Compliments of IBM

IBM Limited Edition

Continuous Engineering for Dummies®

A Wiley Brand

Learn:

- What continuous engineering is
- How to continuously improve complex product designs
- How to anticipate and respond to markets and clients
- How to get the most out of your engineering resources

Cathleen Shamieh

Continuous Engineering For Dummies

A Wiley Brand

IBM Limited Edition

by Cathleen Shamieh

For Dummies
A Wiley Brand

Continuous Engineering For Dummies®, IBM Limited Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-118-90440-4 (pbk); ISBN: 978-1-118-90498-5 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Johnson

Development Editor: Brian Underdahl

Acquisitions Editor: Connie Santisteban

Editorial Manager: Rev Mengle

Business Development Representative: Sue Blessing

Custom Publishing Project Specialist: Michael Sullivan

Production Coordinator: Melissa Cossell

Special Help: Steve Shoaf, Barclay Brown, Greg Gorman, Moshe Cohen, and Sky Matthews

Table of Contents

Introduction

Driverless cars that navigate to open parking spaces, oil-seeking drill bits, and self-irrigating corn fields are just a few examples of the rich assortment of capabilities made possible by the smart product revolution. But all that fabulous functionality comes at a price: increased complexity.

Today's smarter products have more electronics and lines of code than ever before, presenting a tremendous challenge to manufacturers, many of whom are just beginning to embrace systems engineering as a way to harness complexity. Adding fuel to the fire is the fact that today's mass market customers are demanding bargain-priced customized products that meet their ever-changing needs. And they want them now.

As a result, manufacturers like you are struggling along several fronts: managing complexity, achieving quality, getting products to market faster, and staying on top of changing customer needs. Continuous engineering provides you with a way to develop complex systems that promises to reduce your costs, satisfy your customers, and — bonus! — make your engineers happier.

About This Book

Continuous Engineering For Dummies, IBM Limited Edition, explains why sequential product development is passé and how continuous engineering can help you manage complexity, improve efficiencies, reuse engineering assets, and incorporate product performance data in order to improve your offerings.

The chapters in this book are designed to help you understand what continuous engineering is and why it's so important to incorporate continuous engineering best practices into your business.

Icons Used In This Book

Throughout this book, you will notice several icons that are designed to alert you to specific types of information.

This icon points out information that you really want to pay close attention to, because it can have a big impact on your business.

Tips alert you to actionable information that can save you time, headaches, or money — possibly all three!

Warnings are suggestions that you should take to heart if you're serious about competing in the evolving Internet of Things (IoT).

Although most of this book takes a high-level approach to continuous engineering, there are a few places in which I dive a little deeper into technical details. You don't have to study this information because I guarantee it won't be on the test.

Chapter 1

Taming Complexity with Continuous Engineering

In This Chapter

- ▶ Reducing complexity and dealing with change
- ▶ Using collaboration to improve efficiency
- ▶ Looking at the V-model of systems engineering and how it relates
- ▶ Pushing past the traditional product development processes
- ▶ Giving your team the ability to integrate earlier in the development process
- ▶ Modifying designs to create customer variants

As renowned physicist, Stephen Hawking, once said, “Intelligence is the ability to adapt to change.” Enterprises need a way to make change part of the fabric of product development. Continuous engineering is that way. *Continuous engineering* is an enterprise capability that speeds delivery of increasingly sophisticated and connected products by enabling businesses to derive and apply insight, while managing cost, quality, and risk.

In this chapter, you find out exactly what continuous engineering is, how it can help you make changes to the cornerstone of your product development processes, and why it’s important to branch out beyond traditional processes.

Corralling Complexity

Continuous engineering helps you deal with the effects of all the changes within the ecosystem where your product lives. It treats product development as an iterative process, applying techniques from practices such as agile development and lean engineering to embrace change. While continuous engineering retains the necessary rigor and milestones, the methods used are adapted to incorporate change.

Continuous engineering thrives on change, knowledge, and seamless collaboration. At its core, continuous engineering is all about

- **Unlocking engineering knowledge:** Accessing, unlocking and understanding all engineering information, regardless of source, enables you to make the right decisions at the right times — so you can turn insights into business outcomes.
- **Continuous verification:** Verifying requirements and design at all stages of the product life cycle helps prevent rework and achieve faster time to quality. Learning is done with real or virtual running systems, so you need to orient the process to get something executable as early in the process as possible and to be building executable systems during all phases of the life cycle.
- **Strategic reuse:** Reusing components across the engineering life cycle help you increase design efficiencies, engineer product lines, and tame complexity.

Improving Efficiency through Collaboration

Do you think your mechanical, electrical, and software engineers really enjoy working in those traditional “silos of development?” Have you ever heard them grumble about how they spend roughly 30 percent of their time looking for information when they’d much rather be building really cool things? Or how they don’t look forward to change because it causes expensive disruption?

If you ask your engineers what they really want, they’d probably tell you that they want to be able to work and think seamlessly, without stopping to figure out who has the information they need, what the latest requirements are, or how another part of the design interfaces with the system they are designing. They want easy access to all engineering data and artifacts across the development effort so they can do the job you’re paying them to do without interruption (well, except for the occasional coffee break). After all, fragmented information and spasmodic work flows are the bane of an engineer's existence.

If you have any hope of keeping up with change in the marketplace, you'll grant your engineers their wishes. Because your system may be part of a larger system-of-systems involving development teams from several (or even hundreds) of companies, you need an open ecosystem that integrates tools from a variety of vendors and a linked data architecture that facilitates data sharing. Then, your engineers will be able to reach out easily to anyone for help and to access relevant information from anywhere.

TIP

By taking an open, integrated systems approach to product development, you give your team the ability to

- Continuously have all the engineering information they need at their fingertips
- Continuously see the impact of their engineering decisions on other disciplines
- Continuous leverage and use designs that have already been completed — and are known to work

Adding a Twist to the V-model for Systems Engineering

REMEMBER

Continuous engineering represents a new approach to systems engineering. It retains the overall systems focus, levels of abstraction, and core activities that form the basis of systems engineering but puts a new spin on *how* the activities are conducted. It also adds some fresh ingredients to pull in market and operational knowledge from outside traditional processes and suggests ways to exploit strategic assets, such as engineering data and reusable code — see Figure 1-1.

REMEMBER

In continuous engineering, the "V" no longer represents a sequential series of steps (as did the traditional V-model for systems engineering); instead, it represents activities that are conducted iteratively (and, to the greatest extent possible, in parallel) as needed throughout the product development process, relationships between activities, and linkages among engineering, operational, and market data.

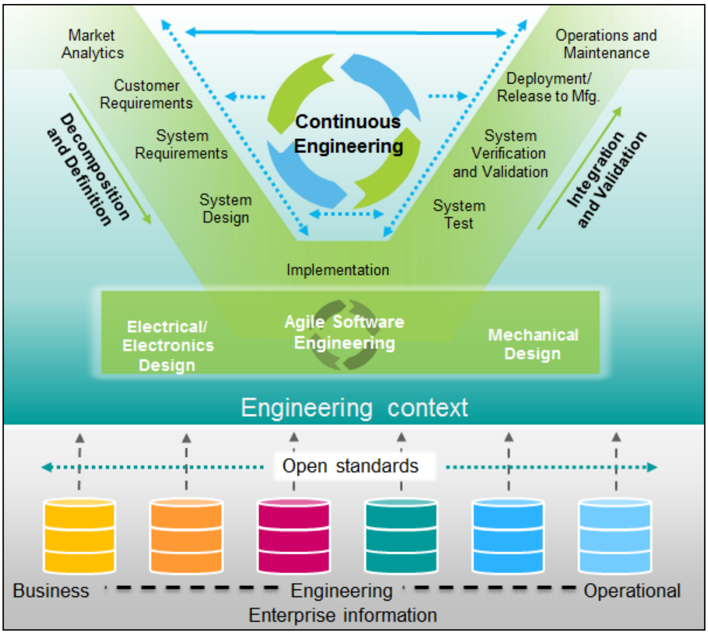

Figure 1-1: Continuous engineering involves iterative execution of key activities.

So, for instance, requirements (on the left side of the "V") are updated as changing or refined user needs are discovered from system verification or new operational data becomes available (both on the right side of the "V"). Updated requirements in turn trigger changes in design, development, and testing. The circle in the middle of the "V" represents the ongoing interactions between left-side-of-"V" activities and right-side-of-"V" activities. This augments the relationships already spelled out by the shape of the "V" itself — requirements are related to design, design is related to development, and so on, with the base of the "V" representing the implementation and embodiment of the requirements. Your focus needs to be on actual running systems (that may be virtual models) so teams can focus on executing system scenarios to manage risk and validation assumptions throughout the project life cycle.

The diagram in Figure 1-1 also illustrates the importance of data relationships in an engineering context. Best practices in continuous engineering include sharing data across engineering disciplines, reusing design elements whenever possible, and incorporating market and operational data into product development activities. Open standards like OSLC (Open Services for Lifecycle Collaboration) help link data and engineering tools to help make continuous engineering a reality.

TIP

The idea is that continuous engineering builds on the foundation of systems engineering practices by persistently applying engineering tools, methods, and techniques to address change and close gaps between current design plans and up-to-minute requirements.

Modeling and simulation

Continuous engineering leverages the modeling and simulation that's part and parcel of systems engineering. You may recall that models are used to represent design entities at different *levels of abstraction,* including system-of-systems, system, subsystem, and component levels. These models capture both the structure (architecture) and dynamic behavior (functionality) of a system, illustrating relationships and interaction between system elements. Models allow you to predict behavior, explore architectural alternatives early in the development process, and perform trade studies to assess which design choices make the most sense.

TIP

To get the biggest bang for your modeling buck, you design your models to be executable so that you can run simulations to test your design against requirements before actually building your system (or even a prototype). In continuous engineering, testing is done iteratively until gaps between design and requirements are closed. Model-based systems engineering (MBSE) and simulation enable you to expedite verification within and across engineering disciplines using open interfaces, such as Functional Mockup Interface (FMI) standards. Rigorous new approaches that leverage FMI allow you to create "virtual products" that you "assemble" from a set of models, each representing a combination of parts from multiple engineering domains, such as electronics, hydraulics, digital systems, and software. "Virtual integration" of multiple models enables you to continuously validate the future behavior of complex cyber-physical systems — a game changer in many respects.

Avoiding defects like the plague

WARNING!

If your design doesn't meet spec, you're introducing technical defects that must be rectified prior to product launch. The later in the process you discover a defect, the more costly it is to fix. If your design doesn't meet user needs, then you are introducing a business defect — failure to achieve intended usage — which is much harder to fix.

Continuous engineering calls for continuous verification and validation throughout the product development process. *Verification* checks to see if the design achieves the defined requirements and complies with standards (in other words, that you are building the system right). *Validation* checks to see if the design meets end-user needs (in other words, that you are building the right system).

Continuous verification reveals defects earlier in the development process, when they're easier and less expensive to remove. Continuous validation goes one step further: It provides early detection of potential business or mission failures, helping you *avoid* defects in the first place.

Continuously optimizing your design

Continuous engineering calls for adjustments to be made when necessary at all levels of maturity of the design. So, if you find gaps between design and requirements during continuous verification testing, you change your design right then and there — you don't wait until you implement — and then you perform verification testing again. Or, if your customer comes to you with a new revelation — "Oh, we just realized that we would prefer to locate the control panel on the *left* side, not the right side . . ." — when you've already built a prototype based on their previously communicated needs, you go back and make the necessary changes to requirements, design, and prototype — and check back with the customer again to make sure your design tracks to current needs.

