

Requirements and UML for Product Lines

Martin Bakal – PLE Market Manager



IBM copyright

■ © Copyright IBM Corporation 2009. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.



Agenda

- **Industry trend - build product lines**
- The role of architecture and modeling
- IBM and Big Lever help move to multiple product focus
- IBM product line solutions
- Customer success stories

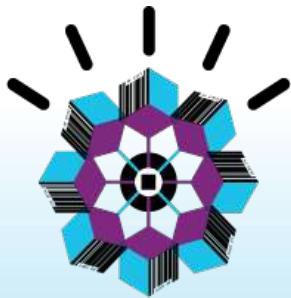


Businesses are facing an unparalleled rate of change

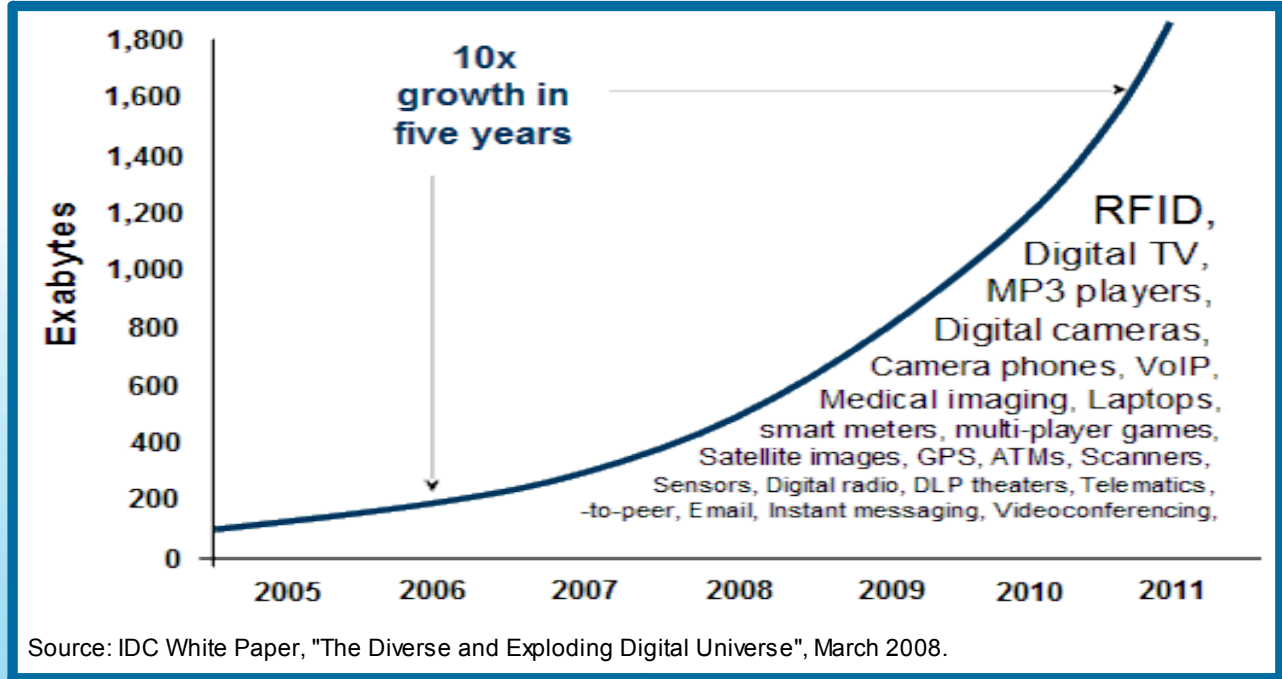


Intelligence is being increasingly infused in devices and systems, making them “smarter”

*The world is coming **10x** more instrumented with connected devices doubling to over **1 Trillion***



Smart Products



INSTRUMENTED



Can measure, sense and see the exact condition of everything

INTERCONNECTED



Enables people, systems to interact in entirely new ways

INTELLIGENT



Responds to changes quickly and accurately optimizing for future events

The **key to business success** depends on the infusion of new ideas about how software-based products and systems are brought to market



Create new value and innovate faster.



Reduce the costs of development and production.



Improve product quality and reduce risk.



Smarter communications



Smarter health care



Smarter personal devices



Smarter hybrid vehicles



Smarter energy



Smarter defense systems



Software Based Product Lines

The New Frontier for Systems and Software Innovation

- Most companies build multiple product variants
 - ▶ No one builds just one
 - ▶ Virtually every reuse strategy is performed in the context of a product line
- Increasing the size of product line provides the opportunity for increased revenue by offering more targeted products
- Companies need **economy of scale** in product lines for profitability
 - ▶ In manufacturing, **greater profitability** is achieved by investing in an **efficient means of production** – manufacturing infrastructure and shared product assets – that can be used to deploy different “flavors” of a product
 - ▶ As product differentiation and innovation move from the physical attributes to software-based features, the need for an efficient means of production for **software-based product lines** has become universal

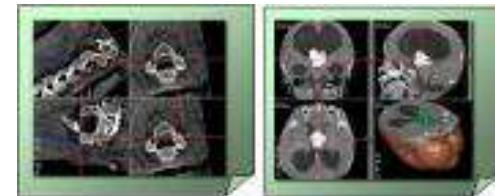


Software Based Product Lines Defined

- We define a software-based product line as:
 - ▶ A portfolio of similar products (or systems) with variations in features and functions, rather than just an individual product

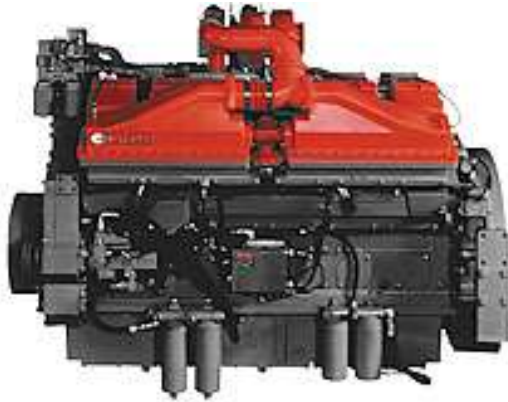


Product Portfolio of Similar Systems



Software Product Variation

Automotive example - Cummins



- **An engine may have:**
 - ▶ **6 ECM's**
 - ▶ **18 cylinders**
 - ▶ **12 turbo's**
 - ▶ **6 datalinks**
- Reference SPLC Hall of fame
- <http://splc.net/fame/cummins.html>

Telecommunication example

- Nokia Mobile Phones produces a wide range of mobile phones. Currently 32 different phones are manufactured covering six different protocol standards, a wide variety of functional features and capabilities, different user interface designs, and many platforms and environments.
- Reference SPLC Hall of fame
- <http://splc.net/fame/nokia.html>



The Product Line Challenge

Relationship between complexity and variance

- Multiple types of weapon systems...
 - ▶ Surface to air
 - ▶ Air to Air
 - ▶ Air to Ground
- Variants in industrial energy (Wind Turbine)
 - ▶ climate, terrain, geography
- Cell phone variations
 - ▶ Country, language, GPS, look
- All of these share many parts of their design but have variations in them



Agenda

- Industry trend - build product lines
- **The role of architecture and modeling**
- IBM and Big Lever help move to multiple product focus
- IBM product line solutions
- Customer success stories



