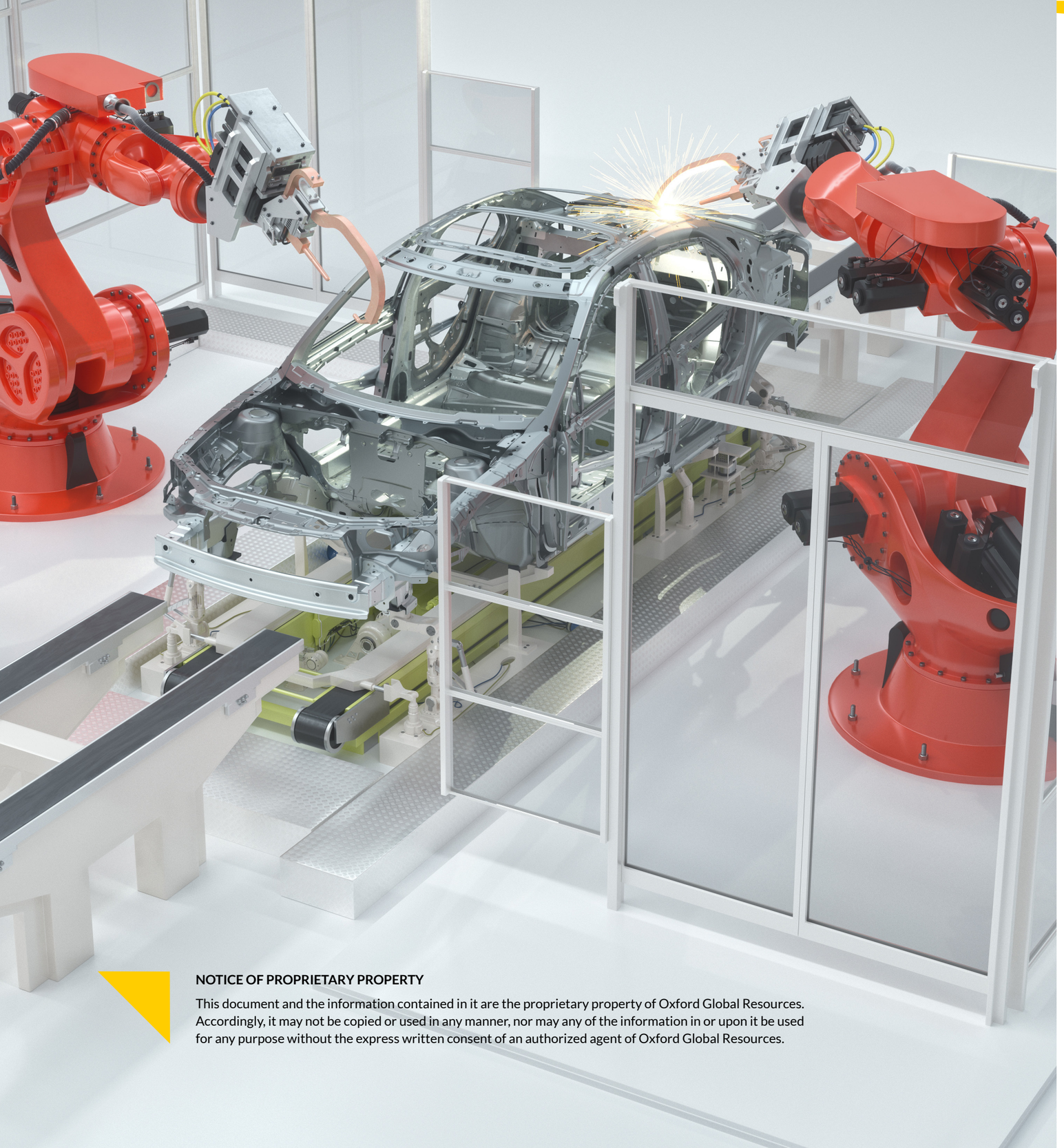
OXFORD™

WHITEPAPER

What Model-Based and Component-Based Software Development Means for Your Organization

Ken Macklem



Contents

Introduction

This paper provides a history of model-based and component-based product development and the benefits of applying these techniques to software development. In this whitepaper, you will learn the various applications of model- and component-based software development, along with the potential challenges and how to overcome them. While many organizations choose different approaches based on their needs, this paper will provide an overview to help you get started and some considerations to keep in mind. Of course, there is no one-size-fits-all method, so if you need support, Oxford can help you with your project.

