository	Integrated, component library	Model reuse with controlled parameters
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL,	Focused testing, reuse results, swap out models
Design Management	unmanaged Cax/SW models	Locally Mananged CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization
CMMI Staged Levels:	(1) Initial	(2) Managed	(3) Defined	(4) Qualitative	(5) Optimizing

Figure 14: MBSE maturity by industry matrix.

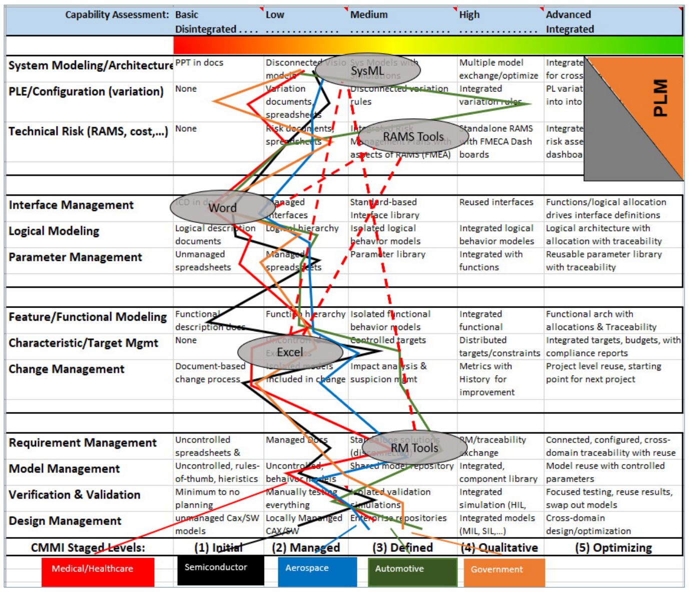
We should point out that this is an optimistic model. We ask respondents to think about the best experience they can remember in their organization, meaning, these lines represent the best projects they've seen. Imagine what the problem programs look like (figure 15).



Capability Assessment:	Basic Disintegrated	Low	Medium	High	Advanced Integrated
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
PLE/Configuration (variation)	None	Variation documents, spreadsheets	Disconnected variation rules	Integrated variation rules	PL variation definition built into into architecture decisions
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS (FMEA)	Standalone RAMS with FMECA Dash boards	Integrated RAMS, continuous risk assessment/alarms with dashboards
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional	Functional arch with allocations & Traceability
Characteristic/Target Mgmt	None	Uncontrolled Excel files	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports
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
Requirement Management	Uncontrolled spreadsheets &	Managed Docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behavior models	Shared model repository	Integrated, component library	Model reuse with controlled parameters
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL,	Focused testing, reuse results, swap out models
Design Management	unmanaged Cax/SW models	Locally Mananged CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization
CMMI Staged Levels:	(1) Initial	(2) Managed	(3) Defined	(4) Qualitative	(5) Optimizing

Figure 15: MBSE maturity by industry with tools.

The communication problem between the various tools used to support a particular part of the systems development process is clearly visible. Drawing lines of communication to exchange information between tools such as Requirements Management and SysML modeling tools, doesn't solve the communication problem since neither tool understands the big picture, 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. This leads to the conclusion of how important the systems development process is on a PLM system to enable models to communicate in context as part of a scalable, single source of product truth (figure 16).