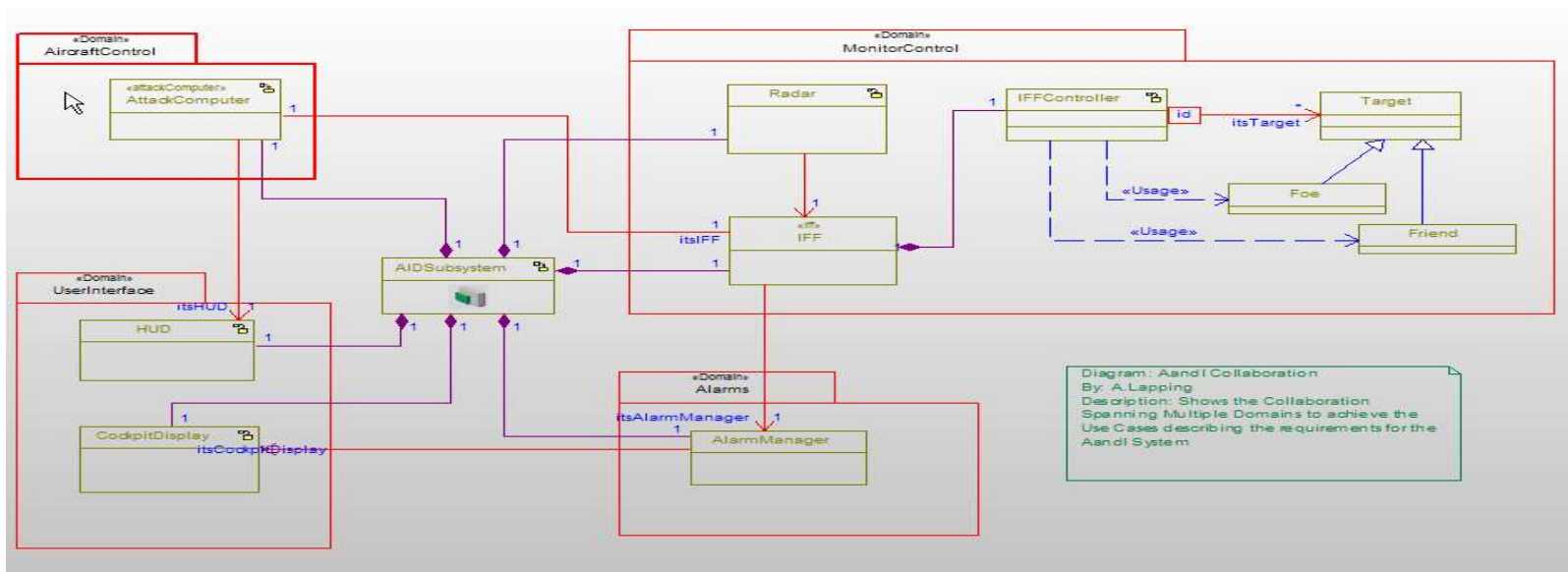
What have we learned

- Those companies that understand their architecture were able to evolve their portfolio successfully
 - ▶ However an architecture can evolve over time
- Systems Engineering practices emerged to play a greater role in product lines
- We need a much more efficient means of producing and delivering product line assets



The Role of architecture in product lines

- Designs are complex and very interrelated
- Need a good architecture to truly understand them
- Need an architecture that grows as you develop which can evolve from existing source code



What is Systems Engineering (SE)?

“Systems Engineering is an **interdisciplinary** approach and means to enable the realization of successful systems. It focuses on defining **customer needs** and required **functionality** early in the development cycle, documenting **requirements**, then proceeding with **design** synthesis and system **validation** while considering the **complete problem**....”

INCOSE

<http://www.incose.org/educationcareers/pdf/12-roles.pdf>

- Requires “Systems Thinking” – seeing the big picture & understanding the broader impact of changes/decisions
- A hierarchical approach is often used to manage complexity
 - Brings clarity and distinction between the problem space and the solution space*
- Strives to construct logical systems that are:
 - Predictable, Competitive, Profitable, Compliant*



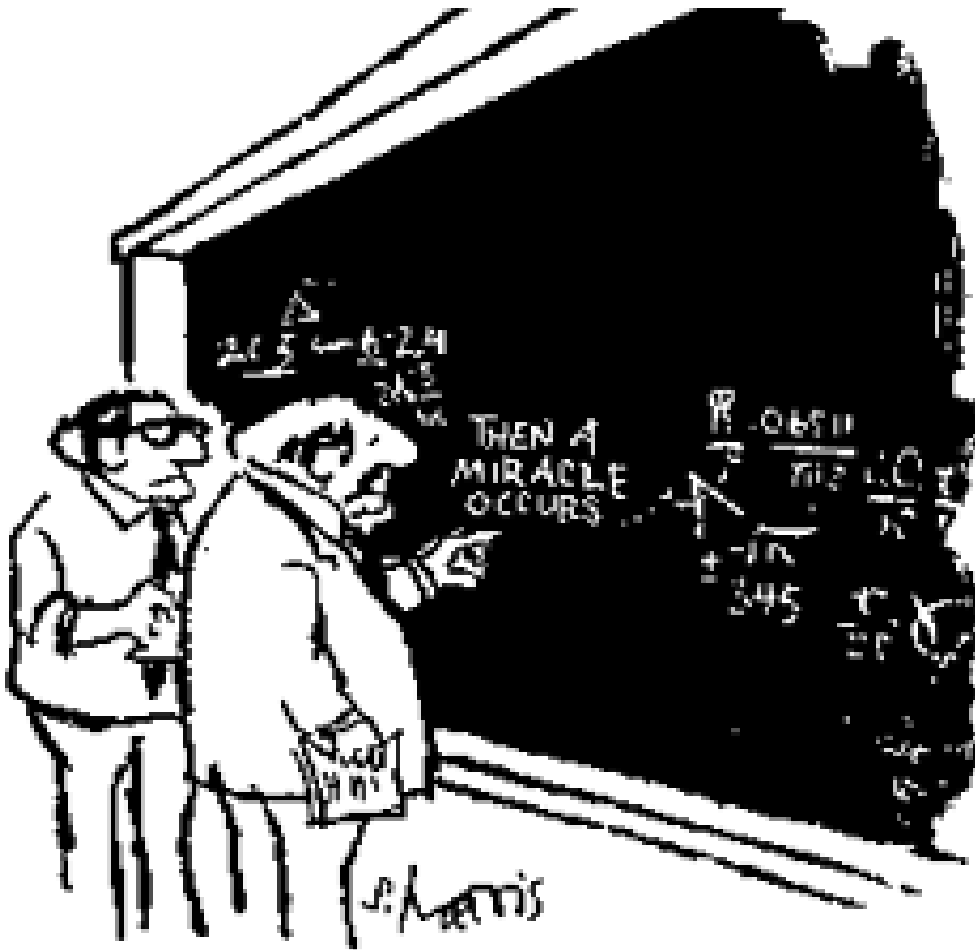
Model Driven Systems Engineering and Architecture

- Picture tells a thousand words
 - ▶ PLE makes designs a little more complex – graphics simplifies them
 - ▶ In order to find patterns for reuse need to see them graphically
- Behavioral simulation and execution
 - ▶ PLE helps target different environments – Modeling makes this easy
 - ▶ Makefile maintenance is tough – New sources files needs to propagate to all products
 - Easy for Rhapsody with automatic generation of product specific makefiles
 - ▶ Filter testing based on products/components
- Documentation
 - ▶ Ability to create product specific documentation without re-writing it
- Collaboration
 - ▶ UML and SysML are different dialects of the same standard
 - This allows everyone to speak the same language



The Power of Modeling

For Systems Engineering and Requirements Engineering



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO"

- ✓ Abstract representation of requirements
- ✓ Aids in understanding the problems and potential solutions
- ✓ Communication method for stakeholders to determine the needs of the software based product/system

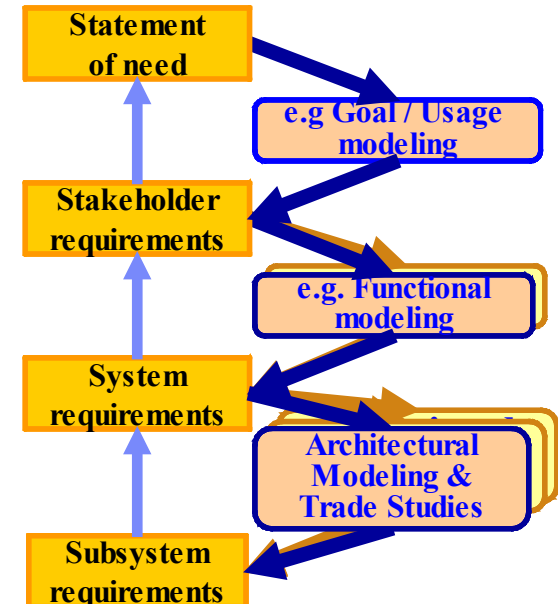
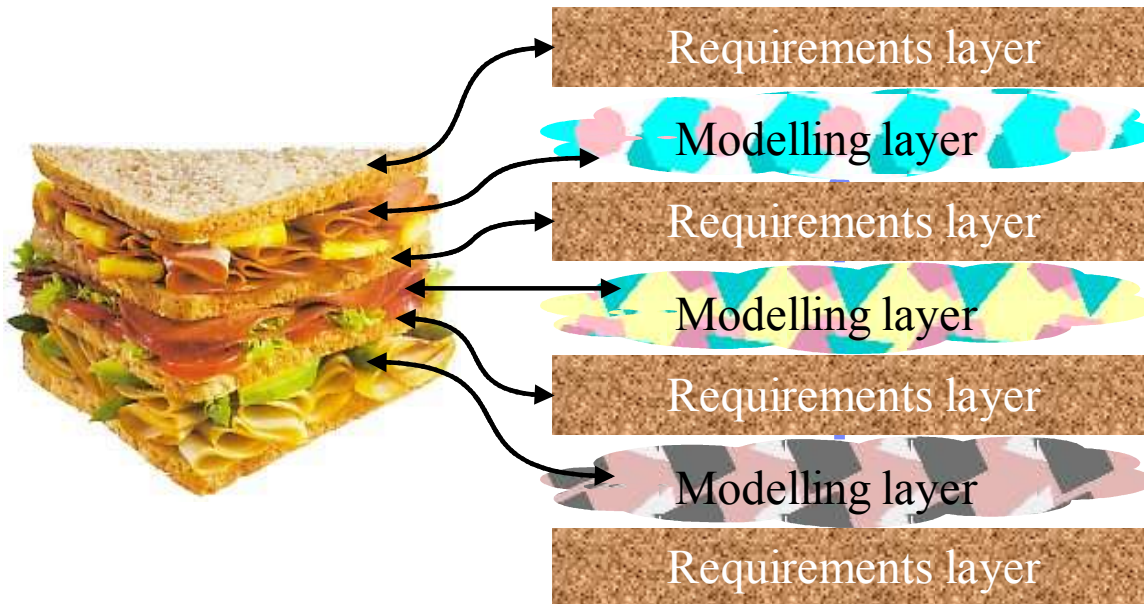
"...by relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect, increases the mental power of the race."