Analyzing continuously

Analytics is a powerful capability that you can leverage within engineering processes in new ways. For instance, you can analyze requirements to look for anomalies that would be impossible to detect manually, or you can use analytics to search for commonalities among design elements that may enable you to reuse parts of your design within the current system or for another system. By continuously analyzing engineering data and artifacts, you provide insights to your engineering team that facilitate faster learning and informed decision making.

TIP

If you play your cards right, you can use analytics as a strategic tool, or *expert advisor,* to help you find solutions to targeted problems throughout the development life cycle. An expert advisor can answer specific questions you may think up while involved in product development activities such as the following:

- **Requirements:** Are my requirements consistent? Are there any contradictions within my requirements?
- **Design:** Can I predict my system behavior? Can I improve my product design? Can I verify that my design answers my requirements?
- **Implementation:** Will I deliver on time? Should I reduce the scope to meet the next delivery date? If I move people from another project to assist on a late one, what's the impact on both projects?
- **Testing:** Do my test plans validate all possible scenarios? Do I cover all requirements? What are the likely quality risks for this change?
- **Manufacturing:** Can I improve my manufacturing throughput? Can I improve Quality Assurance (QA) metrics?
- **Operation:** Can I reduce warranty cost? How can I reduce maintenance cost? Can I postpone end-of-life? How do I handle updates?

An expert advisor can also give you insights into solutions that require analysis across development activities, including the following:

- What's the impact of a new requirement on my ability to deliver on time?

- Can I deliver an architecture that will achieve performance, manufacturing and operational objectives together?
- How a design decision will impact my ability to deliver the next release on time?
- How can I use current operational data to improve the design of the next version?

Leaving a trace

A key component of continuous engineering is establishing linkages throughout all the systems engineering information and activities so you can quickly identify cause-and-effect relationships and cascade any changes appropriately.

A common approach to *traceability* is the ability to link every requirement to three related items:

- The end-user need that it fulfills
- The system elements that implement or realize it
- The test case that verifies it

Establishing end-to-end traceability throughout the development and testing process enables you to evaluate exactly what is impacted by the latest requirement change or an alternative design choice — before you make the change. It expedites the process of continuous verification and validation and helps ensure that when you make changes, you leave no stone unturned.

Extending Beyond Traditional Processes

If you think that your product development process begins with requirements and ends with release to manufacturing, think again. In a continuous engineering world, you look beyond the traditional product development universe for knowledge you can leverage to help make your systems smarter and more appealing to your end-users.

Marveling at market analytics

Unless you're handed a set of requirements by your customer (often in the form of an RFP), it's up to you to define system specifications. Before you pick up your pen to begin writing requirements, get up and go the extra mile to find out more about what your customers really want — and don't want. By collecting and analyzing social content, you discover customer sentiment, market trends, and even competitor information that you use to help make product decisions and drive requirements. And your marketing team may want to pick your brains so it knows which features to promote based on customer feedback.

Viewing manufacturing as only the beginning

Like a college commencement, the end of the product development process is really a beginning of sorts. Continuous engineering means that you're constantly on the lookout to improve the product, learning by observing how well your product is operating. You gather operational data and use predictive, historical, and real-time analytics to transform the data into performance knowledge. And you don't just put that knowledge into fancy reports for senior management. Instead, you make it an ongoing part of your engineering process, so you can improve on your design and enable predictive maintenance and performance optimization.

Integrating Early and Often

You've heard the expression, "The whole is greater than the sum of the parts." Well, after you create an open, integrated systems environment, you enable your mechanical, electrical, and software design engineers to do much more as a team than they could collectively under a siloed development model.

By facilitating collaboration across engineering disciplines, you give your development team the ability to integrate components of your system much earlier in the development process. In fact, your engineers don't even have to wait for prototype builds — they can integrate system models and test the integration by running simulations. And if defects are revealed or requirements are changed, they can update their models and integrate again.

Early and frequent integration is encouraged in an open, collaborative, continuous engineering environment.

Recycling Parts of Your Design

You patent and test a fuel injection system and release it to the market. You discover some problems in the field, so you go back and fix the problems in the next version. By the third version, your fuel injection system is very reliable, and sales are booming. Now, your OEM approaches you, wanting a variant on your successful design. Do you start from scratch, designing and building an entirely new system? Absolutely not. You take the proven mechanical design and modify it to create the variant your customer wants.

So why would anyone in their right mind (or left mind) start from scratch developing software components if he already had code that performs the same or similar functionality? Continuous engineering encourages the strategic reuse of *all* design elements — including software — to save time, energy, and money in the development process. Just as with the reuse of mechanical components, reusing software components takes a little bit of work and organization to design and manage with reuse in mind, but the rewards are well worth the extra effort. And because you have the traceability between the artifacts, you know what needs to go along for the ride. For example, if you want to reuse a design element, you know what requirements it was used for and what tests were run against it, so you can reuse those as well. Of course, you may have to revisit the requirements or tests in the context of the new variant, but that's a much simpler process.

Chapter 2

Enabling a Connected World

In This Chapter

- ▶ Living in the Internet of Things (IoT)
- ▶ Unlocking the hidden value of your products
- ▶ Embracing change on the path to innovation

Remember when your car was just a vehicle that transported you from one place to another, relying on you to guide it to along the way? Cars today not only get you where you're going safely, but also they alert you to traffic conditions, guide you through lane changes, warn you about obstacles — all while providing real-time updates on fuel prices, the weather, sports scores, and your stocks. Before too long, your car will offer in-dash Internet access, notify other cars about road work and traffic tie-ups, and find, reserve, and navigate to an open parking space near your destination. And not far down the road, your car will drive itself — with or without you in it!

Smarter products are redefining the way you do things and changing the way customers make purchasing decisions. Features like safety and reliability that once differentiated leading car models have become commoditized. Automakers and other manufacturers are shifting gears and seeking out new ways to add value through continuous engineering so they can amaze their customers — and astound their competitors.

In this chapter, you discover what's under the hood of today's smarter products, how the Internet of Things (IoT) is changing the game, and what you need to do to keep up.

Capitalizing on Convergence

Rapid fire advances in electronics, materials science, manufacturing, information technology, and software development have paved the way for you to make inanimate objects do some pretty impressive things. Add some exciting new technological ingredients — social media analytics, mobility, cloud computing, and Big Data analytics — and you can really cook up some earth-shaking new products. See the nearby sidebar "Taking a look at new technologies."

In this section, you take a look at what makes smart products so special and then explore some of the driving forces behind the proliferation of smart products.

Taking a look at new technologies

The high-tech world is always ushering in new and exciting technologies and tools. Here are a few that can help you in your quest to develop smarter products:

- **Social media analytics:** A tool that captures consumer data from online sources and exposes patterns and trends that reflect customer sentiment
- **Mobility:** The technology that enables you to communicate and compute over wireless networks
- **Cloud computing:** Refers to the execution of applications and services on computers connected through a communications network, most often, the Internet, and usually sold as a service
- **Big Data analytics:** Scours huge volumes of data, or data that flies by at high velocities, to quickly help you uncover patterns that you can use to your advantage

Sensing, thinking, and interacting

Despite their differences, smart products all share three common characteristics:

- **Instrumented:** All smart products contain one or more sensors that detect conditions and changes in their surroundings. Light, radiation, motion, heat, humidity, vibration, sound, magnetic fields — you name it, there's a sensor that can detect it. Using their "senses," smart products collect data about what's going on around them. Many smart products also contain actuators, so they can control a system or mechanism that acts on the environment.
- **Intelligent:** Embedded microprocessors give smart products their much-touted brain power. Using the real-time sensory data they collect, along with knowledge bases and user profile information, smart products can make decisions, optimize outputs, adapt to their environment, trigger specific actions, and customize the user experience.
- **Interconnected:** Through Wi-Fi or cellular network connections, smart products share data and decisions with people or with other products. When two or more smart products interact with each other and exchange information, they can deliver even more value than a smart product going solo.

Never-ending networking

Lately, there's been a lot of buzz around the water cooler (well, at least among the geeks) about the impending boom in interconnectedness. To communicate with a device over the Internet, that device needs a unique Internet Protocol (IP) address. With today's Internet (IP version 4, *aka* IPv4), a mere 4.29 billion unique IP addresses exist. Seems like a lot — until you start assigning IP addresses to the likes of washing machines, TVs, fitness tracking bracelets, babies' underwear, and more, in addition to the usual suspects (servers, home computers, smartphones, and their ilk).

Fortunately, some forward-thinking engineers came up with a plan to expand the number of IP addresses so more and more inanimate objects can surf the Internet.

IPv6 (apparently, IPv5 wasn't up to snuff) is the new and improved Internet, offering a whopping 340 trillion trillion trillion (that's 340 followed by 36 zeroes) unique IP addresses. That's enough addresses to go around for a long, long time — at least until next year. And the world has already started migrating to this new protocol. So, now that you have a gazillion IP addresses to play with, more and more products are coming equipped with Internet capabilities. They all want a piece of the action.

The Internet is evolving from a network of computers and servers to a network of all sorts of devices or, using scientific terminology, "things." And it's precisely this vision of the IoT that's causing quite a stir among smart product aficionados because of the wealth of possibilities it creates for smart products and connected applications.

MEMS the word!

One of the most exciting new technologies driving the smart product revolution is micro-electro-mechanical systems (MEMS). These tiny devices combine microelectronic circuits with micromachined sensors and actuators to, in a sense, add eyes and arms to the brains behind smart systems so embedded systems can better sense and control their environments.

MEMS are increasing in popularity as clever companies come up with gobs of interesting places to put them. Automobile crash sensors use MEMS technology to detect sudden changes in force, triggering airbag deployment. MEMS in your car's tires monitor pressure conditions and alert you when you need to add air. Strategically placed MEMS in industrial buildings also sense and control power consumption. Some bioMEMS are implanted in patient sense and control defibrillators and pacemakers while others help optimize drug delivery.

MEMS technology makes it possible to place tiny sensors and microprocessors in virtually any "thing" you want to monitor or control, or from which you want to gather data: scalpels, hearing aids, shipping containers, raw food products, office chairs, parking spaces, street lamps, vending machines, livestock, washing machines — even your family pet.

Making smarter connections

Interconnected, intelligent products are adding value all over the ecosystem:

- **Agriculture:** Smart soil sensors that measure moisture and temperature near the root of plants send messages to control the activation and deactivation of irrigation systems, optimizing water usage.
- **Banking:** Computers in ATM manufacturers' service centers perform remote testing and diagnostics on ATM machines, install software fixes, and upgrade features.
- **Energy exploration:** Smart drilling uses geological microphones to detect acoustic patterns deep underground that are then analyzed to reveal the presence or absence of oil and natural gas.
- **Healthcare:** Implantable MEMS-based glucose sensors measure blood sugar levels and communicate with a micropump for more accurate delivery of insulin.
- **Home automation:** Smart thermostats "learn" homeowners' schedules, optimize energy usage, and facilitate remote program control.
- **Retail:** Sensors on store shelves or in vending machines measure utilization rates and initiate stock orders.
- **Public utilities:** Smart meters monitor energy usage and transmit data to utility companies, which upgrade features remotely. Sensors located in water systems monitor water vibration and issue alerts on detecting leaks, changes in mineral composition, and other conditions.
- **Transportation:** Tiny devices installed in parking spaces and on highways detect the presence or absence of "huge metal objects" (*aka* cars) and communicate real-time status to systems that analyze the data, turn it into actionable information on traffic and parking space availability, and send the results to paying subscribers.

