



Product Line Engineering: Using strategic reuse to tame IoT product development complexity



Daniel Moul

© 2015 IBM Corporation

Today we are going to look at some of the challenges development teams face when engineering for the Internet of Things.

There are a large set of complexities in most IoT solutions. Some are typical distributed systems challenges ... things like interoperability, security, performance, troubleshooting, and evolution of the system “in flight”. The IoT challenge is compounded by the many layers of a typical solution and the many actors & suppliers involved.

That’s more than we can address in this one session.

So we are going to focus today specifically on challenges that engineering teams face as products become smarter, more complex, and offered in an ever-larger product line.

We will start with a story of a company that is struggling with their transition to the IoT, and we’ll look at some of the root causes of this struggle. After that we’ll briefly look at PLE as a discipline, and I’ll finish with a quick overview of how the practices and tools that IBM offer can help. I’ll sprinkle in some examples from real companies as we go.

So let’s get started.

The story of Acme Manufacturing*



* fictitious

2

This is a fictitious story to illustrate the need for strategic reuse and Product Line Engineering.

Since 1948, Acme Manufacturing has been a successful manufacturer of widgets. Customers all over the world use Acme widgets in a variety of applications.

The story of Acme Manufacturing



New opportunity

Product
capabilities



Lately, Acme has been facing increasing design and market pressures. Their competitors are adding new capabilities at lower price points, and Acme is finding it hard to keep up.

The story of Acme Manufacturing



New opportunity

Product capabilities



Amount of software



```
001100
011011
000010
```

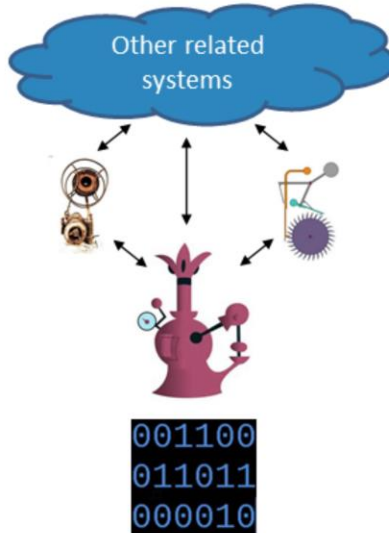
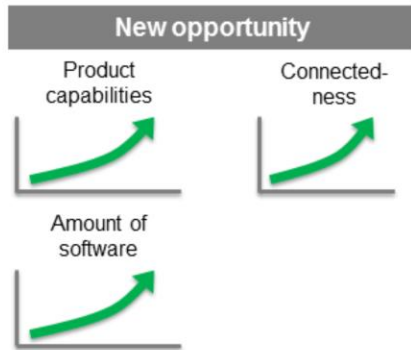
4

They've realized that in principle it's faster and cheaper to implement widget functionality in software, whereas previously it was implemented in mechanical and electrical systems.

The amount of software is growing quickly, and most of the value and competitive differentiation in new models are implemented in software.

Acme has never been a "software development" company, but it's clear that they need to become one.

The story of Acme Manufacturing

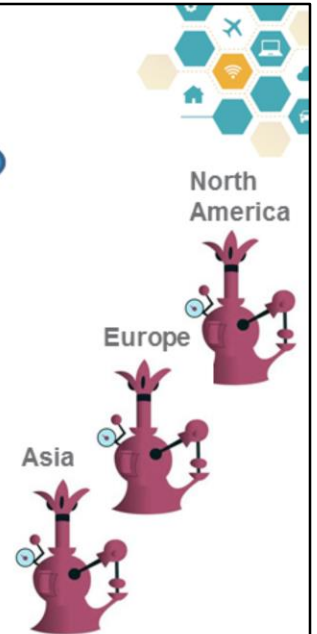
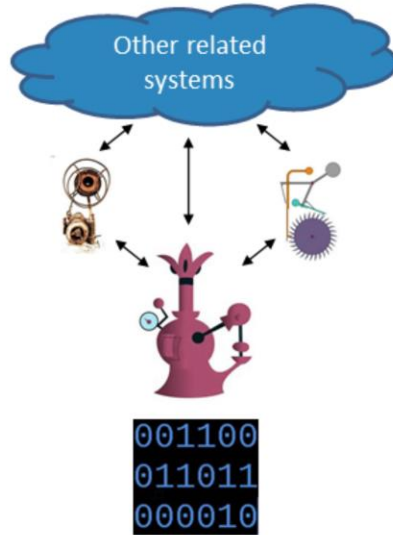
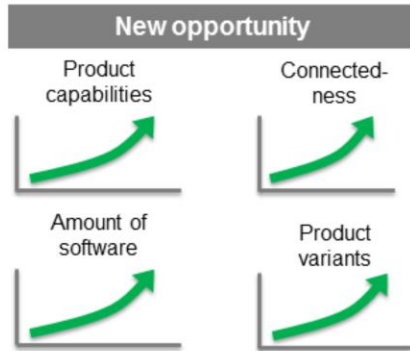


5

Also – and this is important for our discussion today – Acme widgets are increasingly connected—not only to other widgets, but to related systems that open up new revenue streams and provide even more capabilities to current customers.

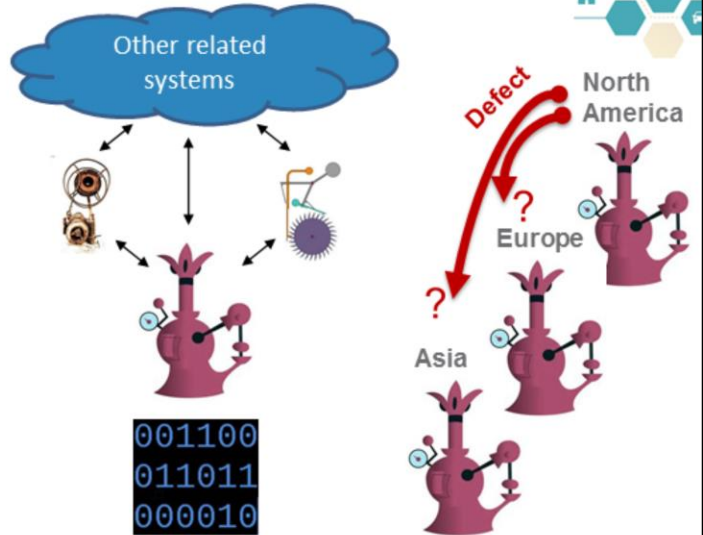
Some of these other systems are barely recognizable to Acme’s traditional core business. These include maintenance and warranty systems, customer relationship systems, and web and smartphone interfaces that augment product functionality for Acme end users. Cloud and back-end systems analyze data streams and make use of data insights and trend information from the web of connected widgets.