Alfred North Whitehead
British mathematician,
logician and philosopher



Integrating Requirements and Modeling to Drive Product Development

- There are two major activities in the systems engineering process:
 - ▶ To analyze the input requirements and create a model
 - ▶ To derive the output requirements from the model
- These steps are applied **recursively** and **iteratively** through the layers until you have a working solution and reach closure

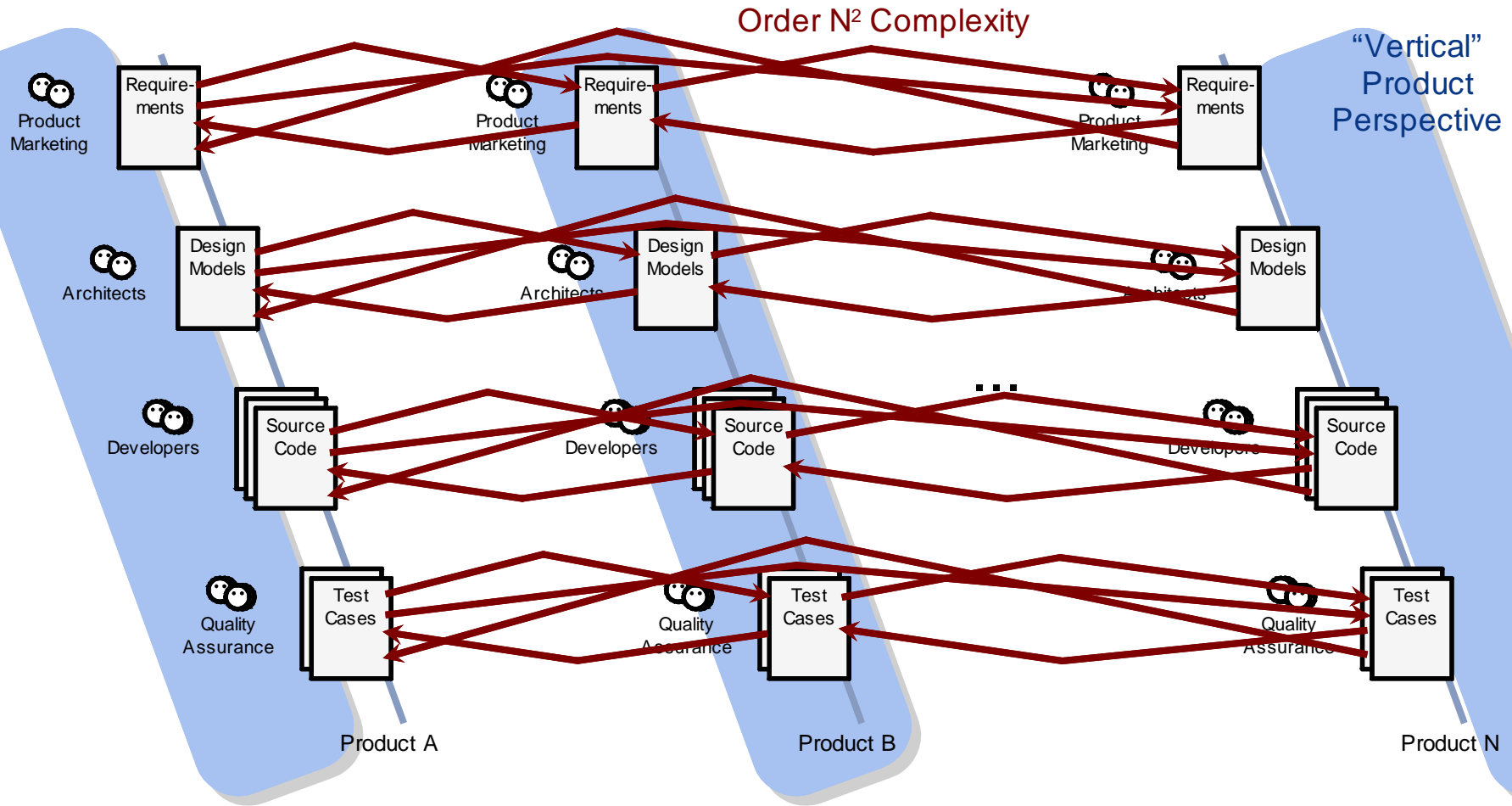


Agenda

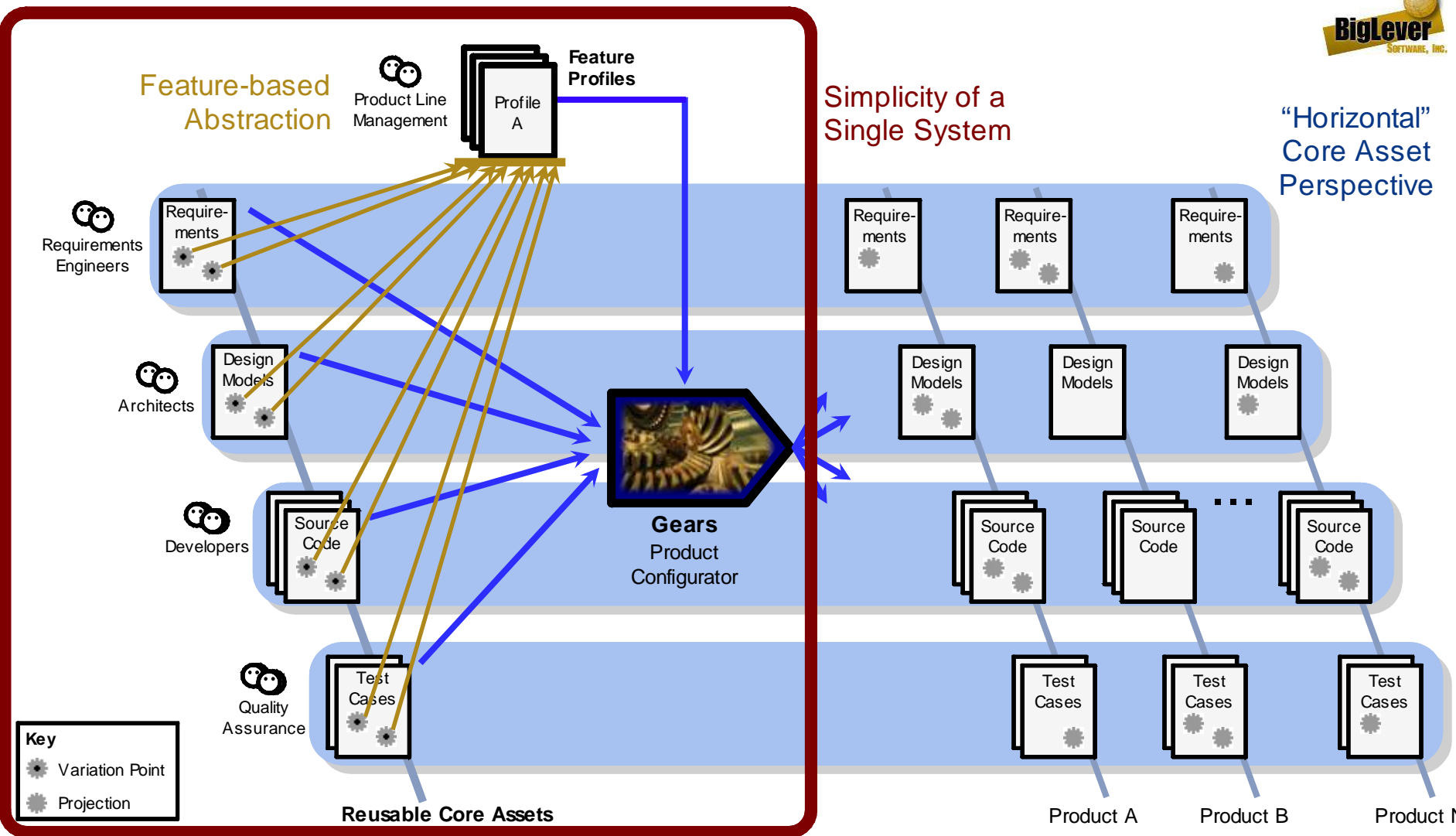
- Industry trend - build product lines
- The role of architecture and modeling
- **IBM and Big Lever help move to multiple product focus**
- IBM product line solutions
- Customer success stories