Dreaming big

Sensors, social media, and mobile devices are all producing a ton of different data, called *Big Data,* at a pace that's hard to keep up with. Recognizing that there's value embedded in all this data, experts are developing new ways to harness and analyze the data.

Big Data is characterized by large volumes of data — data that's moving from place to place with high velocity and/or multiple varieties of data that can't easily be managed and processed by using traditional techniques. The term *Big Data* is also used to describe the advanced techniques used to gather, curate, manage, and process the aforementioned Big Data.

What makes Big Data so new, exciting, and different is that it uses nontraditional techniques, such as inductive statistics, to analyze the data in order to reveal nonlinear relationships and predict outcomes and behaviors. What this means to you is that Big Data can examine a boatload of disparate data flying by at high speeds and pick out the important trends, significant relationships, and other hidden gems — giving you new insights and knowledge that you can use to your advantage.

Cloud computing is a way to create what acts like a single virtual server out of many separate servers. The availability of cloud computing is a bonus to fans of Big Data. Cloud computing gives public safety officials a manageable way to process data from traffic and surveillance cameras throughout a city. Cloud computing provides a safety net — resilience in the face of hardware failure. If one server fails, your application continues running seamlessly as the processing load is dynamically moved to other available servers. By using cloud computing, you can process data with no interruptions and deliver reliable service — guaranteed.

Declaring independence

Technology convergence is making possible new categories of products. Unmanned air vehicles (UAVs), networked minicopters, and driverless cars rely on a variety of sensors, advanced control systems, Internet connectivity, and cloud computing to operate autonomously. UAVs, or drones, perform military or security tasks that are deemed too dangerous, dirty, or dull for manned vehicles. Unmanned helicopters are used in civil and industrial applications to dust crops and to inspect dangerous or inaccessible sites, such as active volcanoes, disaster areas, and pipelines. While the acceptance of driverless cars may face some obstacles, autonomous driving capabilities, such as lane keeping or changing, accident avoidance, automatic parallel parking, and driver fatigue detection, are already available in some models today.

Expanding Your Horizons

With the addition of new technologies — sensors, Big Data, cloud computing, and mobility — smart products are now capable of offering bigger and better benefits. For instance, the "smart drilling" technology of the oil and gas industry, which has already improved productivity by an astonishing 200 to 300 percent over the past five years, is poised for yet another revolution — courtesy of new sensor technology, cloud-based supercomputing, and Big Data — which promises to significantly boost energy yield while drastically reducing survey timeframes.

That's all well and good for Big Oil, but what about *your* industry and *your* business? The answer to that question lies squarely in the ecosystem.

Striking it rich with Big Data

Smart drilling uses geological microphones (geophone) to "see through rock" by detecting acoustic patterns beneath the Earth's surface, which are then analyzed and transformed into 3D seismic maps that help oil companies identify where to drill. Steerable drills equipped with sensors and other electronics enable the system to analyze and react to data collected in real time as the drill bores through rock. Smart drilling has resulted in stunning gains in productivity — but there's room for improvement.

Geophone sensors produce hordes of data that are currently processed offline to produce static, episodic 3D seismic maps. But subsurface rocks and fluids are constantly changing, so even the smartest drills sometimes come up empty. All that's about to change with the emergence of more sensitive geophone sensors, higher-speed communications, cloud computing, and Big Data analytics.

So-called microseismic imaging technology promises to "listen in on" underground activity, such as rocks shifting and fluids flowing, in real-time — producing enormous amounts of data that will hold clues about what's happening beneath the surface. The idea is for Big Data analytics to step in and transform this and other sensor data into dynamic geophysical maps that can be used to home in on subsurface oil and gas while safely guiding the drilling system through rock formations.

Smarter drilling is a big test for Big Data and other new technologies, but it has the potential to pay off in spades — and black gold.

The new normal is to view your product as part of an ecosystem, or system of systems, in which many devices collaborate to perform higher order functions. So think of your product as providing a service to other products in the ecosystem. And think of the data that your product generates as a business asset: You can use it to assess product performance, analyze product usage, and — perhaps best of all — you can share it with other products in your ecosystem (for a nominal fee, of course).

Armed with the realization that your product is part of something bigger, you can set your sights higher and unleash your imagination because in the IoT, the possibilities are virtually unlimited.

Unearthing User Needs

Smartphones are becoming so smart that they're replacing other devices and bringing together previously separate functions, making possible new kinds of applications that consumers are excited about. In 2013 alone, nearly 238 million smartphones shipped, bringing the total number of smartphones to 1.4 billion worldwide. And by the end of 2013, 10 percent of cars offered Internet access and an interactive dashboard, with this percentage expected to rise to 90 percent by 2020.

In a world in which two smartphones exist for every nine human beings and connected cars are available to the masses, it's safe to say that consumers have not only embraced the brave, new connected world, but also they're quickly realizing that in the IoT, nearly anything is possible. With thousands of downloadable apps, customizable displays, and capabilities such as remote control of home entertainment systems at their fingertips, it's no wonder consumers have such high expectations for powerful, personalized features delivered quickly and seamlessly.

So the challenge for manufacturers is to discover what sorts of features and functions your end-users really want — even before they themselves know — and to be prepared to adapt products on the fly when users need change. This is where continuous engineering fits into the picture (see Chapter 1 for more information on continuous engineering).

For instance, say you're designing a highly automated driving system for a new car, and one of the features is to automatically switch lanes so the car can pass the slowpokes on the highway. You carefully design this feature so the car checks for the presence of other cars approaching in the passing lane and calculates whether it can overtake the car ahead quickly enough (without exceeding the speed limit, of course) to pass safely. If the car determines that it's safe to pass, should it just initiate the pass? Or should it request authorization from the driver before initiating the pass?

REMEMBER

Challenging design decisions such as "to pass or not to pass (until permission is granted)" are becoming more and more prevalent as systems get more complex and interconnected. For many design decisions, there may not be a clear cut answer; it may depend on user preference. In any case, the engineers designing the system would be wise to find out what end-users really want instead of making assumptions about their needs.

Anticipating Future Functionality

You may be wondering by now how game-changing technology and the explosion of interconnectedness are going to influence your product design effort. Well, unless your product is something as unintelligent, unconnected, and unsensored as a roll of toilet paper, chances are that it will need to interact with other systems and applications. (Even a boring old chair may contain a microprocessor, sensors, and actuators that connect to an advanced comfort control system in your home or office.)

REMEMBER

If you want your product to play in the IoT, you need to project your product's needs into the future. Here are some things you should think about:

- **Connectedness:** What other products should your product talk to today and what might it need to connect to in the future? Keep in mind that new functionality in connected products may drive the need for new or improved features in your product.

- **Data requirements:** What data should your product collect for current and future needs and how will it be used? If you need data from the product's environment, be sure to design in the appropriate sensors to capture that data. If you want to be able to assess your product's performance, plan to capture operational data.
- **Security:** Because your product is connected and is capturing and perhaps sharing data, it's important to bear in mind the potential need for increased security.
- **Future intelligence:** What software functionality does your product need today to succeed in the IoT, and how might that change in the future? This is a tough nut to crack, since you can't accurately predict future needs. What you *can* do is to recognize that software changes are desirable in today's highly software-intensive smarter products because they facilitate innovation. You should also take steps to integrate change into your product development process.

Coping with Complexity

The smarter and more connected a product is the more electronics and embedded software it takes to run it. For instance, the da Vinci S surgical robotic system contains 1.4 million lines of code and over 10,000 individual parts. Many smarter products are *cyber-physical* in nature, meaning that their behavior is dependent on interactions among mechanical, electrical, electronic, and software components. This cross-domain characteristic coupled with increases in components and code poses enormous challenges for product developers.

TIP

If your livelihood depends on the success of one or more interconnected smart products, you need to take a good, hard look at your product development process and assess whether it's equipped to handle the tidal wave of complexity that's about to roll in.

Chapter 3

Getting Smarter about Product Development

In This Chapter

- ▶ Getting your (business) priorities straight
- ▶ Identifying new objectives for developing smarter products
- ▶ Pinpointing promising new technologies
- ▶ Rethinking how systems engineering is carried out

Today's product development processes were devised in the days when hardware was king, software was an afterthought, and the Internet was just a twinkle in Al Gore's eye. Most products had a finite set of features that weren't expected to change much (at least, not until the next product generation). Interacting with other products was the exception, not the rule, and users accepted the fact that fixes and new features didn't happen overnight. Quality was measured solely in terms of the number of errors in the released product.

Well, those days are over. Software leads the charge in today's products and change is a way of life. In this chapter, you get the low-down on the good, the bad, and the ugly in current product development methodologies and discover some exciting new technologies using continuous engineering that can help bring your product development processes up to snuff.

What's Wrong with Today's Product Development?

Today's sequential product development process that's shown in Figure 3-1 is geared for static, stand-alone, hardware-centric products. The process starts with smart people asking, "What can we build that will make money?" and ends with a finished product being released to manufacturing. It assumes that all requirements can be determined in advance and treats changes to requirements as onerous and undesirable.

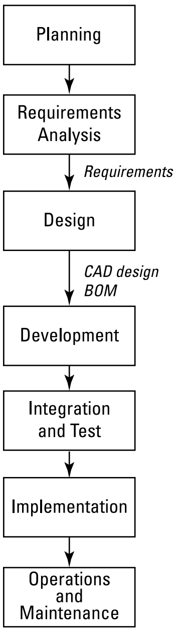

Figure 3-1: Traditional product development is a sequential process.

Static requirements are viewed as outputs of the planning stage and inputs to the design stage, with different, but related, sets of requirements for hardware and software. Hardware and software design and development are done independently, with little interaction between teams. After completion, the hardware design and the associated bill-of-material (BOMs) are handed off to manufacturing, which figures out the fastest and cheapest way to build the product. Integration of hardware and software systems occurs at the end of the development process.

This traditional waterfall product development process seemed to make sense at first by allowing teams of specialists to concentrate on what they excel at. But it's becoming crystal clear that in light of today's smarter, more complex products, you can improve on the status quo in certain areas:

- **Aversion to change:** Sequential product development is overly optimistic about requirements stability. Requirements *do* change because customers and end-users change their minds, competitive products change the market landscape, and new technologies become available. In a waterfall product development process, change is considered highly disruptive to plans and generates a significant amount of rework (which means waste). And yet, change is inevitable, and the rate of change is only increasing, so start rethinking your approach to managing change.
- **Inefficient use of engineering resources:** Silos of development prohibit the sharing of data, information, and expertise. If you develop a new product that shares features with or has similar features to an existing product, you have to start from scratch each time.
- **Internal focus:** Traditional product development often has an internal focus. We need to view the opportunity for gathering more continual information from stakeholders and even from the operational systems themselves as ways to better adapt and improve the engineering process.
- **Misaligned metrics:** Process efficiency measures are just that — metrics about the effectiveness of internal processes. Quality metrics focus on how well a product meets specifications instead of how well it fits its intended purpose. The reported success or failure of a development effort is based on internal metrics — not business outcomes.
- **Late discovery of defects:** Because integration of mechanical, electrical, and software systems occurs at the end of development, any integration problems are revealed very late in the process — and the inevitable integration wars begin. Often difficult to diagnose, integration problems

are costly and time consuming to fix because they aren't discovered until very late in the cycle.