The story of Acme Manufacturing



Adding to the challenge, Acme's markets are no longer homogeneous. In the past, a single widget design could be used worldwide. But now, to be relevant in each market and sub-market Acme must offer widgets with unique characteristics. This is causing an uncontrollable proliferation of widget models.

The story of Acme Manufacturing



Acme engineers are feeling the competitive pressure to meet market demands, but are struggling to adapt to the increasing engineering complexity arising from all of these factors.

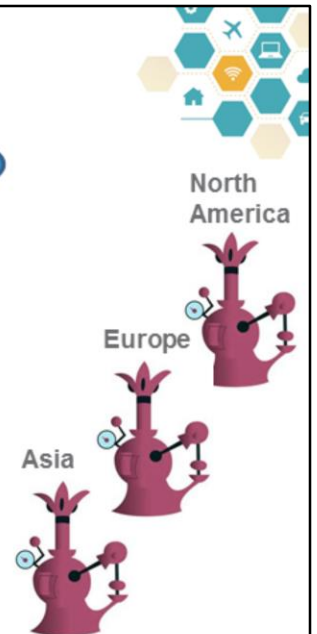
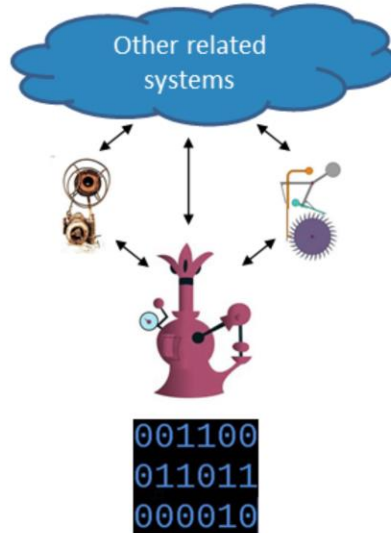
For example, if a particular widget model encounters a problem, Acme engineers can't be sure that the same problem isn't occurring in other models. Similarly if they fix the problem in one model, it's hard to figure out which other models also need the same fix. The amount of engineering data is increasingly exponentially as the number of models grows, and keeping track of it all is proving almost impossible.

Acme engineering has notified management that because they can't trace all of their engineering data across all of their widget models, they risk not complying with safety standards.

The story of Acme Manufacturing

Cost of Complexity

Development effort



8

It's hard to reuse existing designs, so it's common for new engineering work to nearly duplicate existing assets. Manual efforts to manage dependencies and changes across engineering disciplines are time consuming and error-prone. It's taking more and more person-months of effort to complete the development of new widgets.

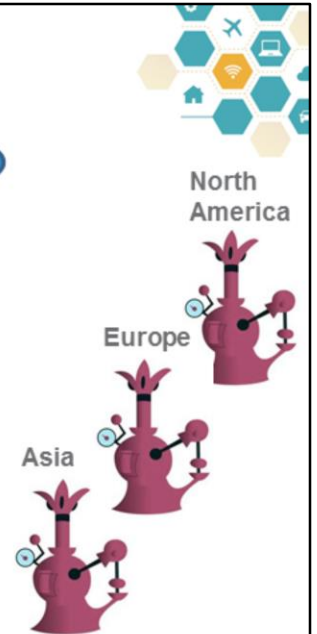
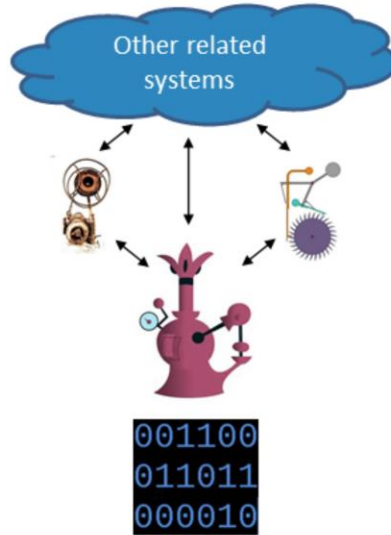
The story of Acme Manufacturing

Cost of Complexity

Development effort

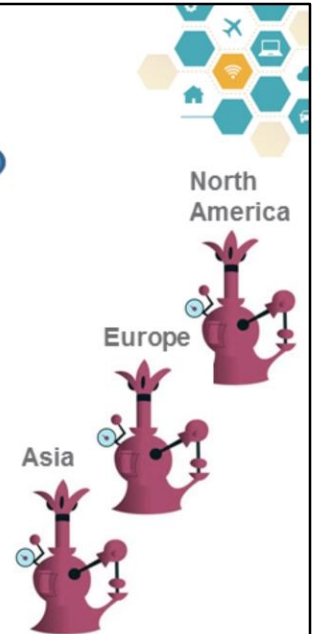
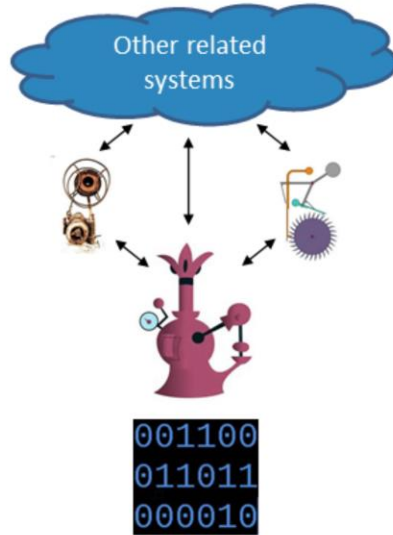
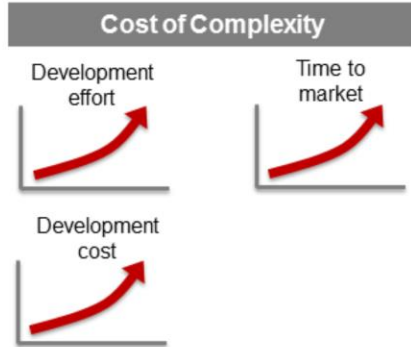


Development cost



So their engineering costs are going through the roof.