Capability Assessment:	Basic Disintegrated	Low	Medium	High	Advanced Integrated
System Modeling/Architecture	PPT in docs	Disconnected Visio models	Sys Models with Simulations	Multiple model exchange/optimize	Integrate for cross
PLE/Configuration (variation)	None	Variation documents, spreadsheets	Disconnected variation rules	Integrated variation rules	PL variat into into
Technical Risk (RAMS, cost,...)	None	Risk documents, spreadsheets	Integrated Risk Management Plans with aspects of RAMS (FMEA)	Standalone RAMS with FMECA Dash boards	Integrate risk asse dashboa
Interface Management	ICD in docs	Managed interfaces	Standard-based Interface library	Reused interfaces	Functions/logical allocation drives interface definitions
Logical Modeling	Logical description documents	Logical hierarchy	Isolated logical behavior models	Integrated logical behavior modeles	Logical architecture with allocation with traceability
Parameter Management	Unmanaged spreadsheets	Managed spreadsheets	Parameter library	Integrated with functions	Reusable parameter library with traceability
Feature/Functional Modeling	Functional description docs	Function hierarchy	Isolated functional behavior models	Integrated functional	Functional arch with allocations & Traceability
Characteristic/Target Mgmt	None	Uncontrolled Excel files	Controlled targets	Distributed targets/constraints	Integrated targets, budgets, with compliance reports
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
Requirement Management	Uncontrolled spreadsheets &	Managed Docs	Standalone solutions (disconnected)	RM/traceability exchange	Connected, configured, cross-domain traceability with reuse
Model Management	Uncontrolled, rules-of-thumb, hieristics	Uncontrolled, behavior models	Shared model repository	Integrated, component library	Model reuse with controlled parameters
Verification & Validation	Minimum to no planning	Manually testing everything	Isolated validation simulations	Integrated simulation (HIL,	Focused testing, reuse results, swap out models
Design Management	unmanaged Cax/SW models	Locally Mananged CAX/SW	Enterprise repositories	Integrated models (MIL, SIL,...)	Cross-domain design/optimization
CMMI Staged Levels:	(1) Initial	(2) Managed	(3) Defined	(4) Qualitative	(5) Optimizing

Figure 16: MBSE maturity built on PLM.

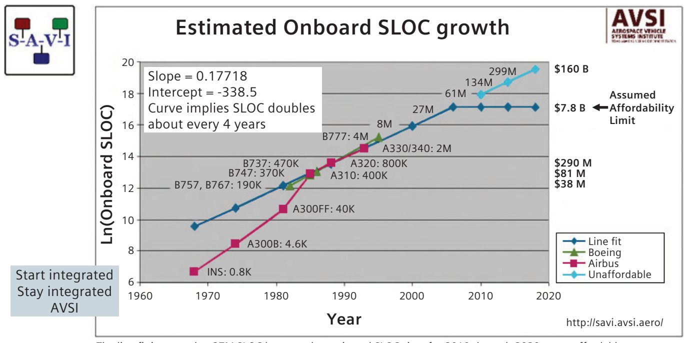
This process view changes our perspective from thinking about individual tools to how to enable a continuous communication model-based approach that leverages integrated systems engineering's ability to create surprise-free products.

Extending MBSE into the supply/design chain

Once we understand this integrated MBSE process, we can look to the left (to our customers) and right (to our suppliers) and realize they are part of our product development system of systems (SoS) that starts with customers and moves through us into our supply chain and back. This product architecture drives and enables communication for an entire product development chain. We call this a model-based design chain (MBDC) that expands the product development journey opportunity left-to-right and back from the customer to the suppliers and back.

This means if we are only thinking about our own organization, we're not getting all the potential value (for example, the value extends into the suppliers that are contributing to the product development process). The product architecture defines WHAT will be done and HOW WELL it will be done not only to inside development but also to the outside/purchased parts. Today, once it's decided what parts to purchase, a request for proposal (RFP)/specifications is sent to the supplier requesting quotes, etc. That spec describes the WHAT and HOW WELL 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 to check progress, to make sure designs are aligned, etc. Once delivered, systems integration brings all the component subsystems into one system, hopefully cooperating to deliver required functionality or sent back for redesign/updates when integration problems are discovered.

Per the Aerospace Vehicle Systems Institute (AVSI), a consortium of the major air-framers, this design, then integrate cycle/approach is no longer affordable. The system integration problem consumes almost half of the product development cycle and often takes longer than original design, making building airplanes from scratch no longer affordable (figure 17).



The line fit is pegged at 27M SLOC because the projected SLOC sizes for 2010 through 2020 are unaffordable.
The COCOMO II estimated costs to develop that much software are in excess of $10 B.

Figure 17: Aircraft unaffordability limits.