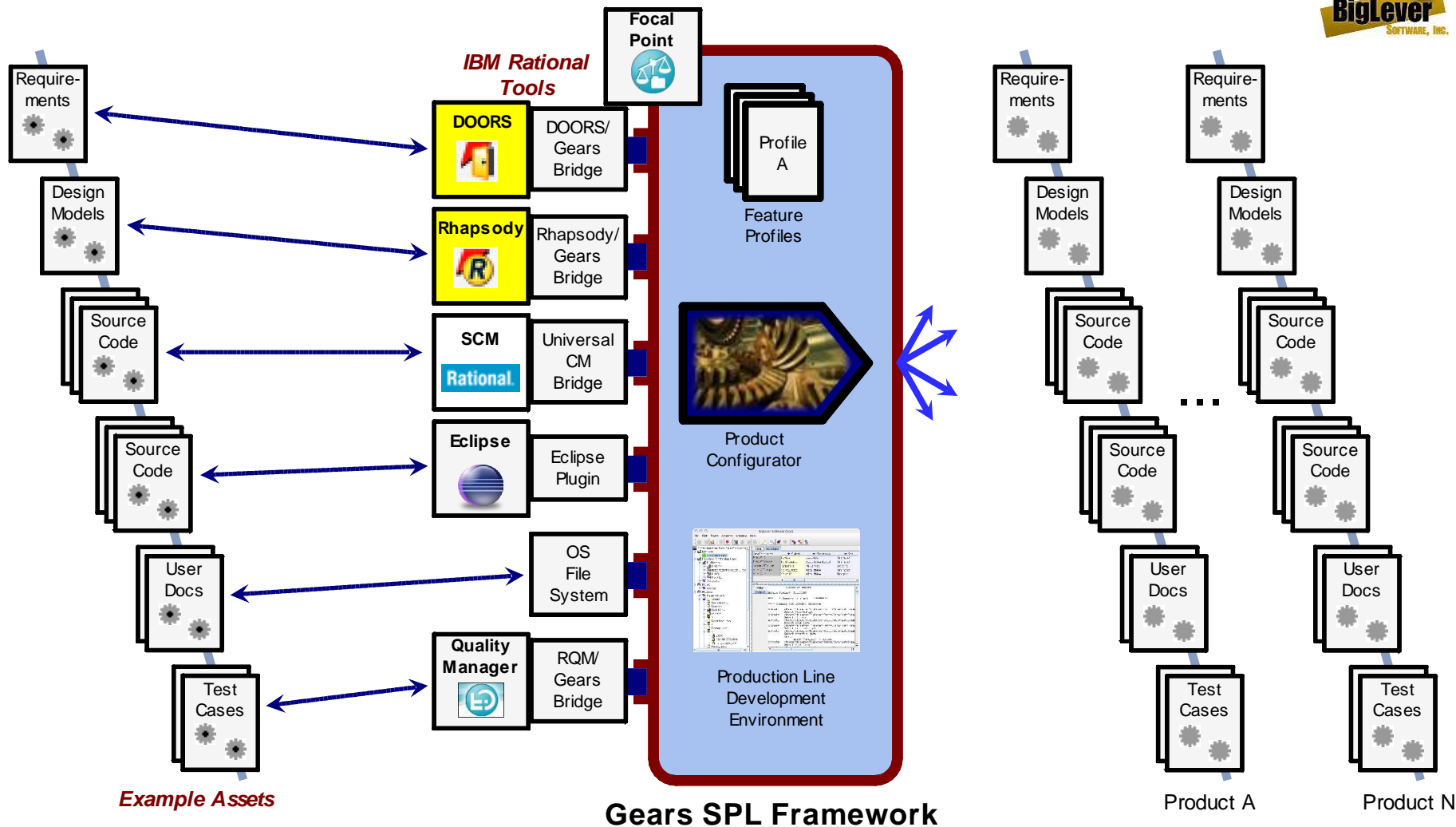
Complexity in Product-centric Thinking Impedes Portfolio Production



Shift in Perspective to an Efficient Means of Production



The IBM BigLever Systems and Software Product Line Solution



Agenda

- Industry trend - build product lines
- The role of architecture and modeling
- IBM and Big Lever help move to multiple product focus
- **IBM product line solutions**
- Customer success stories



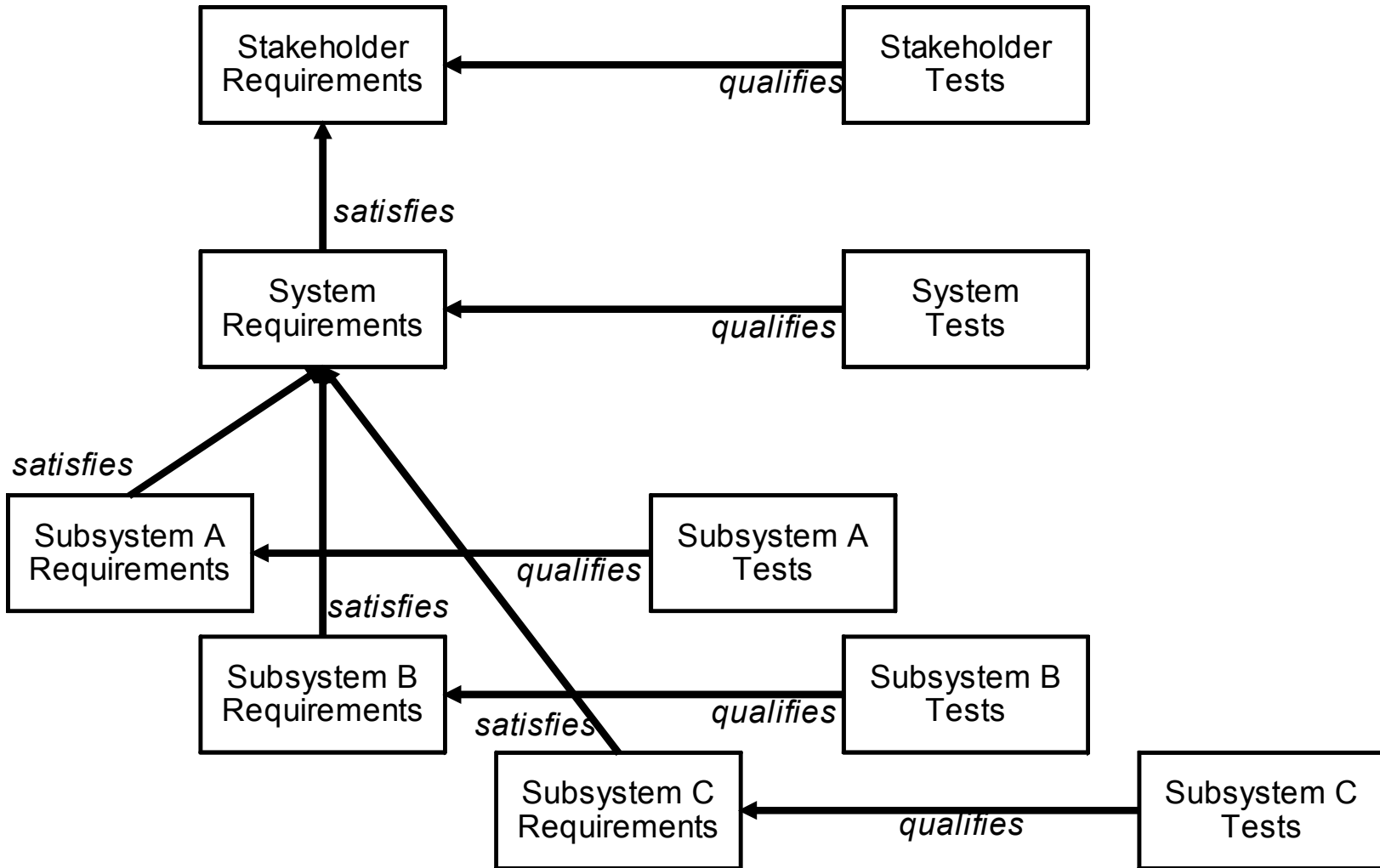
Software Customers' Evolution to Product Lines