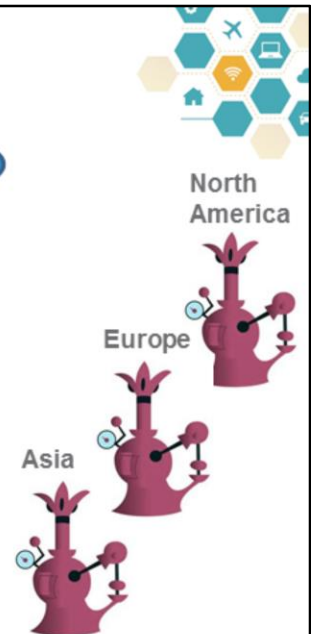
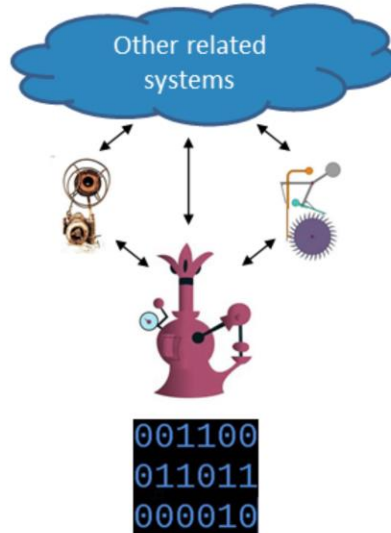
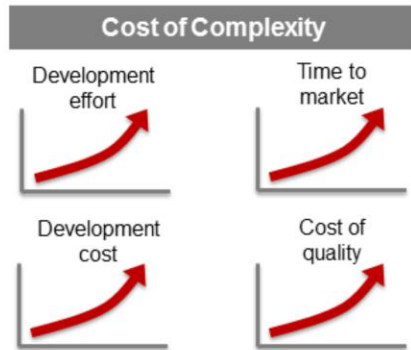
The story of Acme Manufacturing



10

And it's taking longer to develop, test and qualify new widget models. Time to market is increasing – just when their competitors seem to be shorting their development cycles.

The story of Acme Manufacturing



Acme is finding it necessary to add engineers in the QA team to limit the growth of the time needed to test, fix and qualify new widgets models. The cost of quality is going up, and they seem unable to address the front-end causes of their unfortunate QA trends.

After more than six decades in business, Acme Manufacturing is starting to lose both customers and profitability.

How did they get into this situation? And what can they do about it? We'll look at that next.



One successful product



12

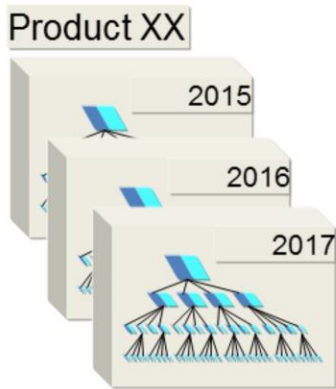
Most companies or company divisions start by offering a single product.

The engineering team deals with the complexity of the product through a “divide and conquer” approach. That is, they decompose the product into systems and subsystems. This may include...

- Hardware
- Electronics
- Embedded software on the devices
- Cloud and back-office software



Evolves overtime



13

It gets more complicated fast. Engineering artifacts created in the development process change over time to reflect changes in the product. These may include

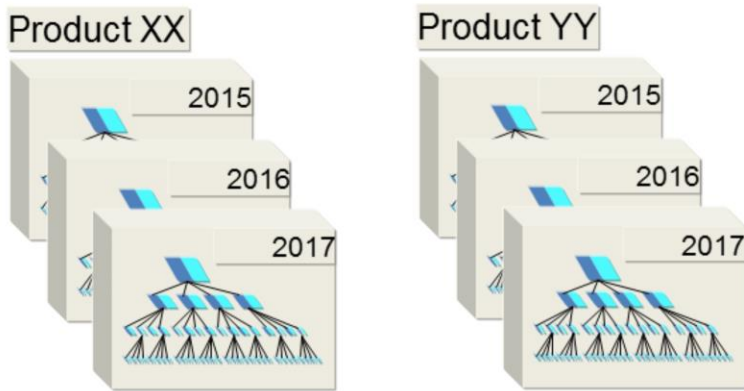
- New features
- New suppliers
- Replacement of some components with newer ones

This is variation over time, what we call temporal variation.

Yet the old products are still in the field and need to be serviced, repaired, and upgraded. So the engineering and field service teams need to keep design artifacts current and available. They now face a version management challenge.