- **Gap between needs and requirements:** The validation process, which checks how well the product meets the customers' or end-users' needs, takes place at the end of the process — after the integration wars are over and the system has been fully tested. All too often, needs have changed, and there's a significant gap between user expectations and product performance.

The bottom line is this: Sequential product development can't keep pace with today's market. It prioritizes internal metrics instead of business outcomes, views change as undesirable, and overlooks opportunities to share and learn from data. It's no longer effective in helping you build the right products.

So Out With the Old and In With the New

There's no question that developing smarter, more interconnected products that cater to a diverse assortment of end-users who want their demands met immediately (if not sooner) calls for a new approach to product development (see Chapter 3 for more information).

Focusing on building the right product

Your product development process should focus on building the *right* product — with the functionality, form, and performance your customers really want and are willing to part with their money for. This means involving customers and end-users throughout the process and shifting your focus from meeting specifications to meeting (or exceeding) customer expectations. Metrics and your definition of quality should be broadened to incorporate business outcomes. After all, what good is a product that's delivered on time, within budget, and with no technical defects if no one wants it?

Welcoming change

Any product development process worth its salt today must welcome change — and that doesn't mean just acknowledging change as you acknowledge someone you see in passing by nodding your head or saying, "Hi." It means embracing change, inviting it into your living room, making it feel comfortable and at home. In other words, you have to build change into the product development process. You do this by staying flexible and not locking down requirements early in the process. You view your product as flexible, adaptable, more like a service that you expect to evolve over time. You treat change as desirable, recognizing that change is necessary for innovation.

Looking beyond your existing processes

One of the smartest things you can do to improve product development is to look beyond your existing processes for knowledge that is available, but not currently being used. Data about what customers want, what customers are saying about your products, how your product is performing, and how your product functions in relation to other products is out there, ripe for the picking! You just need to find it, analyze it, and use it to drive decisions and facilitate faster improvements throughout the development process.

Making better use of engineering resources

It would be well worth your while to figure out a way to make better use of engineering resources so your team can work smarter and get products out the door faster. Small changes to the way you write, structure, and manage code may enable you to reuse or modify code for a future product. Sharing engineering artifacts and tools via an open platform can minimize misunderstandings and streamline collaboration. Best of all, doing away with siloed development efforts can eliminate costly integration problems.

Using Systems Engineering as a Foundation

You probably already know that systems engineering is an interdisciplinary approach for creating large, complex systems that meet a defined set of business and technical requirements. If you think you can use systems engineering to create smarter, interconnected products, you're right — and you're wrong. In this section, you find out that the fundamentals of systems engineering are still valid in the new era of smarter, highly connected products, but the specific approach needs a bit of work.

Born out of the space program, systems engineering is both a practice and a process:

- As a practice, it's concerned with how a system functions and behaves overall, how it interfaces with its users and other systems, how its subsystems interact, and how to unite various engineering disciplines so they work together.
- As a process, it spells out a robust, structured approach to system development that can be applied at a system-of-systems level or within specific engineering disciplines.

The realm of systems engineering is shown in Figure 3-2.

Since the 1990s, systems engineering has been successfully helping companies grapple with increasingly complex products. The systems engineering philosophy of viewing products as one part of an overall system and defining a structured approach to development is just as valid today as it ever was. However, the traditional systems engineering process, characterized by the *V-model* shown in Figure 3-3, is sequential in nature, and you know that won't fly anymore, so you need a new way to implement systems engineering.

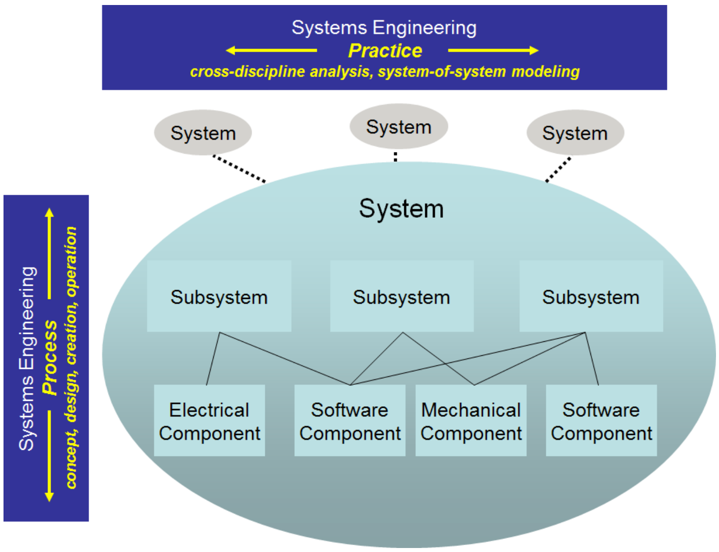

Figure 3-2: Systems engineering is both a practice and a process.

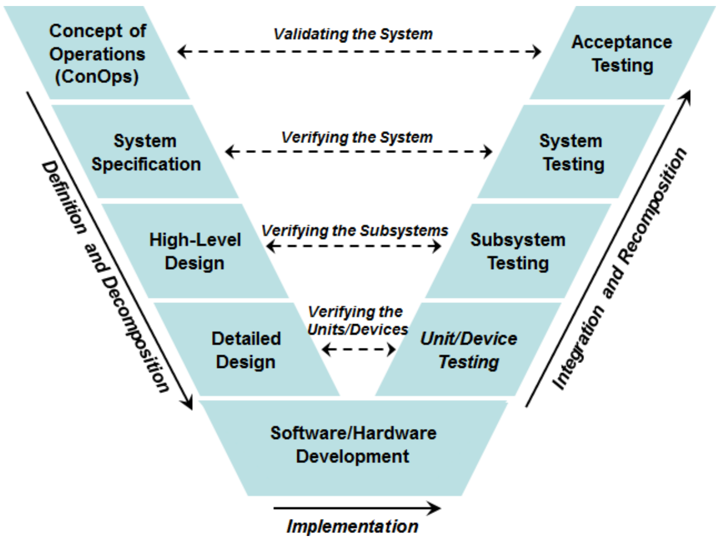

Figure 3-3: The traditional V-model for systems engineering.

Now more than ever, you should keep an overall systems focus not only just on the system you're building but also on the larger system it's a part of and on the system that the larger system is a part of. So, any new approach to systems engineering should build on the following key elements of systems engineering:

- **Levels of abstraction:** This element helps you establish defined scope and boundaries between units of work so those units can be further refined and developed in parallel. System decomposition is a critical technique for improving parallelism in the development process.
- **Modeling:** System models allow you to capture complexity at different levels of abstraction, so you can explore the details of each of these levels and hide or expose details as appropriate and also provides a method for achieving system decomposition and levels of abstraction without having to provide all the implementation details. Modeling languages, such as the Systems Modeling Language (SysML), use a common vernacular to represent system models in order to promote shared understanding.
- **Traceability:** All the steps on the left side of the "V" are linked through requirements, which are referenced as the steps up the right side of the "V" are executed, providing the ability to trace all design elements back to specific requirements.

  A good system engineering process captures traceability between the different artifacts in the life cycle to capture relationships and document decisions so you can better understand cause-and-effect, explore the impact and scope of a change, and provide auditability for compliance reasons.

While systems engineering provides a solid foundation for product development, it has traditionally been carried out as a series of sequential processes, traversing the "V" from left to right, starting with requirements and ending with release to manufacturing. What's needed now is a new approach to systems engineering that welcomes change by leveraging an open tools platform to ensure that all users have easy access to information anywhere in the life cycle at any time. Fortunately, the high-tech industry is awash with new developments that can help turn systems engineering into a powerhouse for smarter product development.

Incorporating New and Exciting Technologies

You may have noticed that over the past few years, experts in computer science and complex systems design have been hard at work inventing new and improved ways of tackling some of life's toughest challenges. And they've come up with some pretty creative ideas that can be put to good use to augment and add value to the systems engineering methodology.

Here's a look at some of the hot new technologies that are causing a stir around the water cooler:

- **Linked life cycle data:** Since 2008, a group of industry leaders has been busy developing ways to make it easier for you to integrate software. Open Services for Lifecycle Collaboration (OSLC) defines specifications that allow your development and operations tools to work together, to share data and other artifacts, and to link to resources.
- **Product line engineering (PLE):** PLE enables you to increase efficiencies by developing a portfolio of products with similar features based on a core set of capabilities. By reusing design components, you eliminate redundant development and testing, avoid rework, and accelerate tailored solutions to market.
- **Analytics:** The more complex and connected a product is, the more data it generates during development and operation. Analytics helps you gain actionable insights into the data, which reveals trends and other critical information that can help you make informed product decisions. Big Data analytics does this on a grand scale, scouring huge volumes of data, or data that flies by at high velocities, to help you uncover patterns quickly that you can use to your advantage.
- **Cognitive computing:** Trained using artificial intelligence (AI) and machine learning techniques, cognitive computing systems attempt to make sense of natural language and data by making predictions and inferences. Such systems build knowledge over time based on experience, and can help your products make better decisions.

Keeping Your Eye on the Prize

In your quest for a smarter way to develop highly connected products, it pays to build on the strong foundation of systems engineering, adapting your approach so that you address the "new normal" of complex system design. Your methodology must encompass the entire development process for your product in its ecosystem, while pulling customer sentiment and operational data into the process and giving engineers and managers the tools and capabilities they need to work together and speed delivery of the right products.

REMEMBER

The challenge for you is to reinvent the sequential product development processes that have contributed to your success thus far by making change a cornerstone of your development effort, looking outside traditional processes for knowledge, and leveraging promising new technologies. Only then will you get positive business results — and isn't that why you're in business?

Chapter 4

Increasing Efficiency through Strategic Reuse

In This Chapter

- ▶ Understanding that good engineering just isn't good enough
- ▶ Thinking in terms of product lines
- ▶ Configuring products from a set of common components

It used to be that you developed a single product for many customers, or an expensive, custom product for one customer. Now, customers want you to customize and personalize practically every product that goes out the factory door. How can you meet customers' needs without watching your profit margin go into negative territory?

This chapter shows you how to maximize reuse of engineering assets across multiple product variants by creating and managing reusable components. This process of continuous engineering can change the way your whole company works.

Assessing the Price of Increased Complexity

According to IEEE Automotive Designline, the average number of lines of software code in Ford vehicles grew from 2.4 million in 2005 to 10 million in 2010. Unless the developers included over 7 million lines of "Hello, world" code, this staggering increase represents Ford's way of satisfying safety and efficiency regulations while meeting consumer demand for performance and convenience.

The average luxury automobile contains more than *50 times* as much code as does a U.S. Air Force Raptor F-22 fighter jet (which contains 1.7 million lines of code). And you can bet that your average Porsche (or even Maserati) doesn't normally break the sound barrier or use stealth technology. Where just a few years ago, cruise control was an option you bragged about at cocktail parties, today's high-end cars sport active suspension and stability control, 360° and distance vision, and much more. Check out Figure 4-1.