Requirements Engineering

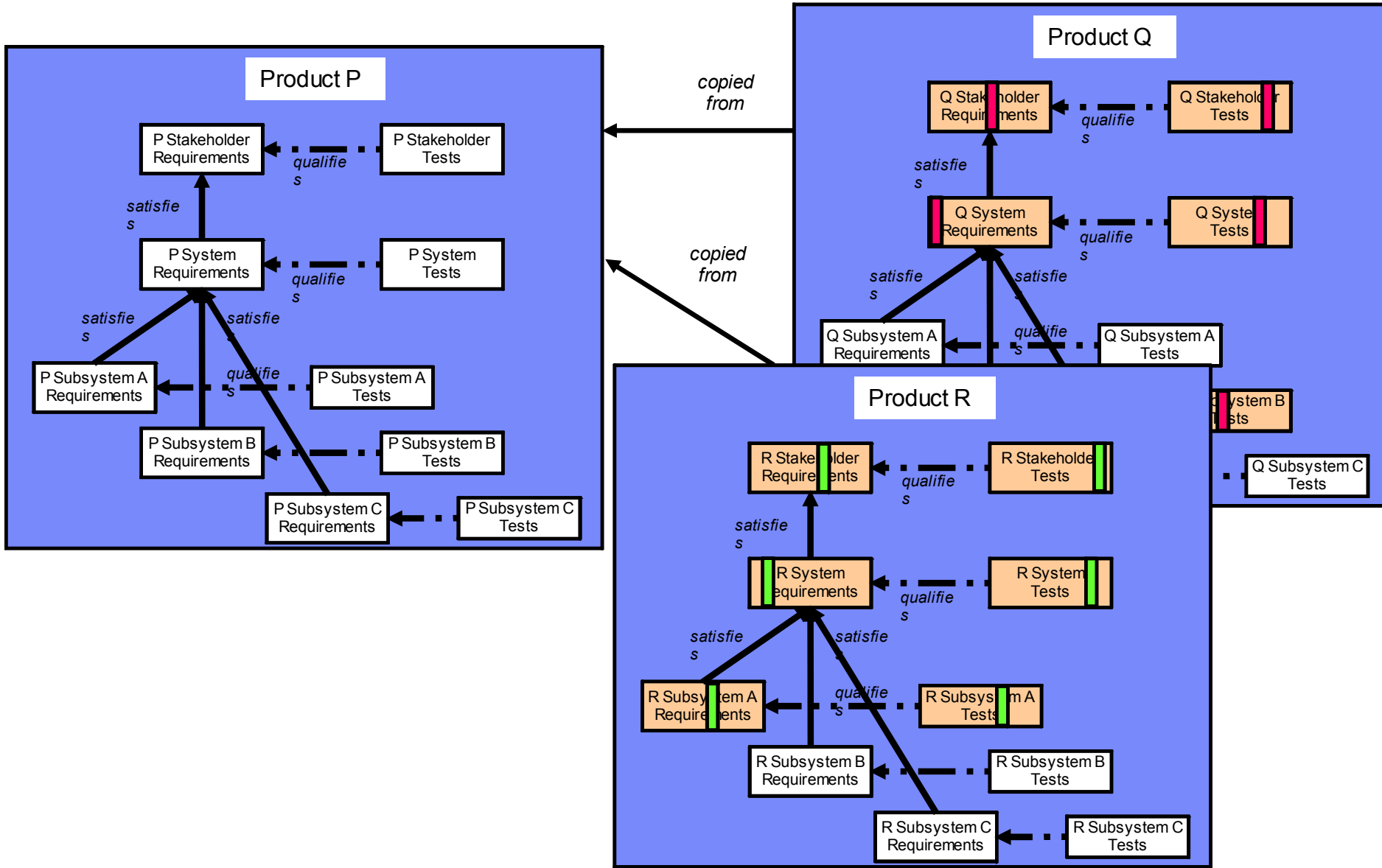
- Current product-centric approaches lead to high complexity
 - ▶ Clone-and-own (and requirements branching)
 - For each new product, make a copy of requirements and modify
 - Leads to expensive duplication, divergence and merging
 - ▶ Attributes and scripting
 - Tag each requirement with one or more attributes about product diversity
 - Leads to high overhead
 - Major effort to define and implement attributes, dictionaries, semantics, schemas, scripts and filters
 - Labor intensive to revisit all requirements and attributes during maintenance and portfolio extension
 - ▶ One-size-fits-all
 - Write the portfolio variations and diversity directly into the requirements text
 - Leads to complexity and errors interpreting requirements for any particular product



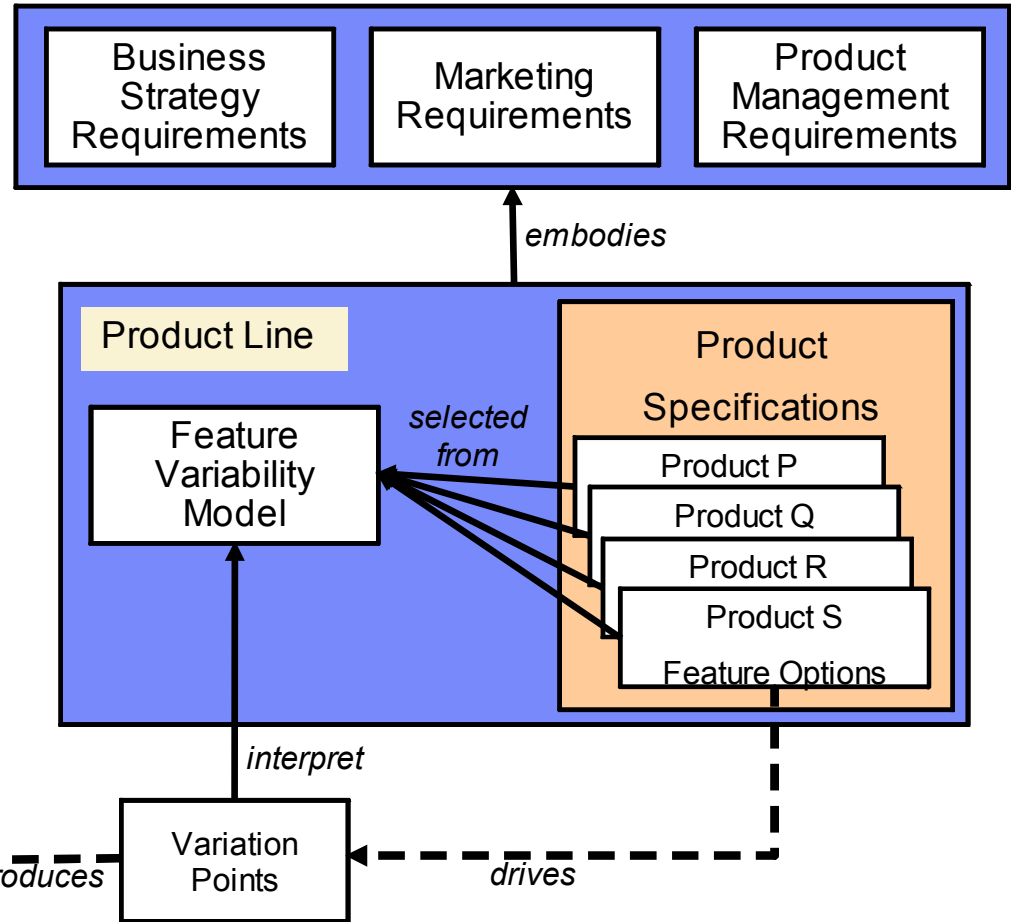
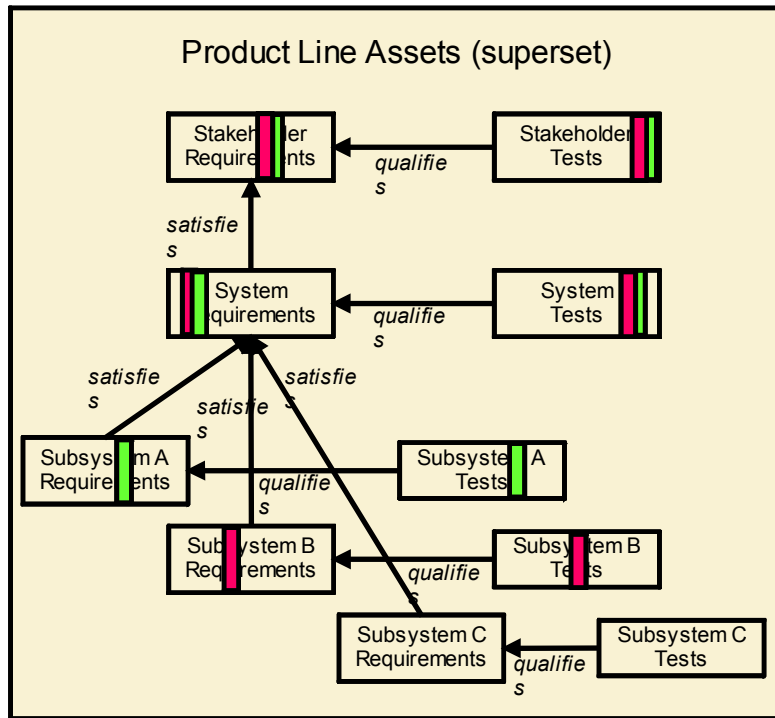
Conventional Single Product Information Model