And quickly turns into multiple products or programs



14

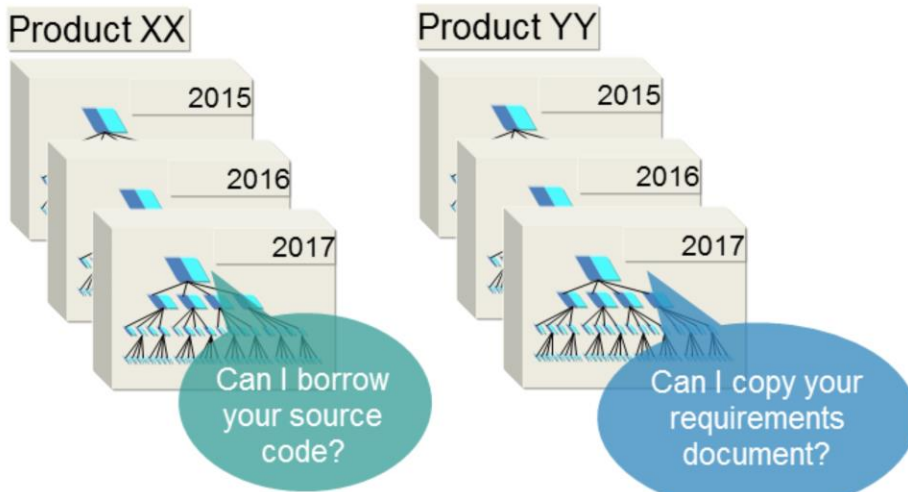
Once successful product leads to others.

You can sell a second product that's very similar to the first one: maybe remove some features and charge less, or address regulatory requirements and certification for a new geographical market.

It's common sense to everyone that the engineering team should reuse as much of the design and manufacturing assets as possible. Anything else adds unnecessary cost and time.



Sharing speeds things up



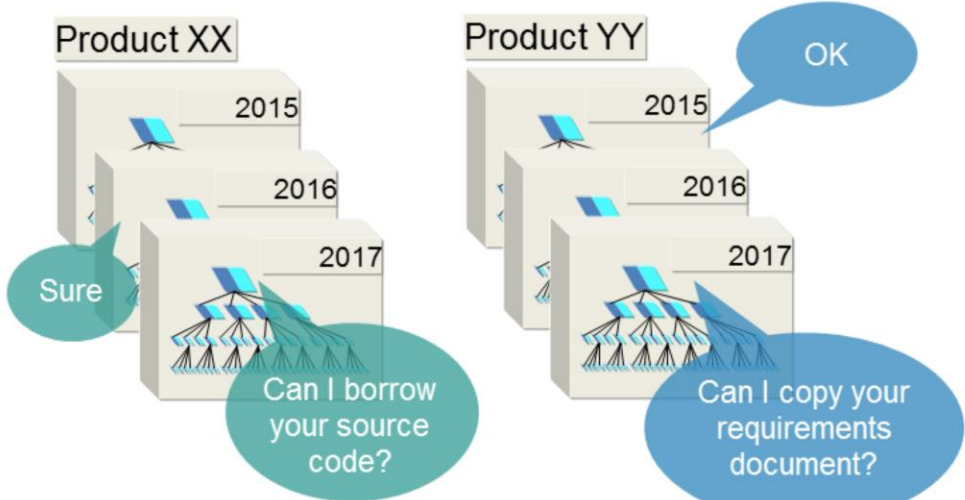
15

So people share, typically by copying.

We call this “clone and own”. You copy it, you own your copy.



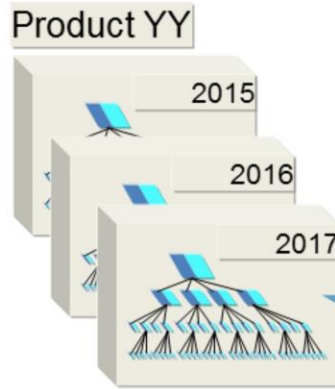
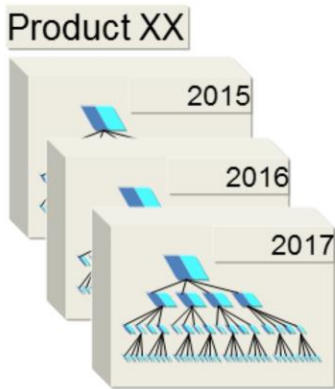
Sharing speeds things up



It speeds things up. But typically reuse is ad hoc, not systematic, and not well managed.



For a while



Wait!
Didn't I copy my
requirements
from you?

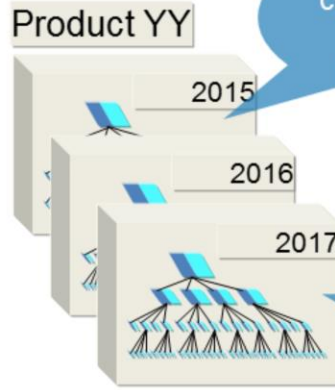
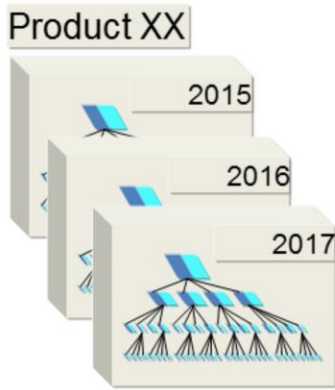
17

And sooner or later the deferred cost of “clone and own” approach comes due.

For example, when someone finds a defect in the requirements, someone has to ask: which other variants have this defect? In which products must we fix it?



For a while



We originally copied ours from Product XX. It's XX's fault.

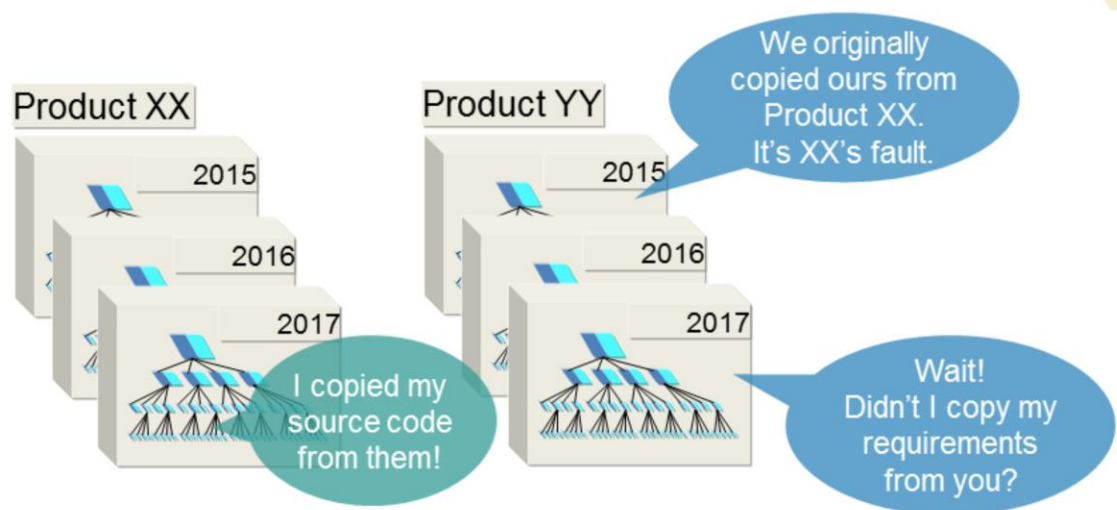
Wait! Didn't I copy my requirements from you?