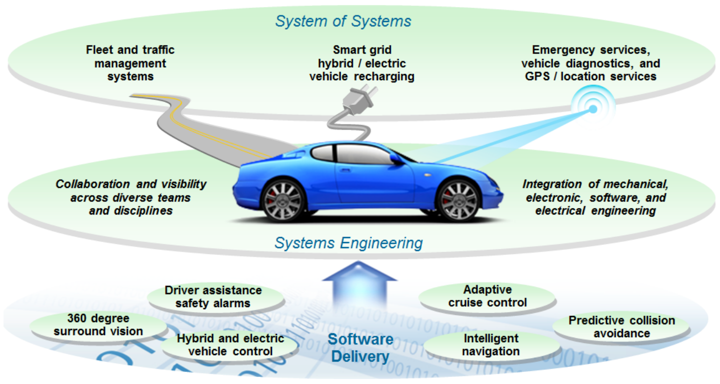
System of Systems
Fleet and traffic management systems
Smart grid hybrid / electric vehicle recharging
Emergency services, vehicle diagnostics, and GPS / location services
Collaboration and visibility across diverse teams and disciplines
Integration of mechanical, electronic, software, and electrical engineering
Systems Engineering
Driver assistance safety alarms
Adaptive cruise control
360 degree surround vision
Predictive collision avoidance
Hybrid and electric vehicle control
Software Delivery
Intelligent navigation

Figure 4-1: A car is a complex system of systems.

Software complexity in modern cars and other smart products has spiked exponentially over the past decade, driving substantial increases in development cost as well as increased risk. It's clear that engineering effort multiplies, but manufacturers are also seeing increases in warranty costs, because, for instance, even one little software bug has been known to trigger a recall of tens of thousands of cars. Adding fuel to the fire is the vulnerability of a smart, connected product's software to hackers, who can create potentially unsafe conditions in addition to their usual mayhem. Protecting smart systems just adds to the complexity and cost today's manufacturers incur.

Embracing New Techniques

Smart manufacturers have been seeking out ways to rein in complexity and improve efficiencies. They've combined forces to deliver more value to customers by connecting multiple products and services into *systems-of-systems*. For instance, an automobile consisting of several subsystems connects with a variety of subscription-based entertainment, communications, and security services.

To get their arms around complex designs, manufacturers have adopted robust, structured systems engineering techniques which accelerate time to market, improve quality, and reduce costs (see Chapter 1 for more about these new techniques). They've also worked hard at developing a core competency in software delivery, recognizing that software is the key to product differentiation.

Multiplying Product Development Efficiencies

In the early days of complex system development, there were two distinct categories of products: mass-produced and custom. Mass-produced products, like the first automobiles, were built to manufacturers' specifications, which were based on what they thought most of their customers wanted. (Recall the paraphrased words of Henry Ford, "You can have it in any color you like as long as it's black.") Custom products, like spacecraft and military systems, were one-offs built for each new customer, and adhered to a long list of exacting specifications supplied by the customer. The development of mass-produced products was very efficient, but the products didn't meet all customers' needs. The development of custom products met the customer's needs but was very time-consuming and expensive.

Today's customers want the best of both worlds. They're demanding increasingly specialized and customized products — but they don't want to pay the piper. To meet their needs, your costs go through the roof — and your profits go out the window. But there's good news. Many products have a high degree of commonality and just minor differences. For instance, the British and American versions of a jet have a lot in common, and newer models of cars are only incrementally different from prior models.

If you can find a way to take advantage of the commonalities between products, you can save yourself a boatload of time, money, and engineer brain power — not to mention, you can get your specialized, customized products out the door faster so you can get to work on the next products on your to-do list.

Even in industries like defense and aerospace, where custom building is necessary, it's important to re-evaluate your engineering processes to see if you can improve efficiencies in order to be more profitable, to offer a better product at a better price, and to win more business.

Making Sure You Don't Reinvent the Wheel

To engineer similar products in a smarter way, you need to take a step back and take a good hard look at your development processes to determine the best ways to exploit commonalities. Then, you need to review your development management processes to see if better ways exist to facilitate your new-and-improved engineering processes.

Reusing engineering artifacts

If the software in a subsystem in Product B performs nearly the same function as the software in subsystem in Product A, you may be tempted to simply copy the code for the Product A subsystem and modify it to meet the requirements of the Product B subsystem. While that sounds like a good idea at first, it really isn't the best way to leverage commonalities.

If you simply copy code from Product A and then modify it for Product B, you own the modified code and are responsible for all the verification testing that goes along with it. You cannot simply assume that the results of the verification testing from Product A for that code apply to the Product B code since you have materially changed the code. Furthermore, if you find a defect in Product A, will you even know that you have to fix Product B, too? Or vice versa? In either case, you end up fixing the same defect twice! Where's the efficiency in that?

A better way to leverage commonalities is to strategically reuse engineering artifacts. If you design the Product A subsystem knowing that there's a Product B subsystem in the wings that requires much of the same functionality, you can create your design in such a way that Product B can *reuse* parts of the design — including requirements, test plans, code modules, and other artifacts — without modification. This way, you just grab the artifacts you need from Product A and plop them right into your design for Product B. No muss, no fuss. And *very* efficient!

Engineering from sets of features

The automobile industry has come a long way since the days when Henry Ford offered a choice of one color. Today, you can go online and build your dream car using your favorite car manufacturer's "configurator." With just a few mouse clicks, you select the type of vehicle (for instance, sedan, minivan, SUV, or truck), model, engine type, color, interior trim, convenience packages (for instance, cold weather package, lighting package, driver assistance package), options, and accessories, and *voilà*! You get the MSRP for your desired configuration and a list of dealers in your area who would be happy to accept cash!

What seems so simple on the surface is actually much more sophisticated behind the scenes. Each selection drives a whole bunch of choices regarding what goes into the car, which subsystems are included, and how they interact with each other. Smart manufacturers will learn from the automobile industry and figure out ways to engineer their products from sets of features so they can readily create dozens, hundreds, or even thousands of product variants.

Managing to improve efficiency

Changing the way you engineer products in order to be able to create multiple variants is just one part of the strategic reuse puzzle. Keeping track of all the product component versions and variants is another critical piece. If you're not careful, you may find yourself with a rat's nest of product versions and variants and no way to tell which ones should be used to configure the new reconnaissance aircraft your favorite government customer has ordered.

TIP

Managing the evolution of your product line in a systematic way is essential if you're going to improve efficiency while offering your customers gobs of product variations.

Shrinking costs while achieving customization

To achieve the right balance between product customization and profitability, you apply the principles of strategic reuse, engineering from sets of features, and smart management of product family evolution. Figure 4-2 gives you a picture of the results you can expect if you manage to pull it all off. For a certain level of customization, you should be able to shrink the development costs down towards the level of mass produced products. In other words, you should be able to an appropriate level of customization and variation to meet customers' needs at a much lower cost than that of traditional custom products.

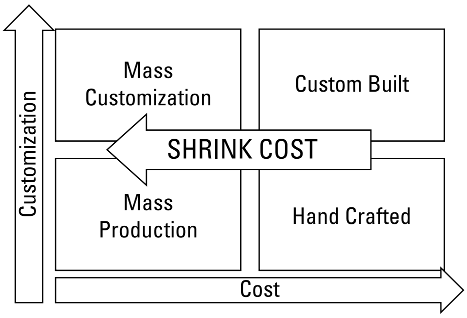

Figure 4-2: Achieving mass customization with lower costs.

In the next three sections, you discover what steps to take to make this happen in your business.

Forging a Trail to Engineering Efficiency

The key to maximizing reuse and managing variation is to develop a product line architecture and methods to express and manage the variants derived from this architecture.

Product lines consist of a set of products or systems that are related through common features but have incremental differences that meet a variety of customer needs. They've been around in the automobile industry for decades, and you may even organize your products into product lines, but that doesn't necessarily mean that they are engineered in the most efficient way possible.

Product Line Engineering (PLE) is a business practice that's concerned with defining, creating, and efficiently managing products that have common features — whether they're part of a formal product line or just a few similar products. Through PLE, you create products and systems that have a high degree of reuse so you can reach more markets, reduce development costs, reduce time to market, and improve engineer productivity.

Creating core assets

At the heart of PLE is the concept of core assets. A *core asset* is a component that's common to multiple products along with the artifacts that define the component in a product development context. Check out Figure 4-3 for an example.

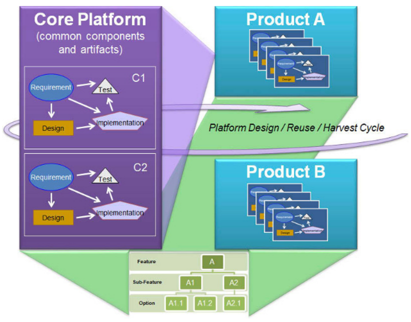

Figure 4-3: Creating product lines with a high degree of reuse.

You create a core asset by going through the entire development life cycle for that particular component. So, for instance, you define the requirements for a traction detector. Then, you design the traction detector, implement your design, and test your design. After you've verified and validated your design, you have a complete core asset that embodies the traction detector used across your automobile product line. You repeat the entire development life cycle for each common component to create a collection of core assets for your product line.

TIP

The benefit of creating a set of core assets is that you define the requirements, create the design, and implement and test the design *once* — and you reuse the asset in many different products. If the requirements change, you make the necessary changes to the design and implementation, test the new design, and redefine your core asset. This saves you the trouble of updating this component of your design in every single product that uses it. (This is strategic reuse at its best.)

Traceability is key for effectively managing core assets because it involves tracking each asset and its associated artifacts throughout the development process. A *where used* report or operation can tell you where a changed asset is being used so those projects can be notified and can incorporate updates at the appropriate time. You can see why traceability and collaboration are so important for strategic reuse.

Designing a core platform

For a very complex product line, you would be wise to design a *core platform,* also called a *common architecture,* consisting of the set of core assets that are common across the product line. You may define the core platform first and build variations on top of it, or, if you've already designed one or more products, you may derive the core platform by harvesting components from those products. After you have your core platform defined, you can derive products by choosing the core assets you need and integrating them according to variation points in the architecture design for the specific product.

Using a feature model

If your product line has many variants or versions (as does, for instance, a line of cars), you create a feature variability model for the platform, such as the feature tree shown at the bottom of Figure 4-3 (see the section "Creating core assets" for more info). You define a hierarchy of features, sub-features, and options, and use this model to drive the selection of core assets. For instance, an automaker may define a feature that is a luxury version of a car with sub-features for a sport version and a family version and options such as different transmissions, trims, and so on. You then map features to individual products, and use the specific set of features to derive the component configuration for the car. So, you can think of the feature model as a map that guides you in the selection of core assets for specific products.

Recognizing Different Entry Points to PLE

You can arrive at the point where you realize that practicing PLE helps you improve efficiencies and better manage complexity in several different ways. They're characterized as follows:

- **Multi-stream/reuse:** You develop a product, and then find yourself developing another product that has similar features to the first product. You decide that it would be a good idea to reuse assets and begin your PLE journey. This is ad-hoc, or opportunistic, reuse, as opposed to planning from the get-go to reuse assets. You take a page from software configuration management in managing variations of artifacts.
- **Parametrics:** You create generic artifacts that apply across your product line, and you define a set of parameters and variation points (possibly quite complex) that govern the selection of artifacts for a particular configuration. By setting a bunch of flags, you trigger the automatic derivation of artifacts for the product you want.
- **Feature management:** You set out from the beginning to define a product line with many different versions and variants. You create feature models to handle complex feature combinations with multiple variation points. Features may be visible to customers (as in the "luxury, sport edition with a driver assistance package" for a customer's configuration of a car), while others (for instance, multiple ways to implement cruise control) are not.