About the Author

Ken Macklem
Practice Director, Engineering

For over 40 years, Ken's passion has been the development of high-quality microprocessor-based software applications. Most of the software he has developed has been for safety-critical, real-time, embeddved applications for the medical device and automotive markets. His experience covers all aspects of software development, and for the last several years, he has had director-level roles responsible for software and product development at automotive and medical device companies. Ken is strong in software development processes, having developed IEC 62304 compliant processes for three companies, one of which was recently certified to ISO 9001.

Definitions and Acronyms

Definitions

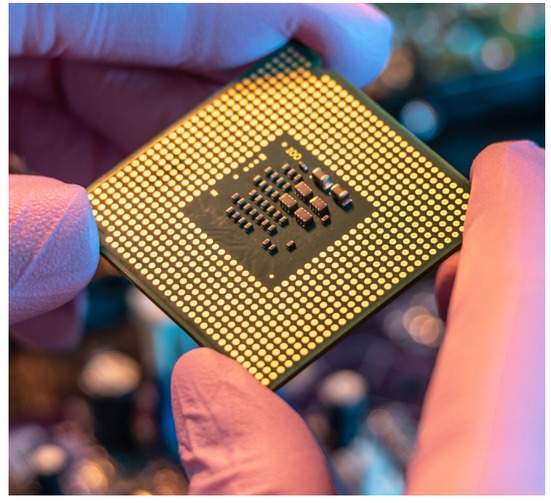
Integrated Circuit:	An electronic circuit formed on a small piece of semiconducting material, performing the same function as a larger circuit made from discrete components.
Microprocessor:	An integrated circuit that contains all the functions of a central processing unit of a computer.
CASE:	The domain of software tools used to design and implement applications.
Model-Based Development:	A software development methodology based on V-cycle, which enables a developer to simulate models for complex control processes.
Component-Based Development:	A procedure that accentuates the design and development of computer-based systems with the help of reusable software components.

Acronyms

CASE: Computer-Aided Software Engineering

CBD: Component-Based Development

MBD: Model-Based Development



SECTION 1

Model-Based Development

History of Model-Based Development

1960

Model-based development began in the aerospace, defense, and automotive industries.

1970

Yourdon introduced structured analysis and design methods for modeling software.

1980

Workstation-based plant modeling tools were developed. Several real-time software analysis and design methods were developed. Case tools supported some of these methods, including simulation. Some of the tools combined plant models and control models.

1990

Plant model-based algorithm development with code generation combined plant and algorithm simulation. Software in the loop simulation combined production code running in tandem with a plant model. Hardware in the loop simulation combined production code running on a hardware simulator.

In the 21st century, plant modeling tools continued to improve as computer hardware improved. We saw the advent of powerful CFD modeling tools on workstations.

After the tech crash, much software development went offshore to countries like India and Asia. Companies decided to outsource labor at a lower wage than what local expert engineers expected to be paid. As a result, software methods and tool development slowed significantly.

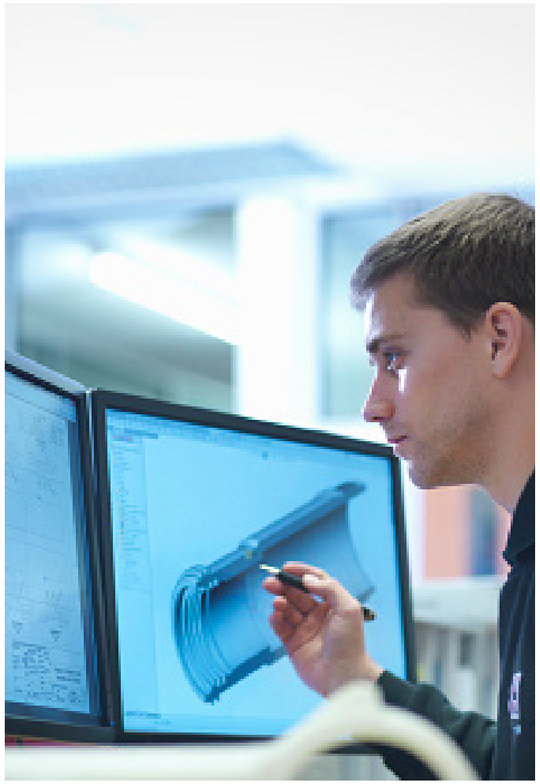
Definition of Model-Based Development

A model is an abstraction of some real entity that describes one or more aspects of that real entity. The model accepts input stimuli and shows the real entity's response to the stimuli. Different types of models of the real entity can be built and combined to provide a detailed representation of the real entity and its operation.