18

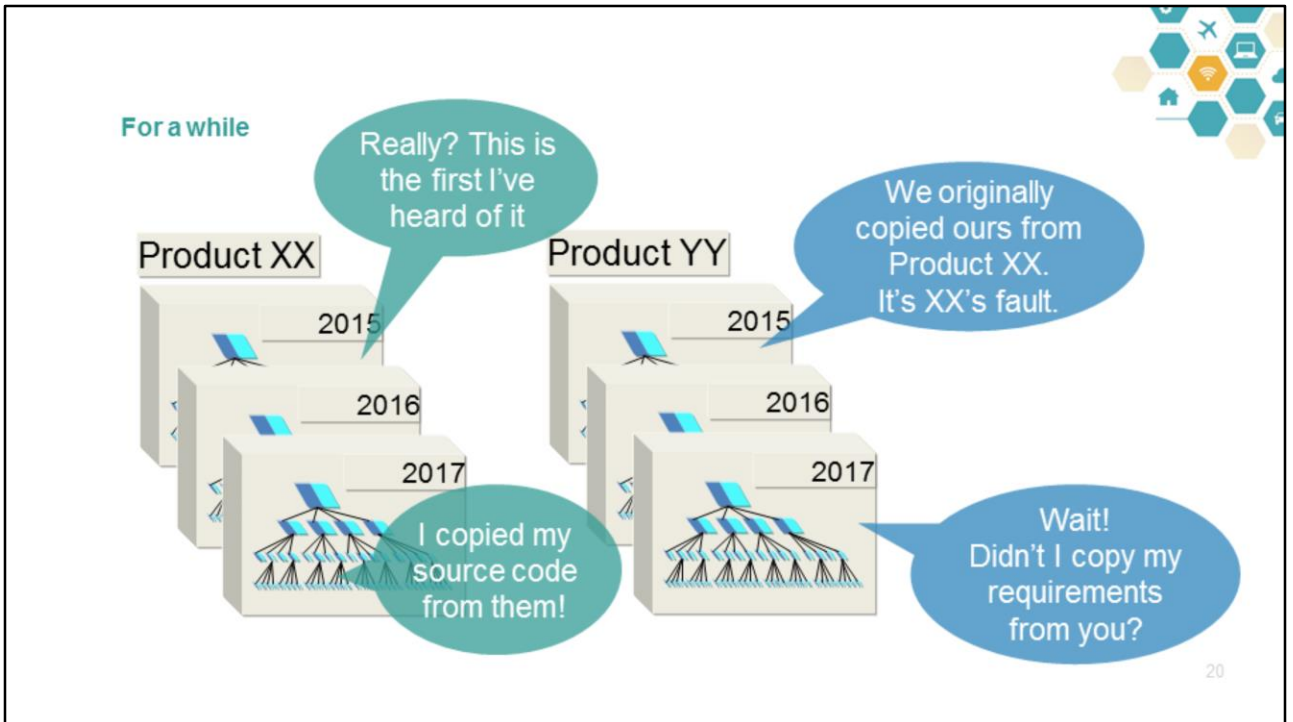
This can lead to a long and unsatisfying research project that relies on the “tribal knowledge” of old-timers – and leaves the team without confidence that the analysis is complete without omissions.



For a while



Similarly teams reuse software components. That's a fantastic way to speed up development.



But without good governance and record-keeping, expensive and awkward dependencies can come out of the woodwork at the most inconvenient times.

Sound familiar? Next I'll share five statements we hear that are symptomatic of the challenges we have just discussed. Have you heard them in your organization? Have you said them yourself?



Do you hear comments like these?

Each of our products has unique capabilities, yet we need to better reuse the **common 85%** across all of them.



Do you hear comments like these?

We need to do our **engineering work in parallel** on multiple products or releases, but it's hard to coordinate across teams and products to keep everything in sync.



Do you hear comments like these?

It is **difficult to determine the impact** when there are engineering change orders, for example, when a supplier changes something.



Do you hear comments like these?

When a defect is identified, we **waste time and money** determining which products are impacted. And we **wish we had more confidence** that we are finding the full set of impacts.



Do you hear comments like these?

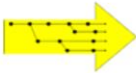
It's a time-consuming and potentially error-prone task for our engineers to **reconstruct their workspace** when switching to work on a different product variant or release.



Product Line Engineering enables...



Strategic reuse of engineering artifacts (without simply **copying** them)



Engineering dozens, hundreds or even **thousands** of product variants in a controlled manner



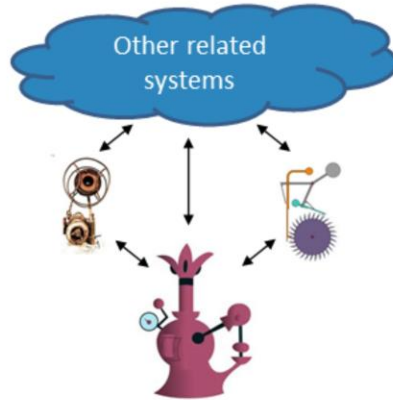
Reuse of what's **common** and specification of what's **unique** in each variant



Evolving a product line as a strategic initiative



Strategic reuse
in many places



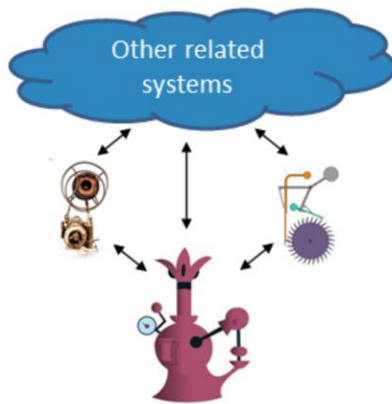
27

Now let's look at what can be reused, and how the IBM solution helps teams to accomplish that.