Putting it all Together

In PLE, each product of system in your product line is made up of components, each of which may have multiple versions and variants. For instance, a specific product may consist of a particular version and variant of one component, and a different version and variant of another component, and so on. PLE helps you manage all the different versions and variants of components that are common to products across your product line.

You may be thinking that PLE is somewhat complex to set up, and you're right. However, if you play your cards right, you set up PLE as an overlay on top of your existing product development system and tools (for instance, your requirements management system, test system, and so on), and you save yourself and your development team loads of work when it's time to develop a new product that is an outgrowth of existing products. And when your best customer comes to you with a request for a new product variation, you smile and tell him you'll have it ready in a jiffy!

Chapter 5

Achieving the Right Quality at the Right Time

In This Chapter

- Recognizing that quality is in the eyes of the end-user
- Converging earlier on the right requirements
- Compressing time to customer feedback
- Improving your time to market predictability

You've spent the past year developing a sophisticated, high-performance, feature-rich, consumer electronic device, making sure that it meets all customer requirements and is error-free, only to discover that the end-users find it too complicated to use, so instead, they snatch up your competitor's easy-to-use product (even if it's less functional), and all your carefully laid plans — along with your annual bonus — go in the scrap heap.

Is it possible that a high-quality, technically sound product designed to exacting requirements could be such an abject failure in the marketplace? Absolutely! The high-tech landfills are full of products like this, so beware!

In this chapter, you discover why quality is much more than meeting specifications and what steps you should take in using continuous engineering to ensure your product is exactly what end-users really want.

Redefining Quality

Everyone agrees that quality is a really important attribute of a product that can make or break its success, but if you ask a few people what quality is, you're bound to get several different answers. Traditionally, for manufacturers, quality has been defined as the degree to which a product conforms to specifications. In fact, most companies have Quality Control functions tasked with testing product samples against specs. But for end-users, quality is more about fitness for purpose — *their* purpose — and about how a product compares to competitive products in the marketplace. So which definition is right? The answer is: both.

REMEMBER

Consumers are increasingly intolerant of products that don't work as advertised. At the same time, they want products that serve their needs — even if they can't quite articulate exactly what those needs are.

Failure to deliver quality can often land you in a heap of trouble. The impact of a small mistake in your product is often magnified and distributed throughout an interconnected system. A software bug left to simmer throughout the development process can blow your budget and push out your schedule — that is, *if* you discover the bug before product launch. If you leave it to the end-user to sniff out your software problems, brace yourself for severe consequences in the form of recalls, penalties, loss of business, litigations, and even decreased stock valuation.

Even with a renewed focus on quality over the past few decades, the news is chock full of hard-luck stories about costly mistakes. According to the FDA, in 2013 alone, there were 63 medical device recalls. Government investigators are still trying to get to the bottom of a failed missile-defense test (which cost $240 million), and once they do, you can bet that they will hold the responsible parties financially accountable. According to CNNMoney, software errors are causing problems with the brakes of a leading hybrid car model, sparking massive recalls that could end up costing the manufacturer $2 billion.

WARNING!

If you think that this sort of thing will never happen to you, think again. Add up the number of lines of code your product contains and consider that, for every thousand lines of code, there are typically one to seven defects. And check with your business partners to see if they plan to hold you accountable for defects in your products that can be tied to a recall of one of their products. Typing a few simple keywords in your favorite Internet search engine reveals thousands of stories that help you remain humble. The more interconnected your product is, the greater the risk of defect exposure — and potential business loss.

Measuring Twice, Cutting Once

As the wise business philosopher Peter Drucker once said, "The test of innovation lies not in its novelty, its scientific content, or its cleverness. It lies in success in the marketplace." So, how can you determine what will be successful in the marketplace? The answer to that is this: ask your customers. (And keep asking them.)

Understanding market needs . . . initially

You start by eliciting initial end-user needs. Sometimes, the end-user is your customer, but many times, the end-user is someone else. For instance, your customer may be an airline or a military or civil air force, but the end-user is a pilot or navigator. Your customer may represent the end-user, but your best bet is to hear directly from the horse's mouth.

During the initial needs assessment, you help the customer or end-user flesh out his needs, offering suggestions and possibilities. The result is an initial, yet incomplete, imprecise understanding of the end-user's needs, but that's okay because you're going to work with the user to refine those needs throughout the product development process.

TIP

The initial system requirements for your product are derived from your initial understanding of the end-user's needs, but these requirements will change as you go through the development process.

Think about the process of designing and building a house for a customer. The architect conducts an initial consultation with the customer, and gets a broad understanding of what the customer wants (for instance, a Spanish-style home with four bedrooms, three bathrooms, kitchen, family room, and a portico). From this needs assessment, the architect can draw up initial plans. But a smart architect will refine those plans as the construction project progresses.

Continuously verifying that you're building the system right

The system requirements provide high-level specifications for what the system should do but not how the system should do it. Just as a builder takes architectural plans and decides how to implement them, so your development team takes the initial system requirements and makes decisions about how to satisfy these requirements. Throughout the development process, your team needs to continuously verify that it's building the system right.

REMEMBER

Verification is the process of making sure that the design outputs of a particular development phase meet all the specified requirements of that phase. Software verification confirms that the design is robust, testable, maintainable, and so on — and that the code behaves as expected. System verification confirms that the entire product has been built according to the requirements. Through verification, your team will be able to answer questions, such as

- Does the code contain any potential divide-by-zero or other errors that may cause a crash or other unexpected condition?
- Does the design represent the best architecture for the system?
- Does the code implement the design pattern properly?

TECHNICAL STUFF

Continuous verification means applying verification throughout the whole development process, and re-applying it every time you change something or refine the requirements (more on refining requirements in the next section). Rigorous application of continuous verification enables you to detect and remedy defects earlier — drastically reducing the technical risks and development costs associated with your development process.

Keep in mind that verification is an internal checks-and-balances process. It utilizes tests, simulations, analysis, inspections, and demonstrations to ensure that the design is robust and complies with industry regulations and system requirements. But it doesn't tell you a darn thing about whether your product is *the right product* for your customer.

Continuously validating that you're building the right system

Making sure the system is built to spec is all well and good, but what if it turns out that the spec isn't quite what the customer *really* wants? That's where validation comes in.

REMEMBER

Validation is confirmation that the specifications conform to the users' needs and intended uses. It involves these two things:

- Helping your customers figure out their needs because users can't always conceptualize exactly what it is they need
- Making sure to share the same understanding of what you're going to build for your customers

Here are some examples of questions that you will be able to answer through validation:

- Should a user be able to customize the fields in the display on a runner's watch?
- What's the right behavior of a WiFi-enabled home dialysis system when the connection is intermittent?
- Is this {command1, command2, . . . } the right sequence of voice commands for making a phone call while driving your car?

Validation isn't a new concept in product development. It's common practice to validate a system *vis-à-vis* customer needs — *after* the system has been designed, developed, tested, and integrated. What's new about validation in a continuous engineering context is *when* and *how often* it is done. Continuous engineering challenges you to validate your progress continuously with your customer throughout the entire design — not just at the end.

TECHNICAL STUFF

Continuous validation is an ongoing, iterative discovery process through which you continuously improve your understanding of end-user needs *throughout* the product development process. In doing so, you help end-users articulate and refine their needs, and you update system requirements to reflect those newly discovered needs.

If you hired an architect and builder to create your dream home, would you be content with seeing the house for the first time only after the construction is over? Or would you prefer that the architect and builder check in with you periodically to validate their assumptions and make sure the project is on track to meet (or exceed) your expectations?

People aren't always good at imagining the end product based on requirements. For instance, is a 110 square foot kitchen big enough to comfortably prepare a meal for six people? Who knows until you can walk around in it or at least see it mocked up in a drawing or model.

While you may wish that you could get to a stable, "frozen" set of requirements, that is unrealistic. And if customers' requirements get refined or even changed, wouldn't you prefer to know as early as possible, rather than at the end?

Continuous validation enables you to compress time to end-user feedback, which means that you find out *early on* if there's a gap between the latest end-user needs and the current set of requirements. By closing the loop with the end-user as often as possible, you avoid defects and reduce your business risk.

The V model (covered more in Chapter 1) applies to waterfall just as it applies to iterative processes and even to Agile development around the bottom of the V. But the truth is that in reality, product development goes thru many Vs, as shown in Figure 5-1.

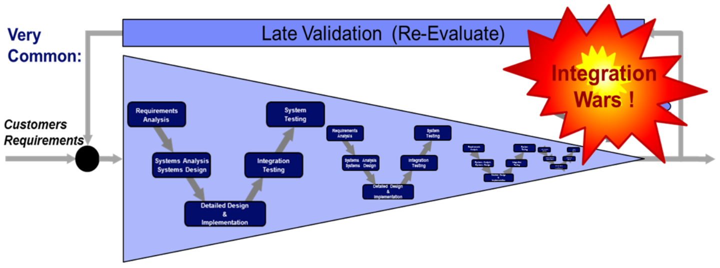

Figure 5-1: The common process is late validation leading to integration wars.

With continuous engineering, the interaction with the customer and/or end-user is much more frequent, which is shown in Figure 5-2.

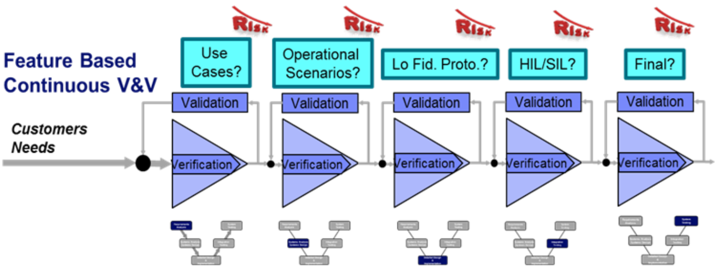

Figure 5-2: Continuous validation with the customer in the loop reduces risk throughout the whole development process.

Making Continuous Quality a Reality

By now you're probably thinking that continuous verification and validation sound great in principle, but putting them into practice while staying within your budget and schedule might be nigh well impossible. As it turns out, continuous verification and validation can save you a lot of time, effort, and money because they reduce surprises, allow for changes (the *right* changes), and detect defects early.

In this section, you take a look at some of the ways you can make continuous verification and validation a reality in your development process.

Collaborating through reviews

As you make your way through the product development process, it's a good idea to stop once in a while, get input from key stakeholders, and make any necessary adjustments to your plans. Short, frequent, constructive reviews are essential for continuous verification and validation.

There are two main types of reviews: internal and external. Internal reviews can be informal meetings with your colleagues on the development team or with management that usually focus on verification. More formal external reviews with customers or end-users are designed to help you validate (or invalidate) your assumptions and fine-tune requirements.

Examples for reviews include an external review with a validation focus where you identify the main use cases, the personas involved, and their main operational scenarios. You may even want to have a formal review. Another example that could be either internal or external, often informal, is a test plan review.

Your objectives in conducting these reviews include the following:

- To solicit early and ongoing feedback from both your colleagues and your customers
- To detect and remedy incorrect assumptions, misunderstandings, and other defects as close as possible to when they are introduced
- To identify potential improvements or risks
- To verify compliance to regulations and conformance to standards

REMEMBER

Reviews should be conducted on a per-feature basis throughout the development life cycle. Ideally, you should focus on end-user features rather than design features.