Of course, it's not just aerospace that has this problem; almost all multidomain 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 original equipment manufacturer (OEM) and semiconductor business, it's not uncommon for OEMs to wait for prototypes 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 integrated model to work from in developing their subsystem. With the help of PLM, periodically they "check in" models so OEM-level integration can start earlier and continuously, essentially eliminating the long, multiple, expensive serial system integration cycles. This can potentially save as much as half of the product development cycle because the product is continuously integrated. This is what we mean by start integrated, stay integrated (figure 18).

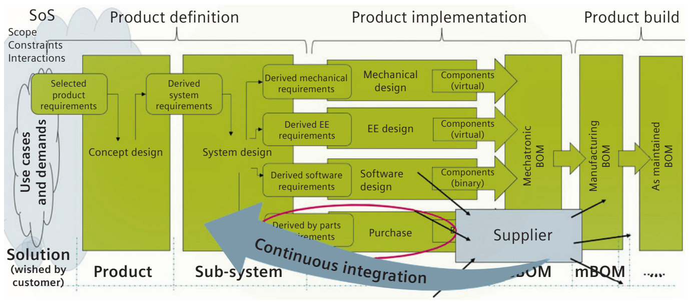


Figure 18: Supplier feedback loop to enable continuous integration.

If we could look at your development problem histories, we have high confidence that changes were made either on the OEM or supplier side that were not communicated/managed, resulting in system integration problems (discovered late and very expensive). This makes the integrated product architecture's value even greater when integrated with Teamcenter, keeping track of who has what version and ensures everyone (including suppliers) are 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
Management Consultant, Educator and Author

Problem in a nutshell

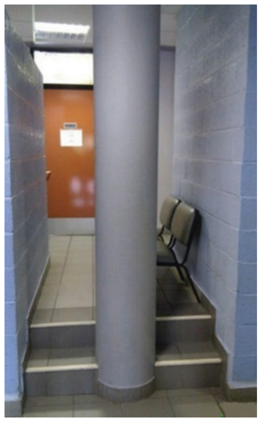
The fear of product development surprises is what keeps you up at night. We provide an integrated MBSE solution that enables continuous communication across domains through an integrated SoS product architecture that prevents product development surprises by ensuring we are always integrated across the entire design chain (from customer to OEM to suppliers and back). Remember, start integrated, stay integrated and you'll sleep much better.

Teamcenter advantage

Teamcenter offers you three important advantages:

1. By integrating SE/MBSE/product architecture with embedded standard methodology with Teamcenter PLM we enable continuous communications across organizations. This drives product development from a systems perspective under the watchful eye of an integrated product architecture and system design methodology that ensures the WHAT and HOW WELL are implemented by the downstream development disciplines (including suppliers) in an integrated way. Doing this enables built-in compliance to requirements and architecture alignment enabling organizations to start integrated, stay integrated (versus start design and integrate/fix later). This is kind of 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 up to half your program schedules for system integration. Imagine the value if you had confidence in your shared architecture to eliminate that program padding.
2. Integrating MBSE with the product lifecycle also integrates the product architecture with standard product lifecycle services like change, variation, workflow and more; for example, configuring the product also configures the product architecture, requirements, targets, test cases and interfaces across the entire design chain.
3. Finally, a scalable Teamcenter infrastructure enables global complex product development by ensuring everyone participating in product development, including suppliers, are on the same page and have continuous communication. This ensures that everyone, no matter what time zone, is guided by the product architecture and working from current information, creating integrated by design products.

Conclusion



The problem in almost all complex product development is keeping everyone on the same page to avoid 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 work around discussions that leave significant damage in its wake. There is some value in these monuments for future generations like this "Column of Shame" at a major engineering college that serves as a lesson/warning to others when an architecture mistake (or no architecture at all) is discovered too late to correct, thus the integrated SE/MBSE mantra of "Start integrated, stay integrated."

In conclusion:

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 WELL 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 and captures product architecture
- Unlike disconnected MBSE tools, the integrated architecture establishes the cross-domain dependencies needed to understand and manage complex cross-domain products
- You can discover and better communicate issues now versus later, allowing organizations and their supply/design chains to start integrated, stay integrated

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Xcelerator, the comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software, helps companies of all sizes create and leverage a comprehensive digital twin that provides organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

siemens.com/software

82097-C7 7/20 H