Clone-and-own approach for Deriving a New Product Specs



Viewing Product S Spec



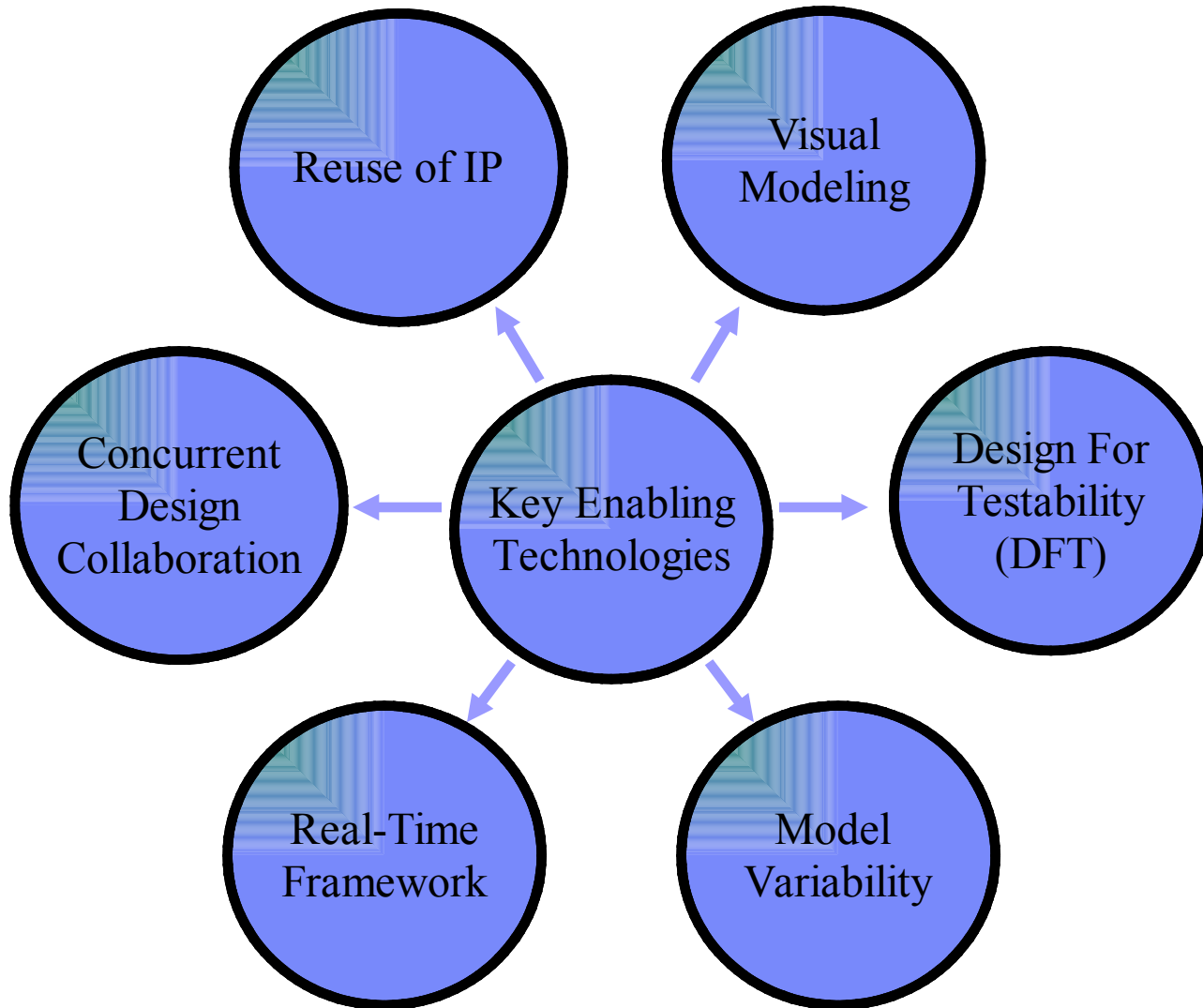
Software Customers' Evolution to Product Lines

Model Driven Systems Engineering

- Basic software reuse
 - ▶ Make software engineer's job easier: Why rewrite when can reuse?
 - Works well within the same product or with multiple products that have similar and/or repetitive needs
 - Inheritance & polymorphism
 - Clone and own
 - Software libraries
- Common services
 - ▶ Widen the scope of reuse, provide basic services
 - Works well when have same service needs by multiple clients/products
 - Event mechanism, Alarm & Logging services, etc.
- Design patterns
 - ▶ Provide reusable solution to a commonly occurring problem
 - Higher-level reuse
 - Proven and tested design solutions, including architecture and domain-specific
- Software frameworks
 - ▶ Give up a measure of control to gain **ultimate** reuse and commonality
 - Hollywood Principle - "don't call us, we'll call you"
 - Trade-off between the generalization and specialization



Rhapsody: Key Enabling Technologies for PLE

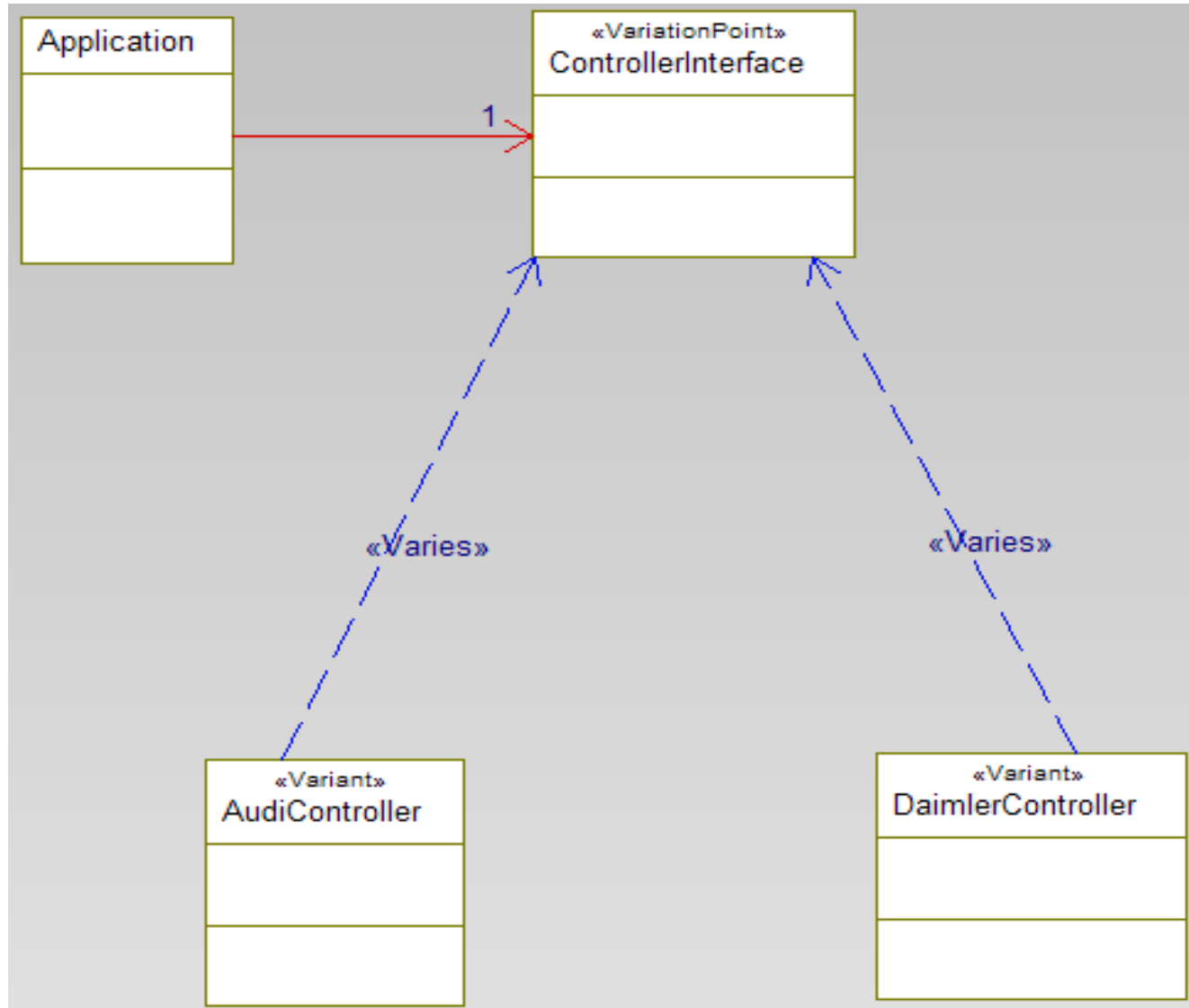


Rhapsody supports component variation

- Users need to produce different variants of the same component for different products in a product line.
- Variants visible and documented on model level
 - ▶ variants not hidden with `#ifdef`'s down at code level)
- Variant independent of Architectural and Instantiation Modelling
 - ▶ Since the architecture is identical – only some parts are variable
- Variant Selection done on Component level
- User marks model elements as variation points
- Variant support is available in Rhapsody C and C++