Simulation of the models approximates the behavior of the real entity. The accuracy of the simulation of models of physical entities depends on how well the models are characterized. The characteristics of actual parts and prototypes are measured, and the data is entered into the models. As more parts are characterized, the models produce better results.

The two main uses of model-based development are modeling physical characteristics of the mechanical and electrical components of a product or modeling the dynamic behavior and control algorithms of a product.

Most companies are already using CAD / CAM tools for modeling the physical characteristics of a product.



These models are used to:

- Ensure all combinations of in-spec parts will build correctly
- Analyze stresses and behavior under load
- Design optimizations using Design of Experiments (DOEs)
- Rapidly prototype using additive and subtractive manufacturing techniques

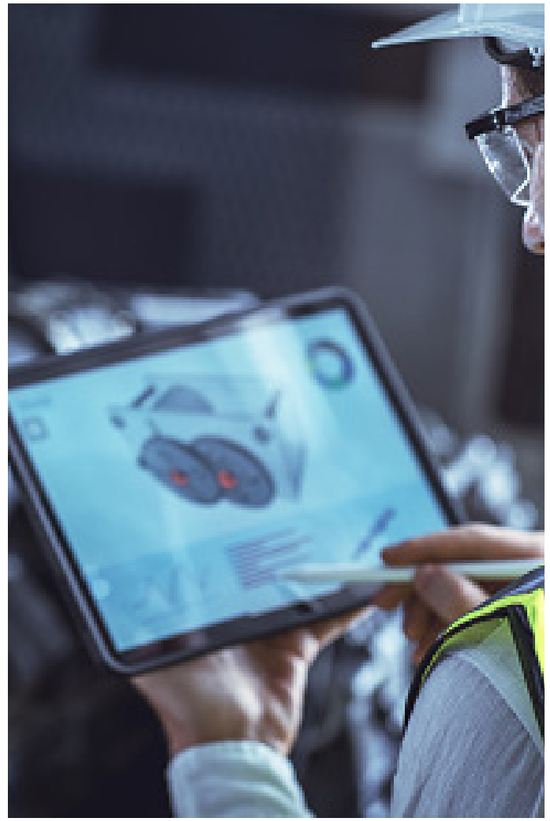
Model-based development creates models of the product to be built and, through simulation, executes them to ensure the design is correct and robust.

Application of Model-Based Development

Models are typically built to:

- Understand the behavior of the physical aspects of the product during operation
- Develop algorithms to control the dynamic behavior of the product
- Develop models of various aspects of the software that runs on the product

Some companies only model the dynamic behavior of the product. Other companies only model various aspects of the software. Control algorithm development is typically done in conjunction with dynamic modeling. The most significant benefit of model-based development is derived when all three types of models are built for the product.



Dynamic Models

Dynamic models use mathematical equations to model the physics of the product. These models are often referred to as plant models. Simulation of these models shows how the product will respond to varying input stimuli over some time. Examples of dynamic models include:

- Automotive vehicle models that determine vehicle performance and fuel economy
- Aircraft models that show how a wing responds to various types of airflow
- Computational fluid dynamic models that show how fluid moves through a flow path
- Models that show how a suspension will behave over different road surfaces

Small models that simulate a part of the product can be combined to form larger models that affect the behavior of significant sub-systems of the product or even the entire product.

These models are often used in DOEs to optimize certain aspects of product performance. For example, these models might be used to find an acceptable trade-off between the fuel economy and the cost of parts of the vehicle.

Control Models

Control models are used to develop control strategies and algorithms to ensure that the product performs the intended task. Control models use dynamic models. The controls engineer develops a control algorithm and then simulates the algorithm on the dynamic model to see how well the control algorithm performs. The controls engineer can iteratively tune the control algorithm to optimize the performance.