Strategic reuse
in many places

Mechanical
Electrical
Embedded software

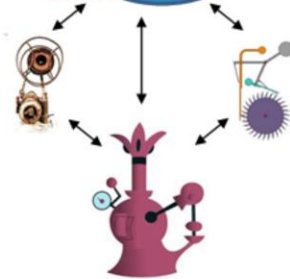


Your widget probably has all three aspects, and each has potential for reuse.



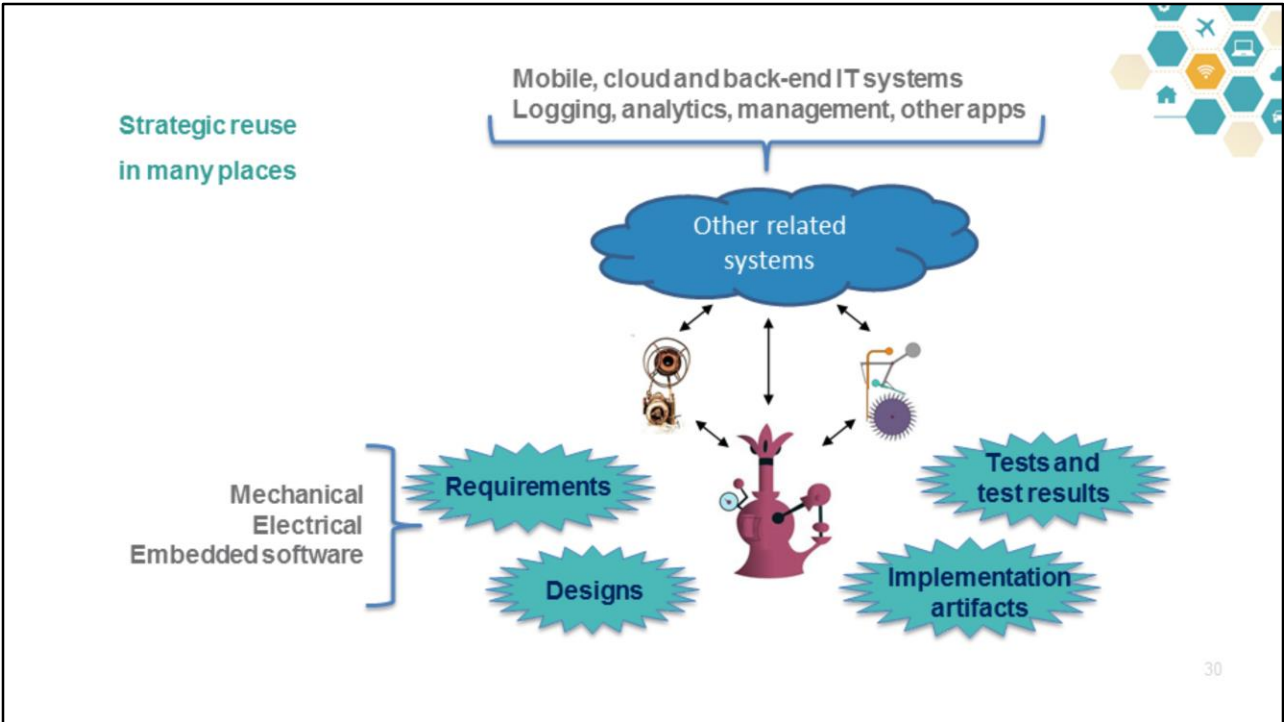
Strategic reuse
in many places

Mobile, cloud and back-end IT systems
Logging, analytics, management, other apps



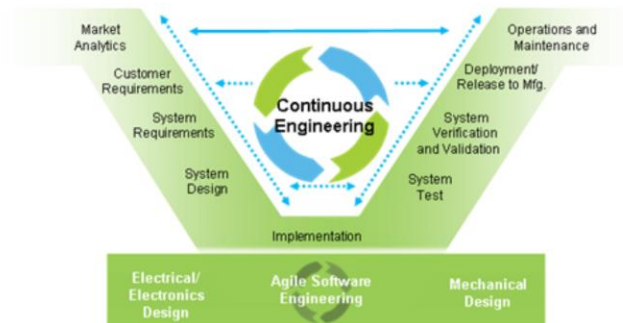
Mechanical
Electrical
Embedded software

Likewise there is potential for reuse in the mobile, cloud and back-end IT systems that power your end-to-end IoT solution.



Specifically I'm referring to the engineering artifacts that define the product and are used to test it.

IBM IoT Continuous Engineering Solution



31

A cross-discipline engineering effort addresses what's commonly called the "system V". Teams can be more or less formal in implementing a systems V.

A group of people follow a set of engineering practices, making use of a set of tools.

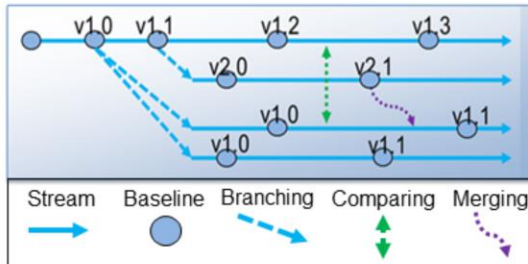
The better the collaboration and productivity of the engineers, the better and more cost effective is the outcome.

At IBM we promote Continuous Engineering practices and we offer the IBM IoT Continuous Engineering Solution.