You'll be happy to know that these short, sweet, social opportunities are roughly twice as efficient at sniffing out and removing defects as formal testing. So, go ahead and make a pot of coffee, buy some donuts, and invite your favorite co-workers and customers to some fun-filled, constructive, and collaborative review sessions.

Exploring end-user operational scenarios

Engineers often make key decisions about how a system should operate without even realizing it, unwittingly believing that all end-users are just like them. Multiply this effect by the number of decision makers on the development team and you can see why many so many high-tech products miss the mark in the marketplace.

To avoid this kind of defect (yes, this *is* a defect), conduct formal reviews with your customers and end-users and create and refine use cases and operational scenarios. This is especially important for connected products where you must think about what operational measures you need to measure and communicate from the device to assist the engineering team in optimizing the design or to enabling operational analytics. Use cases describe all the possible ways the system's functions will be used. Operational scenarios are composed of sequences of one or more system functions and how they relate to different personas that together execute the task specified in a use case.

For instance, say you are designing a driver assistance system for a highly automated car. The use cases for this system may include

- Detecting if another car or person crosses your lane
- Reading dynamic signs on the road and act accordingly
- Automating switching between lanes (passing slower cars)

These are high-level descriptions of what the driver assistance system should do. Take a look at one possible operational scenario for the automating switching between lanes use case:

- Check max speed with Adaptive Cruise Control (ACC) as well as dynamic road signs.
- Check if there is a car fast approaching from behind in the left lane.
- Calculate whether you can overtake the car ahead in less than X seconds without exceeding max speed.
- If ready to pass, ask Driver to acknowledge the pass
    - Unless Driver's preferences are set to "Not Ask."
    - But then should the car alert Driver that it is intentionally passing (so Driver doesn't think car is drifting)? What does this alert look and/or sound like? Do we now need a new or larger display?
- Execute the pass.
- Move back to the slower lane (same scenario).

In reality, the operational scenario is a lot more complex and includes interactions with many subsystems in the car. Even this simple example is enough to show you how important it is to define these scenarios and discuss them with your customer.

TIP What's nice about operational scenarios, especially when capturing the point of view of different personas (such as the car driver, the kid passenger, the mechanic), is that they make your requirements testable and are easily converted into system-level end-to-end test cases representing many different scenarios. Throughout the development process, you use these test cases to verify your designs. You start using them when your design is all or mostly models and continue to use them as you replace models with concrete implementations — all the way to delivery. This requires bit more planning and special attention to your test benches, but it is definitely worth it since you will detect deviations from these end-to-end test cases very early on.

Using virtual models

After you have your operational scenario ducks in a row and you have converted them into end-to-end test cases, you can apply them to your design models to verify and validate your designs, even before you start building or coding your system. Many important design tradeoffs can be validated early on to reduce risk and improve efficiency.

TECHNICAL STUFF *Virtual models* abstract mechanics, electronics, and software entities so that you can create a virtual prototype to test your system before you build a single thing. Using tools such as IBM Rational Rhapsody, Mathworks, Simulink, MatLab, National Instruments LabView, supporting standards such as SysML, and SystemC, you create executable models that enable you to perform early analysis and trade studies of the functionality, behavior, architecture, structure, performance, reliability, and safety of the system very early in the development process.

TIP Many modeling tools offer correct-by-construction synthesis capabilities to automatically generate code based on your virtual models. It's no wonder that model-based software development methodologies report exceptionally low rates of introducing defects and a 96 percent efficiency rate for removing defects.

Ensuring effective and efficient testing

A central quality management hub that acts as a single source of truth for all quality-related artifacts and processing can help streamline continuous verification. Providing an open platform for integrating test plans, test schedules, test cases, requirements links, test component links, and quality dashboards gives your team a birds-eye view of everything related to quality throughout the entire system and enables

- **Collaborative test planning:** Developing Validation Test Plans and Verification Test Plans (yes, separate plans) is a collaborative effort, with input from the project manager, systems engineer, software engineer, test lead engineer, and many other stakeholders, including, at times, your customer.
- **Test execution management:** Manual and automated tests can be centrally managed so it's easy to see and track information such as test schedules, test lab and equipment reservations, who is running the test and why, test results, overall quality status, and more.
- **Integrated defect management:** There's nothing worse than the need to stop a test execution when something goes wrong just to capture the defect. (Imagine testing a train on a track, probably at a high speed, usually in the wee hours of the morning.) Efficient tests capture defects for offline analysis of what went wrong with capabilities for reproducing, resolving, and reporting on defects.
- **Traceability:** Traceability extends across user needs, requirements, test cases, test results, defects, and resolutions. And instead of using proprietary links, it's better to use standards such as OSLC (Open Services for Lifecycle Collaboration).
- **Analytics:** You can't improve what you can't measure. You have to have actionable analytics that will allow you to drive measurable improvements.

Reaping the Rewards of Continuous Quality

Continuous engineering allows you to validate your design with your customer and end-user, eliminating problems that arise due to late validation. It enables you to keep the customer in the loop and conduct communications using artifacts that customers can relate to, such as use cases, operational scenarios, and various levels of prototypes that demonstrate those scenarios.

Don't be alarmed if this process creates more changes to the initial requirements. After all, by compressing the time to user's feedback, you help your user figure out what he *really* wants so there are fewer surprises later on. Early and frequent feedback means that you discover the need for changes sooner — not just when you deliver your system for customer acceptance.

The bottom line is that continuous verification and validation helps you converge on the *right* requirements earlier, so you can deliver the system your user really wants with a lot fewer defects and with better time to market predictability.

Chapter 6

Turning Data into Knowledge

In This Chapter

- Enabling engineers to visualize, organize, and analyze data
- Capitalizing on social media
- Translating operational data into design improvements

The Internet of Things (IoT) generates loads of data that can be very useful — if you can collect it, organize it, and analyze it in the right way. The development of complex connected systems also generates data — engineering data — that's becoming increasingly difficult to keep track of. One of the keys to speeding the delivery of complex, connected products is the ability to gather, extract, analyze, organize, and share data and insights across the team by using the continuous engineering process. This chapter shows you how to conquer complexity and make better use of engineering, market, and operational data.

Conquering Engineering Complexity

Many of today's products and systems, such as cars, mobile phones, aircraft, and medical devices, contain substantial amounts of electronic components and embedded software, enabling them to deliver phenomenal functionality that was unheard of just a few short years ago. But, as the saying goes, "No pain, no gain," and the pain in this case is in the form of escalating product development complexity.

Understanding the rise in complexity

The complexity of today's smarter products comes in two different flavors covered in this section.

Increasing scale

The number of lines of code is increasing as a growing number of features and functionality are built in. Additional hardware and interfaces — which are often sources of error and challenges — add to the complexity.

As customers demand more specialization and customization, manufacturers are creating more product versions and variants, which necessitate more requirements, test cases, models, and other artifacts. And all of these increases drive the need for larger engineering teams.

Increasing cross-domain dependency

System behavior is highly dependent on interactions between different engineering domains, such as mechanical, electrical, electronic, and software, which poses enormous challenges in terms of coordinating activities and integrating data across multiple disciplines. Chances are you've heard your engineers lamenting about how difficult it is to answer questions such as the following:

- Which open work items are related to requirements, tests, or model elements that contain the words *cruise control*?
- A safety standard has changed. Which requirements, tests, design elements, and implementation artifacts are impacted?
- I need to define a new variant for France that reuses parts of the U.S. model. Which artifacts define the U.S. variant?
- How many requirements for the heart monitor are related to tests that failed on their last execution run?
- Are we ready to ship our new UK phone variant?

Questions such as these are often difficult to answer — especially in a timely fashion — and can consume an inordinate amount of your engineers' time when they would be better off (not to mention happier) being creative and productive. Check out the nearby sidebar "Potential impact of complexity" for more information.

Engineering teams need information in context at their fingertips in order to be productive and to make good engineering and business decisions. More often than not, the information they need is housed in several locations, and pulling it all together and making sense of it takes a lot of time and effort and is subject to errors.

Potential impact of complexity

The sidebar table illustrates a hypothetical — but realistic — situation in which, once a year, each engineer on a 100-person development team is tasked with answering a complex question development question such as, "How many requirements for the heart monitor are related to tests that failed on their last execution run?" The cost per engineer per year assumes an annual cost (salary, benefits, overhead) of $150K.

To answer such questions, an engineer performs several tasks. First, he conducts a search that may entail simple text searches across distributed data sources (for instance, requirements, test cases, work items, and design elements) that may contain a certain term, or it may require more complex queries. Next, he pulls together integrated reports and other documents from multiple sources (including data and relationships). Finally, he examines specific artifacts and identifies related artifacts to assess the impact of a change or to verify coverage (for instance, whether a requirement has a related test case).

The likelihood of overlooking something during this complex process isn't insignificant and can lead to costly errors. For instance, say you need to change the properties of several requirements to reduce the response time of various system functions, but you miss one requirement. You design and implement the hardware and software and deliver the system — only to find out that the function responds outside the desired parameters. The result is costly rework — or worse, product recall.

Task	***Time Taken***	***Cost per Engineer per Year****	***Team of 100 Cost per Year***
Search	15%	$22,500	$2.25M
Creating documents	5%	$7500	$0.75M
Impact & coverage analysis	10%	$15,000	$1.5M
Total	30%	$45,000	$4.5M

Creating systems of tools

Inspired by the Internet architecture, a group of system life cycle development tool vendors initiated an effort to facilitate the integration of systems life cycle tools. The result is an open community known as the Open Services for Lifecycle Collaboration (OSLC), which is committed to creating and promoting specifications for integrating data and workflows across conforming independent software and life cycle development tools.

Acceptance of OSLC specifications means that engineers can create relationships between artifacts that are managed in disparate tools. For instance, an engineer can create a relationship between test cases managed in one tool and the corresponding requirements managed in another OSLC-compliant tool. An index of engineering data can be created from federated domain tools, allowing for cross-domain *life cycle queries*. This way, an engineer can execute a simple search for "cruise control" and find all the requirements, tests, and other artifacts related to that feature.

Accelerating product development

After you tear down the walls between life cycle development tools, you give your engineers visibility into data from multiple sources so they can organize information in context and get answers to critical engineering questions with minimal effort. You enable them to make timely technical and business decisions and understand and react to engineering change.

For instance, say you want to understand the impact of replacing a sensor array with a part that's cheaper or has different performance characteristics. Because you can see the big picture across the product development life cycle, you can visualize and analyze the impact of the proposed change and make a swift, well-informed decision.

What's more, because everyone on the development team can access information across the product life cycle, you enable developers to collaborate on engineering processes — regardless of their location or time zone. Your team can even practice "follow the sun" engineering, having some engineers pick up the work where their teammates have left off.

Evaluating engineering processes

As an experienced developer, you have a rich assortment of historical engineering life cycle data in your development system that may contain hidden knowledge about your engineering processes. New "expert advisor" tools and analytic techniques can help reveal information about design change trends, data relationships, requirement conflicts, emerging problem trends, and much more. Such tools use brute force methods to explore many parallel paths through the data — more than you or I ever could — to reveal patterns and discrepancies. Using such tools, you continuously improve your processes and practices based on analytics. See Chapter 1 for more about the new engineering process.