Many control development tools will generate code in C, C++, and other computer languages that can be used directly in the product. This can reduce the software development effort and time. However, there are downsides to this approach as well, such as:

- Code can be difficult to understand and debug
- Documentation may not be sufficient for some regulated industries
- The code might be more difficult to unit test
- New code is generated every time a change is made to the model, making regression testing harder to identify
- Differences in the target hardware and development hardware may result in the algorithms not performing the same as the control algorithms simulated on the development hardware

Software Models

Models of software can be developed to:

- Understand software requirements
- Document the architecture and component designs of the software
- Understand and document the desired state or control behavior of the software
- Document the interaction of the software and the users of the product
- Simulate the execution of the software to ensure proper operation of the product
- Provide inputs to automated testing of the system

Combining Models

Models can be co-simulated to understand how the product will perform. First, control models can be co-simulated with the plant models to determine how the product will perform. Later, software models can be co-simulated with the plant models to assess how the control algorithms were implemented. Finally, production software running on target hardware can interact with plant models running on test hardware to test aspects of the software.

Building the Models for your Company

Here are a few steps that are required to build and execute any model, including:

Generally, a tool is used to develop the model. Tools such as Matlab are used to model the plant, while tools like Simulink are used to model control algorithms. Rhapsody or similar programs can be used to model the software. These devices are used to create the models, and then simulation is used for debugging the models. The process continues until the model displays the desired behavior.

Once the models have been created, they need to be validated. For plan models, this generally means measuring the characteristics of production or prototype parts and entering that data into the models. To validate control algorithms, plant models must first be validated.

Once models have been validated, the simulations can be used to tune or optimize various aspects of the product performance.

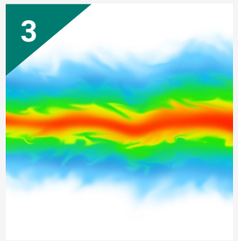
How to Implement Model-Based Development for your Company

There are many ways to implement model-based development in an organization. Some companies start small, focusing on a single type of modeling, and then add more modeling techniques over time. Others combine plant, algorithm, and software modeling all at once.

Examples of Model-Based Development

An automotive Original Equipment Manufacturer (OEM) tasked a development team to create an electronically controlled automatic transmission. They were instructed to reduce the time to a production-ready transmission by 12 months. Model-based development was used to model the plant and control algorithms. The control algorithms were implemented in the software. The team met both the schedule and cost targets. An additional benefit was seen after the development was complete and the transmission was installed in a vehicle. The planned six months of wheels-up testing before calibration was reduced to two days. **The total schedule reduction was 18 months.**

The software for a home hemodialysis machine with 20 operating modes, each with dozens of states, had been under development for over two years by 30 software engineers. The team was making little progress. A CASE tool supporting State Charts and code generation was employed to model the software and generate the code. **A prototype was ready for animal testing in less than six months.**



For two years, during the development of an implantable left ventricular assist pump (LVAD), engineers tried to solve the problem of blood clotting in a particular area of the pump. These clots could lead to a stroke in the patient, a common problem with LVADs. Several prototypes were developed, which cost both time and money. Finally, a company was hired to create a CFD model of the blood flow through the pump. **A solution was found in a matter of weeks.**

Regardless of your approach, the following steps apply:

- Determine the main characteristics of your product
- Determine how you will use the models
- Determine what you will model

To determine the main characteristics of your product, ask the following questions:

- Is there a lot of dynamic behavior that requires closed-loop control?
- Is there much state behavior?

Depending on the systems you use, various tools can be implemented, such as:

- Systems that exhibit mainly dynamic behavior (like vehicle control systems) can use tools like Matlab and Simulink, as they excel at modeling mostly control behavior.
- Systems that exhibit mostly state behavior (like medical devices) can benefit from tools like Rhapsody, which provides excellent real-time modeling systems.
- If the system exhibits a mix of dynamic and other states of behavior, including automatic transmissions and process control systems, companies will often opt for multiple tools to model various types of behavior.

Once you understand the characteristics of your system, determine what you want to get out of the models. For example, do you want modeling to reduce development times and costs and improve quality? How will you achieve these goals—co-simulation of plant and software models, DOE studies to optimize design, generation of code from models, or automated test generation?

Next, determine what you will model, such as:

- A detailed plant model
- A detailed control model
- A detailed software model
- All or some combination

Finally, pick one or more modeling tools that support your decisions.