Approaches to Product Line Engineering

Development Streams



33

With the Continuous Engineering solution teams can create requirements, designs, tests, and implementations that are related – and all under configuration management. Perhaps you have experienced the benefits of software source code under configuration management, or a Bill of Materials tool with the BOMs under configuration management.

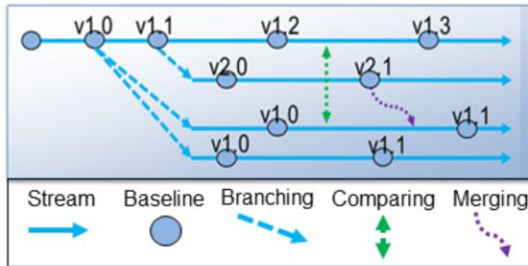
You can create insulated streams of development work (again, of requirements, designs, tests and implementation – or any subset of those you wish) – and you can baseline them together, branch them to create a new product variant, etc.

And artifacts are not copied. They are reused where the same version of an artifact is in multiple streams or baselines. The system keeps track of the dependencies – and computers are a lot better at that kind of things than people (even good engineers). So the engineers can spend less time on bookkeeping and unsatisfying research projects and more time on the creative work they enjoy and that adds value to their business.

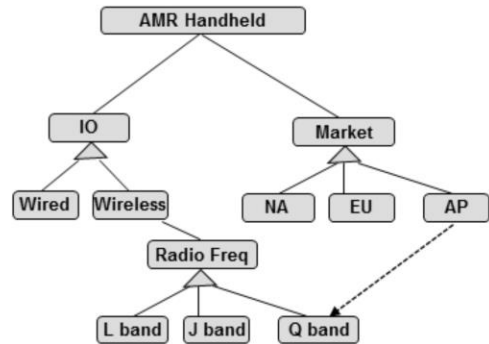
How is variation specified? Some teams use attributes on requirements or model their features as requirement artifacts...

Approaches to Product Line Engineering

Development Streams



Feature Model



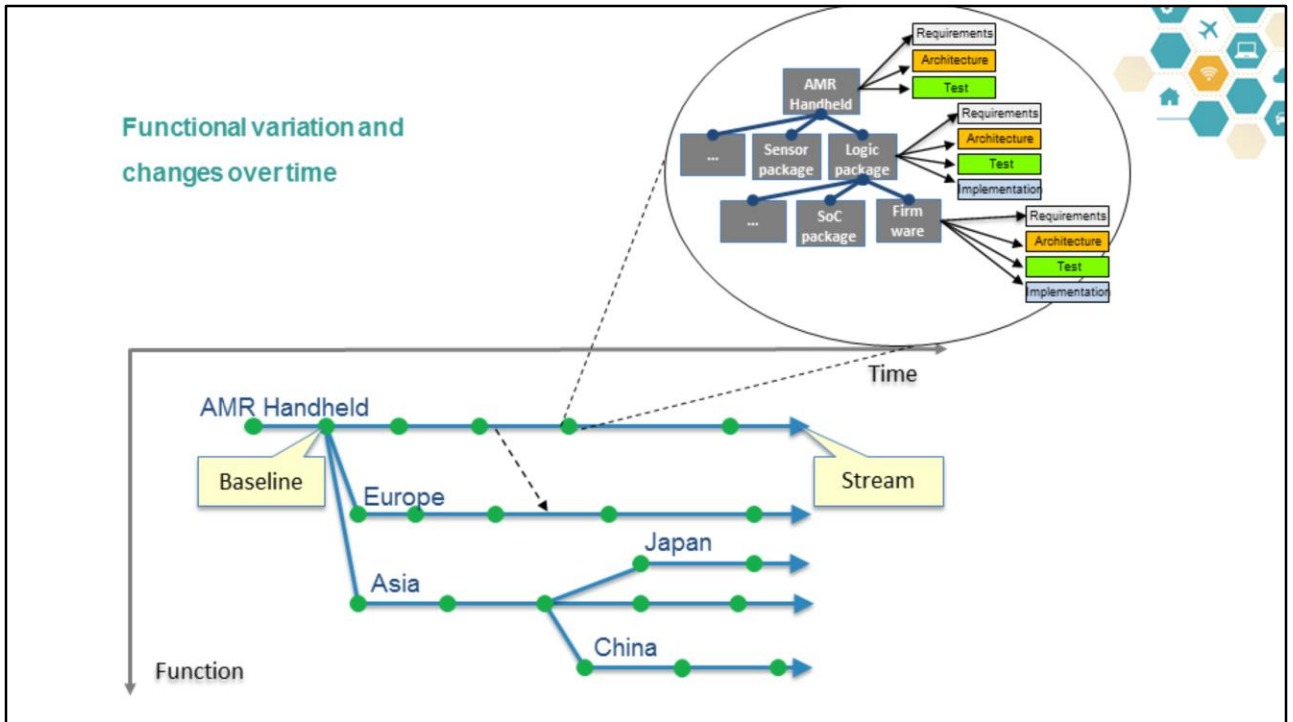
34

Other teams use an external definition of their feature model.

It's a common way to represent the variation points in a product line along with higher-level abstractions including constraints among features and feature profiles.

The IBM solution works with leading feature modeling tools, including those provided by Big Lever Software, a US company, and pure-systems GmbH, a German company.

Some of our clients have developed their own feature model databases over the years and use these to drive their engineering tools.



Each green dot is a baseline of the whole system/sub-system hierarchy, and engineers can return to a past baseline by simply selecting it in their tool's user interface. The tools show the user the right artifacts at the right versions with the right relationships – as captured in that baseline.

And a baseline can be the starting point for a new product variant, as you see in this picture.