Gaining Insights from Customer Interactions

Wouldn't it be nice to understand what your customers want *before* you build your product? You could conduct research upfront, but that just gives you a snapshot at what some potential customers think. If you really want to know the masses think, you should turn to social media.

Discovering what customers are saying

Many of your customers use social media to connect with friends and businesses and to share personal experiences. Blogs, Twitter, Facebook, and other social media sites are rife with chatter about all sorts of things. Consumers often post product reviews on vendor websites, saying what they like and dislike, and sometimes even mentioning what features and functions they'd really like to have. By collecting social content and analyzing social content, you can get a good understanding of how customers really feel about your products — and your competitor's products.

Calling all customers!

Do you want to know what your customers think of a specific design element, such as changing the layout of the controls on your product? Use Facebook and ask them. Or tweet about it. Focus groups are just one way to get customer input, but you can also use social media to help refine your design.

Understanding customer sentiment

A few years ago, the prospect of analyzing gobs of unstructured data from social media would've been nearly impossible, but today, you can turn to Big Data for help. Big Data is coming of age with advances in cloud computing, and it's geared for analyzing large volumes of data, many different varieties of data, and high velocities of data.

TIP

Big Data tools can help you analyze the unstructured social media data you collect and extract actionable information, such as what customers are interested in and what motivates them. You can also find out how customers are using other products in your ecosystem, what's trending, and much more.

Deriving Knowledge from Operational Data

If you're like most manufacturers, you design and build products, you set aside some funds to finance future warranty claims, and you keep your fingers crossed that you've accurately estimated your warranty expenses. Typical warranty costs are 2 to 3 percent of revenues in many industries, including automotive, semiconductor, appliances, and computers. Manufacturers of consumer-facing brands have reduced warranty accrual rates over the past few years by minimizing inefficiencies in warranty processing, improving business processes, and learning from current warranty data.

Because today's smarter products come equipped with microprocessors, sensors, and connectivity, they're capable of generating gobs of operational and performance data that you can use to your advantage — if you play your cards right.

Extracting value from uncertain data

To turn the data you collect into actionable knowledge, use Big Data and other analytical techniques, including

- **Historical analytics,** which identify patterns in historical data enabling you to learn from past product performance. By analyzing warranty and service records, for example, you can learn about the past performance of your product. This is challenging because those records are usually in a natural language format and aren't easily searchable by simple query strings. Each person who creates a record is likely to describe the same warranty problem using different words — or even misspelled words.
- **Predictive analytics,** which analyze both current and historical performance data and makes predictions about future events. Smart cars are designed to piece together information about the current condition of the car with historical performance and maintenance records in order to make predictions about future service needs.
- **Real-time analytics,** which analyze current performance and operating conditions. By analyzing real-time road and traffic conditions, smart cars can recognize potentially dangerous conditions or situations.

Acting on new knowledge

After you've turned performance and operational data into knowledge, put that knowledge to good use in several ways:

- **Service notifications:** You (or a service provider in your partner network) can use the results of predictive analytics to notify your customers of upcoming service or maintenance needs based on the actual condition of the product they own and operate instead of traditional recommended service intervals. This applies to cars, industrial equipment, appliances, smart drill bits, and all kinds of other products.
- **Real-time alerts and adjustments:** You can inform your customer in real-time of critical conditions, such as a fast-approaching car. You may design your product to

take immediate action, such as slowing down a car or adjusting the speed of a smart drill bit, to adapt to real-time conditions.

- **Continual design improvements:** You can feed the results of historical analyses back into your product development process to enhance your design. For instance, a recurring repair may lead you to create or alter a requirement in order to fix the defect.
- **Continuous software updates:** For long-lived assets, such as cars, aerospace products, and energy exploration systems, you can use the knowledge you've gained to update software to improve performance or add features.

Use all this data to improve your design. Check out Figure 6-1 for an example.

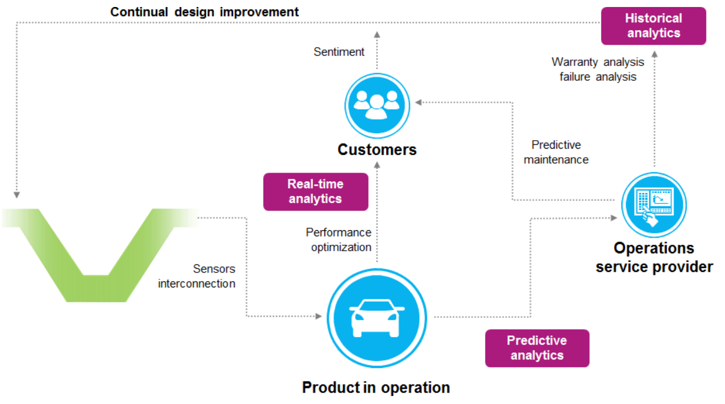

Figure 6-1: Using product performance data to improve your design.

Turning data into actionable knowledge has many business benefits. You can continue to innovate even after product launch and postpone obsolescence with timely product updates. By intervening before product failure, you can extend the useful life of your products and avoid costly recalls. Best of all, you can wow your customers by continuously improving your product and enabling it to adapt to specific conditions — providing your customers with more of an ongoing, personalized service, instead of just a product (and you can charge extra for it!).

Chapter 7

Reaping the Rewards of Continuous Engineering

In This Chapter

- Channeling customer sentiment into your product design
- Mining big data to minimize warranty problems
- Accelerating innovative features to market
- Banking on strategic reuse to save you time and money

Major technology trends, like the Internet of Things (IoT), cloud computing, and Big Data, are poised to drive mammoth changes in the way you live your lives as increasingly smarter, highly connected products enter the mainstream. These technologies have the potential to significantly augment both product performance and product development — for manufacturers who have the right know-how.

Continuous engineering offers you game changing opportunities to capitalize on these new technologies so you can radically improve efficiencies and reduce costs while keeping up with the ever-changing demands of customers who are snatching up smarter products by the dozens.

In this chapter, you look at some examples of companies in the auto industry that have already embraced continuous engineering — and are being richly rewarded.

Turning Customer Comments into Better Cars

If you're an automobile manufacturer, you know how challenging it can be to create new car models that meet all sorts of federal and state standards for safety, emissions, fuel economy, and so on, yet have the right mix of comfort, style, infotainment, and other features to compete in today's marketplace. Consumers often make buying decisions based on an assortment of advanced features — not just how well the car gets them from one place to another — and figuring out what the feature-du-jour is can be quite a challenge.

A Japan-based automaker saw an opportunity to get a leg up on the competition by capturing consumer sentiment about automobile ownership from consumer opinion websites and incorporating it into the design of its cars. It began using an advanced analytics tool to analyze and quantify unstructured customer feedback data from the Internet. Now, the automaker knows what's hot and what's not and can use these insights to fine-tune its designs.

TIP

As an added bonus, the automaker began using positive feedback about its cars — and negative feedback about its competitors' cars — to determine which features of its models to promote in order to increase sales and boost market share. The automaker's investment paid off in the forms of a $1.8 billion increase in revenue and a one percent gain in market share.

Predicting Peculiar Warranty Situations

In 2012, automakers spent anywhere from 1.1 to 4.5 percent of their revenues on warranty claims — and that's just the cost of warranty processing and repairs. Lost sales due to a tarnished image can rack up the pain another notch or two.

A European car manufacturer set out to reduce warranty costs by examining data from vehicle diagnostic read-outs, warranty claims, in-vehicle sensors, and service activities. Its aftersales field data analysis team is tasked with collecting and analyzing this data, which can reach up to 16 gigabytes (GB) per day in both structured and unstructured form. Standard data mining tools just didn't cut the mustard. Analysis typically took several days of computing and sometimes couldn't get the job done at all.

With the help of Big Data and other smart computing techniques, the automaker was able to analyze over 40 terabytes (TB) of vehicle data in a few hours. Successful data mining algorithms and other analyses helped explain and predict unusual warranty conditions. As a result, the automaker reduced warranty costs and improved customer satisfaction.

Speeding New Features to Market with Continuous Verification

The key to building the right product — and building the product right — is effective requirements development and management. You need to have a good handle on end-user needs, define your system requirements around those needs, design, implement, and test with those needs in mind, and continuously update your requirements as needs change. Without a systematic, efficient approach to requirements management, you'd have a heck of a time keeping track of a complex set of requirements, much less adapting to change.

A major automaker was struggling to keep up with increasingly complex requirements, especially with growing (and changing) market demand for in-vehicle infotainment systems. Management was becoming increasingly concerned about the timeframes involved in adding software-based features to vehicle design. Engineers were frustrated that they spent an inordinate amount of time analyzing code to determine the source of software bugs.

The company implemented an enterprise requirements management and modeling solution along with new engineering processes designed to verify and validate requirements early and often. As a result of these efforts, the automaker reduced the amount of time it takes to fully verify software from six to eight weeks down to three days, and decreased bug-cause detection time by 90 percent — from three days to 30 seconds. The solution enabled the automaker to accelerate time-to-market for in-vehicle entertainment systems while freeing up engineer time to create new innovations.

Drastically Reducing Engineering Costs with PLE

One of the big three automakers in the United States manages an incredibly complex product line consisting of 300 hierarchical subsystems, thousands of variant features, millions of product instances, and tens of thousands of unique product variants. The company recently experienced a dramatic increase in product line variation as a result of the emergence of alternative propulsion systems. It faces tremendous challenges in the form of global diversity in legislative regulations and extreme economic and competitive pressure and responds by evolving its product line and feature set annually. The company manages 15 concurrent temporal development streams (and you thought *your* product portfolio was complicated!).

Recognizing that many of the software features in the Engine Control Unit (ECU) were similar across product lines and models, the automaker set out to enable the strategic reuse of code through Product Line Engineering (PLE). It defined a core platform consisting of design components that, with a little extra work, could be transformed into reusable assets. The company then developed an ECU using this PLE strategy — an effort that incurred roughly 8 percent higher engineering costs than developing a non-PLE ECU.

TIP

The results were astounding. The company found that the subsequent applications of a complex PLE-based ECU were created at *less than 25 percent of the cost* of developing a non-PLE ECU. This supports the assertion by industry experts that applying the principles of strategic reuse of engineering assets through PLE across an entire product family has the potential to reduce product development costs by an order of magnitude — or more.

Accelerate innovative products to market and watch your profits soar

Wondering how to manage increasingly complex system designs while keeping fickle customers happy? Relax! *Continuous Engineering For Dummies,* IBM Limited Edition, is your tell-all guide for learning how to successfully launch smart products that mingle seamlessly with other products in today's expanding Internet of Things.

- ***Manage complex connected products** — adopt a continuous engineering approach to systems development*
- ***Turn insight into outcomes** — unlock engineering knowledge to visualize, share, and manipulate engineering assets*
- ***Measure twice, cut once** — continuously verify and validate your design against changing customer needs*
- ***Don't reinvent the wheel** — improve efficiency through strategic reuse of engineering assets*
- ***Turn data into actionable intelligence** — analyze customer sentiment and product performance data to improve your design*

Open the book and find:

- **How to build the right product — and build your product right**
- **Ways to support creative "play"**
- **How to reduce re-invention and speed re-innovation**
- **How to improve cross-discipline decision-making and collaboration**
- **Real-world companies that benefit from continuous engineering**

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

FOR DUMMIES®
A Wiley Brand

ISBN: 978-1-118-90440-4
Part #: RAM14022-USEN-00
Not for resale