When making your selection, consider the following:

- What is the cost per seat of your selected tool?
- Does your computer hardware support your selected tool?
- What are the IT requirements needed to support the tools and hardware?
- How many individuals do I need to train? (Plan for three days to four weeks of training per user depending on the complexity of the tool.)

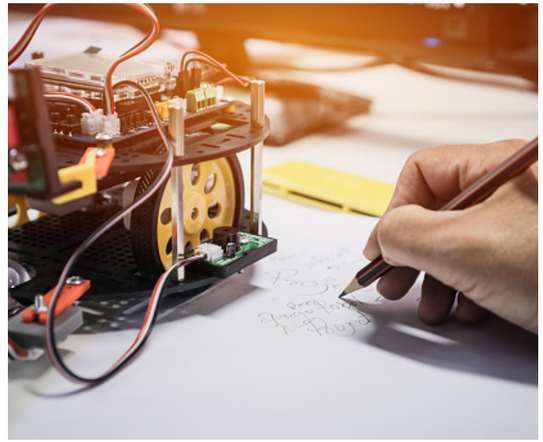
Benefits of Model-Based Development for your Company

If done correctly, there can be several benefits to model-based development.
For example, this type of product modeling can help:

- Find errors in requirements very early in the process
- Find mistakes in design very early in the process
- Reduce the number of prototypes that need to be built
- Reduce the time to develop robust algorithms
- Reduce time to produce production code
- Create and test the test protocols

These benefits can reduce development time and costs and improve quality. However, there is a caveat—the quality of the simulation is only as good as the accuracy and fidelity of the models.

Challenges with Model-Based Development



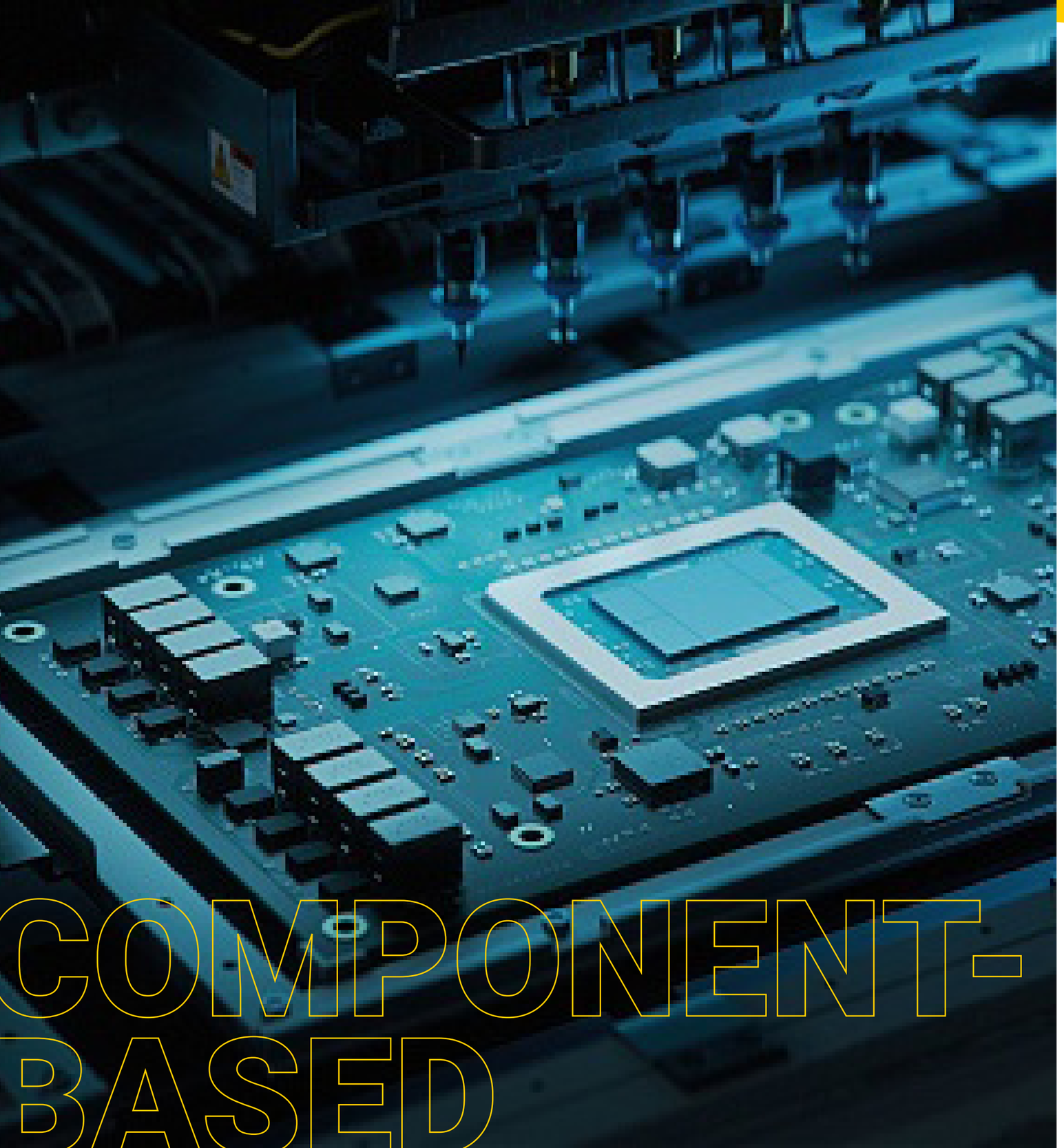
There can be several challenges with employing model-based development.