There are other more sophisticated ways of organizing product variants, but I hope you get a sense of what's possible:

- No more complicated spreadsheets manually tracking all the baselines and sub-systems making up a particular product build.
- No more time-consuming reconstruction of an engineer's work space when they need to review the engineering artifacts to trouble-shoot a problem or participate in an audit
- Simpler and more complete analysis when analyzing a potential change order or figuring out where a defect originated and which product variants are impacted



Crawl, walk, run

1. Ensure a foundation of integrated methods and tools across engineering disciplines
2. Develop comprehensive approach to configuration management and coordinate across product build space
3. Ensure that requirements are well-managed today
4. Identify opportunities for strategic reuse
5. Choose your product line approach
6. Plan and implement transformation: in organization, development practices, etc.
7. Take a measured approach

36

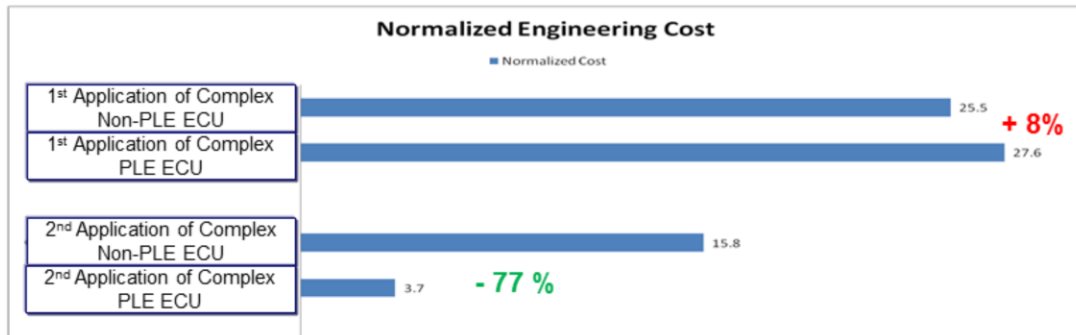
The “journey of a 1000 miles” cliché applies. You have to walk before you run.

Product line engineering requires effective, disciplined work in the underlying engineering domains.

Here are some suggestions for growing in your maturity across engineering disciplines.



GM started a PLE approach in software engineering
with astounding results:



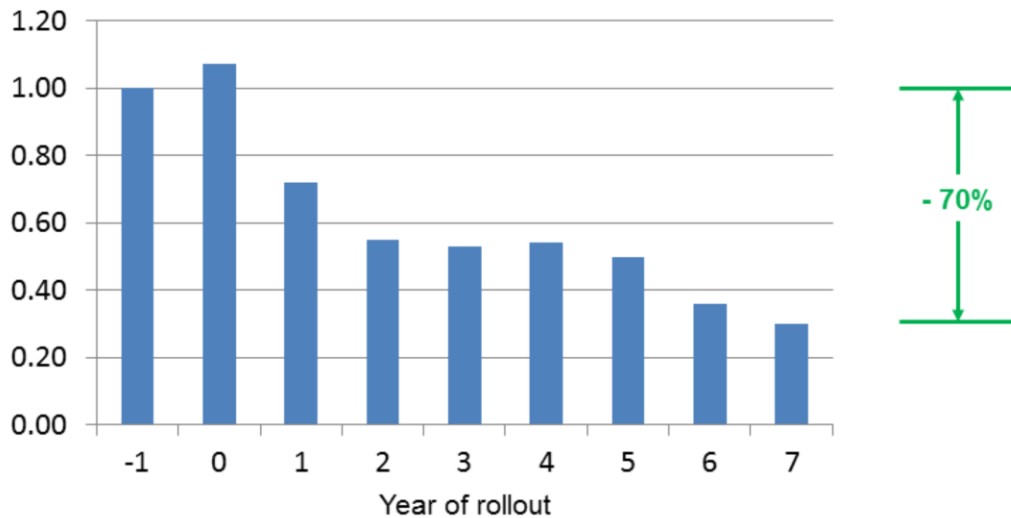
They compared the cost of creating an ECU and reusing it the old way and using a product line engineering approach.

The first version of the PLE-designed ECU did cost more; it is harder and more expensive to design for reuse.

But the second use cost less than a quarter of the first, and I can imagine subsequent uses could cost even less.




Normalized warranty claim cost





They found that in the first years after applying PLE methods, warranty costs rose modestly. If there is a defect, it will be in every product, not just one.

But GM has seen reduction of nearly 70% in warranty costs after 7 years! I'm sure that gets senior management's attention.



Automotive 



BOSCH

... Connected vehicles, assisted driving (lane change alerts, emergency braking), driverless vehicles

"Tool-based product line engineering helps Bosch to faster tailor its products to meet the needs of world-wide markets."

Before we end let me briefly mention three companies in different industries that are on this journey.



datamato[®]
Technologies

IT 

... IT consulting for multiple manufacturing industries

"To meet its objective of delivering high-quality products and services, Datamato depends on leading tools. We leverage product line engineering, and software development capabilities from IBM will help us provide our clients with a competitive advantage."



Aerospace &
Defense



... Embedded systems, systems engineering, embedded software, complex connected devices

"This capability will allow us to avoid the 'clone and own' technique, and the resulting attribute maze that has hindered our efforts to manage system requirement versions and variants."



Learn more

1. Read the free eBook: [Tame Complexity through Product Line Engineering](#)
2. Get Continuous Engineering resources on [ibm.com](#) and [Jazz.net](#)
3. Watch some short videos on [YouTube](#) or [IBM developerWorks](#)
4. Read blogs: [IBM Big Data Hub](#), [Continuous Engineering](#), [Jazz.net](#)
5. Explore feature modeling with partners [BigLever Software](#) and [pure-systems GmbH](#)

42

I encourage you to learn more.

And I wish you well on your IoT journey.

1. Read the free eBook: [Tame Complexity through Product Line Engineering](#)
2. Get Continuous Engineering resources on [ibm.com](#) and [Jazz.net](#)
3. Watch some short videos on [YouTube](#) or [IBM developerWorks](#)
4. Read blogs: [IBM Big Data Hub](#), [Continuous Engineering](#), [Jazz.net](#)
5. Explore feature modeling with partners [BigLever Software](#) and [pure-systems GmbH](#)



Product Line Engineering:
Using strategic reuse to tame
IoT product development complexity



Daniel Moul

© 2015 IBM Corporation