Variation Management in Rhapsody 7.5



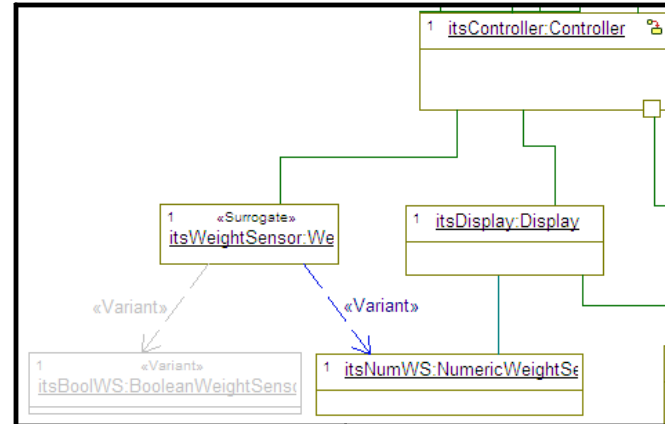
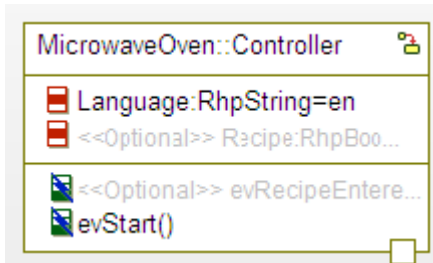
Code generation for Component Variations

- Variation Management is supported directly in Rhapsody 7.5 in C and C++
 - ▶ Can use Inheritance to implement the derived class
 - ▶ The C StaticInheritance is now supported
 - This is **not** classic inheritance. There is no polymorphism.
- The user isn't required to use inheritance to implement the child – It is just an option
 - ▶ If you need very custom code in each derived you do that as well

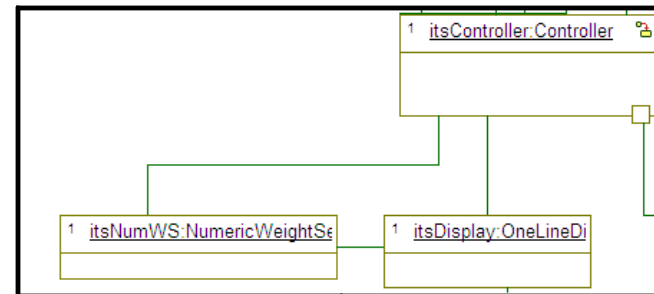
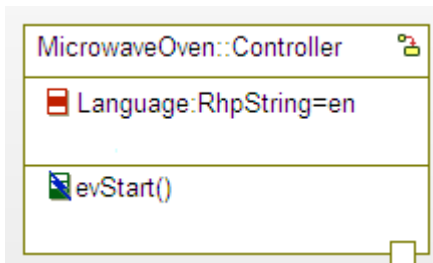


Next Steps: Product View and Product Export

Product View:



Product Export:

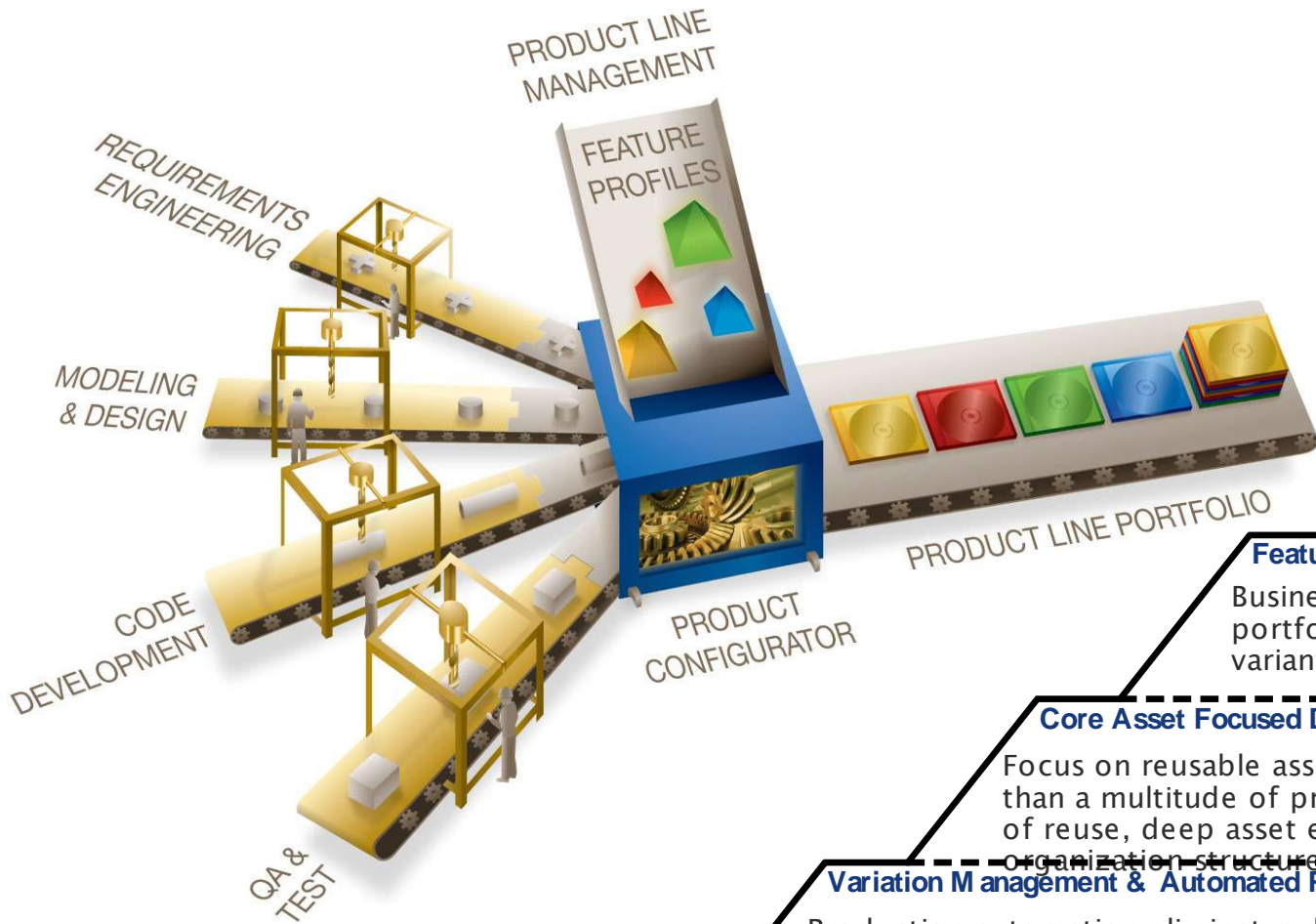


Surrogate elements content was replaced by the variant content



IBM BigLever Product Line Lifecycle Framework

A More Efficient Means of Software Production and Delivery



Feature Based Portfolio Delivery
Business-wide management of portfolio by features and feature variants rather than by products.

Core Asset Focused Development
Focus on reusable assets for single system rather than a multitude of products, leads to high levels of reuse, deep asset expertise and stable organization structure.

Variation Management & Automated Production
Production automation eliminates duplication, divergence, merging, ad hoc variation techniques, lifecycle silos, and manual production.



Common Product Line Reuse Strategy

Lockheed Martin's Maritime Systems and Sensors division have opened up new opportunities for innovation through advances in systems and software product line engineering.



Business challenge:

Needed faster time-to-market for new products, greater productivity and optimized product quality

Solution:

The company chose to integrate the BigLever's Gears SPL Framework with IBM® Telelogic DOORS and IBM® Rational ClearCase to create an end-to-end lifecycle approach to systems and software development for the delivery of product diversity at the speed and efficiency level it takes to satisfy today's cost-conscious. The unified approach enables inter-stage traceability and processes to flow cleanly from one lifecycle stage to the next.

Benefits

- Improved quality of deliverables
- Faster time to market
- Enhanced customer satisfaction project requirements are fulfilled and delivered as expected
- Reference: Embedded.com article:
 - ***“Use product line engineering to reduce the total costs required to create, deploy & maintain systems & software”***
 - <http://www.embedded.com/212400014?pgno=1>

“At Lockheed Martin, the timely and cost-effective delivery of the latest technological advances to our customers is mission critical. Our goal is to constantly ‘push the envelope’ with state-of-the-art product development tools and methods.”

Norman Malnak

*VP of Technical Operations for
Marine Systems and Sensors*

Lockheed Martin



Ikerlan

Smarter wind power solutions

What's smart?