Many companies don't have the in-house expertise to develop suitable models. A company might make a significant investment in acquiring tools and development of models, only to learn that the models are not helpful. If your organization does not have experience with model development, you can benefit from bringing in an experienced contractor to help.

It takes time to develop a useful set of models. It can be difficult to create complex physics-based models and control algorithms. In addition, complex state behavior can be difficult to model. A Unified Modeling Language (UML) software model for a product is also time-consuming. In addition, the models need to be validated. The result is that it usually takes longer for real hardware and software to be seen. However, if done right, the overall development time and costs of a reliable, quality product are lower.

Good modeling tools are expensive. There can be a significant upfront investment with annual renewals of around 20% of the initial product cost. However, inexpensive tools with limited functionality can be problematic for several reasons, including:

- It takes much longer to develop models with the lower end tools, and the resulting models are not very useful
- These tools tend to have inferior simulation capabilities
- Support tends to be very limited
- Better quality tools cannot use the models should you decide to upgrade
- There is minimal reduction in schedule and budget
- There is very little improvement in quality



SECTION 2

Component-Based Development

History of Component-Based Development

Component-based development began in the early 19th century by introducing interchangeable parts for gun manufacturing.

1959

The invention of the integrated circuit in 1959 enabled component-based development of electronics. Reusable transistor circuits could be implemented in an IC. These reusable circuits could then be combined with other reusable circuits on an IC to create a chip with greater functionality. Today, there are chips with billions of transistors, all built from smaller, reusable circuits.

1968

The concept of component-based software development was introduced

1986

The Objective C language was developed for the express purpose of creating reusable software components. Unfortunately, these early attempts to build reusable software components were not entirely successful.

1990

In the early 1990s, the Design of Experiments (DOE) techniques were developed for design optimization. Models of the product were built. By specifying an objective function and varying parameters of each of the components in the development, an optimal design could be arrived at in a relatively short time.

Today, the development of reusable mechanical and electrical components is commonplace. In addition, many companies are developing reusable components that include reusable software. For example, AUTOSAR, in the automotive industry, allows companies to build hardware with software sub-systems that can be integrated on an OEM platform independent of the target hardware and underlying architecture.

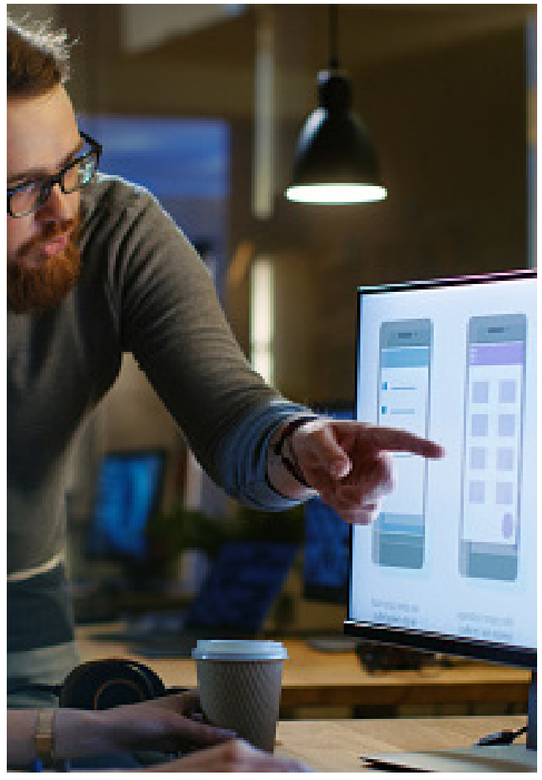
Definition of Component-Based Development