- Wind turbine systems that automatically optimize performance based on environmental factors
- Customized product variations that address the needs of a global energy market

Smarter business outcomes

- 90% reduction in development time for each customized wind turbine model
- 25% reduction in cost of development for wind turbine control systems

How Rational enables smarter products

- Model driven development for optimization of wind turbine control systems
- Supporting their innovative use of SysML and UML
- Product line engineering to more efficiently produce software-based product variations



Think Rational

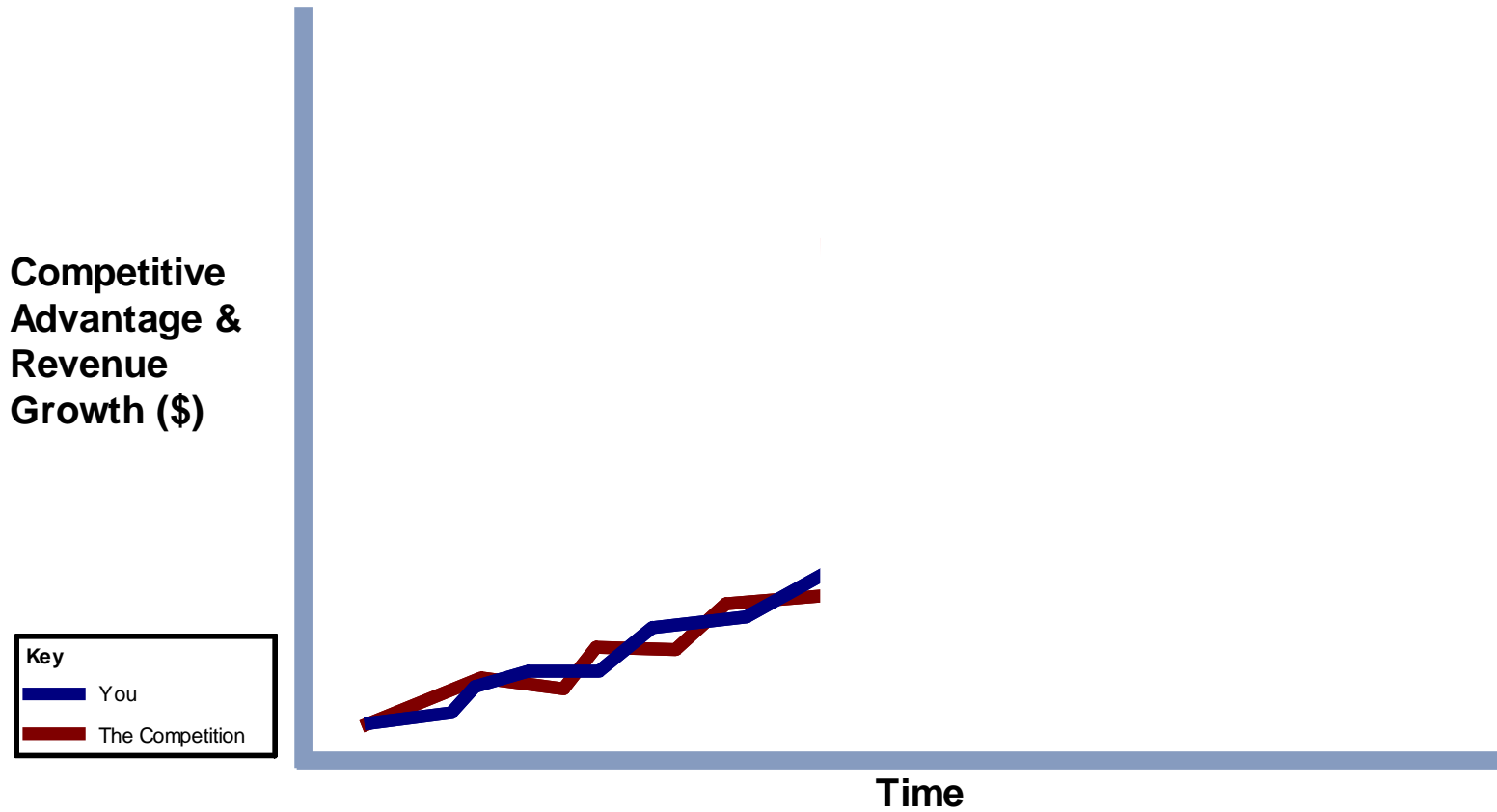
One of many ways Rational enables a smarter planet.

“Our use of Rational Rhapsody for model-driven development, integrated with BigLever Gears for product line engineering, allows us to reuse software assets and manage variations at a pace that lets us keep up with market requirements.”

The advertisement features a large image of a white wind turbine against a blue sky with light clouds. The turbine has "ALSTOM" written on its nacelle. The text is overlaid on the image in a clean, professional layout.

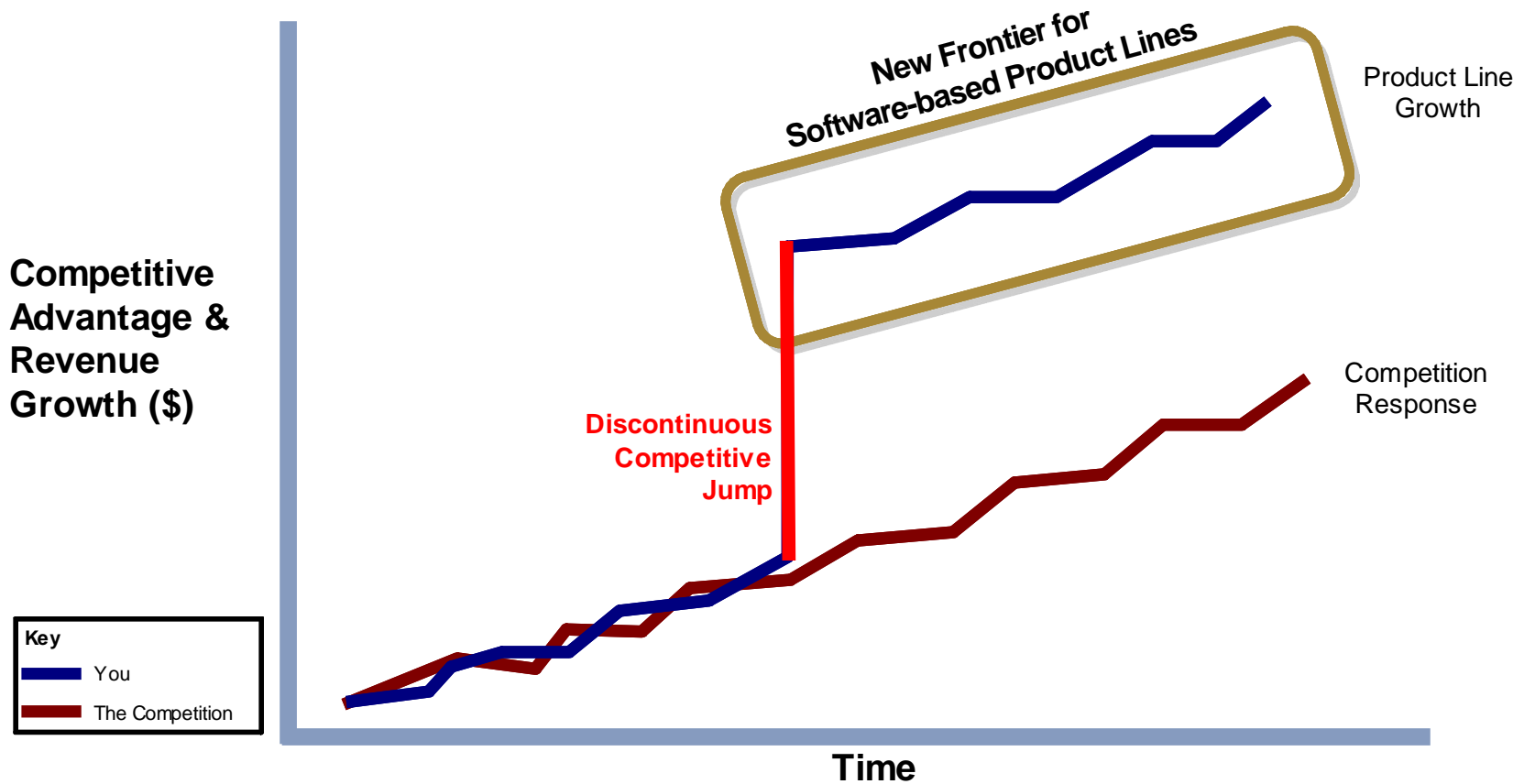


A Discontinuous Jump



IBM BigLever Product Line Lifecycle Framework

The New Frontier for Innovation in Software-based Products and Systems





Learn more at:

- IBM Rational software
- IBM Rational Client Programs
- IBM Rational Software Delivery Platform
- Process and portfolio management
- Change and release management
- Quality management
- Architecture management
- Rational Solutions
- Rational trial downloads
- Leading Innovation Website
- IBM Rational TV
- IBM Rational Business Partners

© Copyright IBM Corporation 2007. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, the on-demand business logo, Rational, the Rational logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.