A reusable component is a component that can be used in many different designs with little or no modification. For example, the same electric motor can drive a fan, spin a wheel, open a door, etc.

There are **three types of reusable components**, including:

1. Hardware (mechanical and electrical)
2. Software
3. Hardware with software content

A fundamental reusable component generally performs a single function. For example, a motor produces torque to drive a process. That same motor can be used in several types of applications. As long as the engineer knows the essential characteristics of the motor (dimensions, weight, torque output, etc.) they can integrate the motor into their design. Electronics and software can be combined with the engine that provides closed loop motor control. If the interfaces to the electronics and software are known, this new component can be used in many different applications.



Reusable components are essentially building blocks. These building blocks can be integrated with others to form larger building blocks that perform more complex functions. Finally, these components can be integrated to create fully functioning products and systems.

Implementation of Component-Based Development for your Company

First, identify hardware and software elements common to several of your products. These include things like:

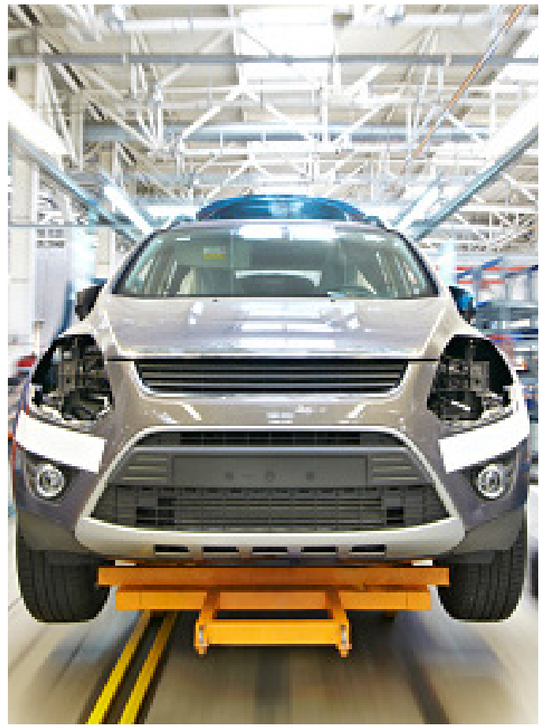
- Motors
- Pumps
- Valves
- Imaging elements
- Graphical User Interface (GUI) elements

Next, develop the following types of hardware and software architectures to support reusable components:

- Standard operating system
- Standard mechanical and electrical interfaces
- Standard hardware and software communications interfaces
- Standard GUI development environment and widgets
- Standard microprocessors and microcontrollers
- Standard sensors and actuators

Finally, identify the parameters of the components that can be adjusted to configure the component for different applications. These parameters are hardware and software calibrations whose values are set for each application.

Model-based development tools can help develop and manage reusable components. In addition, simulation can be beneficial when integrating components for a new design. Lastly, DOEs can be very useful for optimizing a new design.

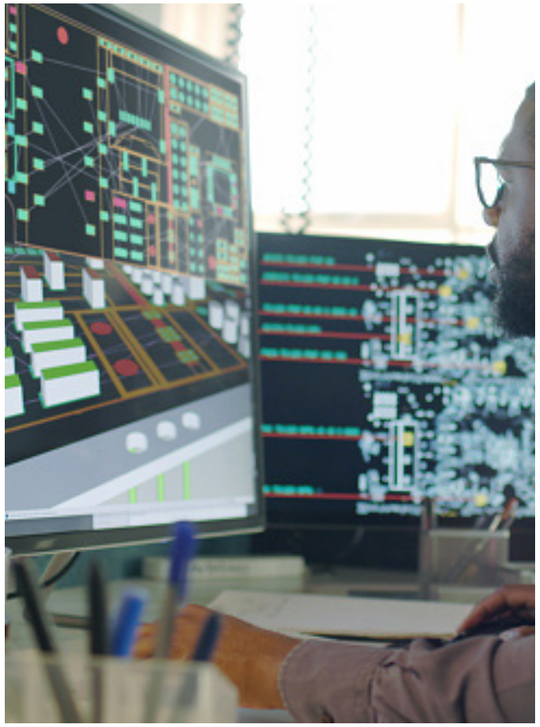


Benefits of Component-Based Design for your Company

The main benefits of component-based design are:

The initial development of a reusable component tends to take longer because the designer needs to anticipate how it will be used in other designs and interfaces to accommodate those applications. However, development times can be significantly reduced when that component is reused in a subsequent design.

The component is tested once for the initial design. If the component is reused without modification in a subsequent method, the testing for that component is significantly reduced. This is especially true in highly regulated industries like the medical device industry, where software testing effort accounts for two-thirds of the total software development effort.



When a component is reused, it is configured rather than developed. Configuration of a component takes a small fraction of the effort required to create it. Simulation can reduce the time it takes to configure the component.

When problems with a component are found in the field, they are fixed and tested. If this is a reusable component, it is more robust when used in the subsequent design than in the original design. As a result, reusable components tend to improve over time, reducing costs associated with failures in the field.

Reusable components benefit from economies of scale. Not only are development and maintenance costs reduced, but larger quantities of the components can be purchased at one time, leading to lower unit costs.

Challenges of Component-Based Design

There are several challenges with implementing component-based development.

As mentioned earlier, designing a component that can be used in different applications takes longer and costs more. Suppose you have a single product that you refresh every 10 to 15 years. In that case, you are unlikely to see any benefits, but will face the additional development costs associated with developing a reusable component.

Over time, some parts become obsolete, which means components will have to be redesigned and retested. This is generally cheaper than developing the element from scratch, but it may impact the component's interfaces. A redesign might also introduce flaws that are not identified before the release of the component.

It isn't easy to anticipate how a component will be used in different applications. This difficulty increases with the complexity of the component. It's common practice to add many calibration parameters in both the hardware and software to counter this. Many calibrations can lead to a much longer time to calibrate the device for a specific application.

Examples of Component-Based Development

1

Automotive OEMs purchase sub-systems such as brake by wire, throttle by wire, and other sophisticated control systems from vendors. In the past, the OEM was forced to buy the hardware and software as a package from the vendor. As a result, they would have different hardware and software for the same type of sub-system on various vehicle platforms. However, to reduce time to market and development costs, many OEMs are adapting the AUTOSAR architecture. This allows the development or acquisition of software that can be reused across many vehicle platforms, independent of the underlying hardware. **There have been some initial examples of success using this approach.**

2

A medical device manufacturer creates reusable pump units that combine hardware and software. These pumps are integral to several products manufactured and sold globally. **Significant reductions in development times and costs have been realized through reusable components.**

3

Automotive OEMs have a few engines and transmissions that can be configured and used in dozens of vehicle models. The OEM uses large-scale DOEs to optimize fuel economy, performance, unit costs, etc. for a specific vehicle platform. **This results in significant savings in development time and cost.**

Over time, organizations have managed to reduce development costs through model-based and component-based development, even as regulations and device complexity grows. Using these tools and methodologies can make development methods more practical, and they can significantly reduce development time and cost and improve quality. However, if your organization requires an expert level of support, Oxford can help you implement these techniques and strategies.

**Data was pulled by Ken Macklem through his own expertise and research.*

About Oxford

Oxford Global Resources is known for our unmatched ability to deliver *The Right Talent.* **Right Now®**. As a leading staffing and consulting company with offices across North America and Europe, we focus on proactively building a network of highly skilled professionals so that we can immediately connect our clients to the expertise they need and provide rewarding opportunities for our consultants. We leverage over 35 years of recruitment expertise and specialize in fields where superior resource solutions are most in demand.

16,000+ CONSULTANTS with software system skills in our network

252,000 ENGINEERING CONSULTANTS in our network

381,000 IT CONSULTANTS in our network

173,200 COMPANIES supported with IT & Engineering staffing & consulting services

Learn what Oxford can do for your company at oxfordcorp.com

OXFORD™