# Addressing Future Automotive Needs with Model-Driven Systems Development

*Unprecedented Shifts Mandate New Approaches in Product Development and Reshape PLM*

Product Value Management

Collaborative Product
Development Associates, LLC
2001 West Main Street, Suite 222
Stamford, CT 06902
(800) 573-4756
www.cpd-associates.com

January 2009

PRODUCT LIFECYCLE MANAGEMENT
ROAD MAP™

## CPDA: Collaborative Product Development Associates, LLC

CPDA's Product Lifecycle Management (PLM) research programs target the critical decisions in Product Lifecycle Management challenging Design, Engineering, Manufacturing, and Information Technology managers and executives. CPDA's PLM collaborative research programs provide in-depth analysis of strategies, products, issues, processes, technologies, trends, case studies, and surveys for assessing technology, business goals and objectives, and implementation road maps.

The cohesive suite of collaborative programs clarifies and evaluates new capabilities, standards for frameworks, and development issues; it highlights the most advanced uses of leading technologies, and it links the technical effort to the realization of business value. The four collaborative research programs include:

**Design Creation and Validation*:*** A bottom-up view of engineering requirements from the desktop across the enterprise. Advanced computer-aided design (CAD), engineering analysis, manufacturing technologies, collaboration, and visualization software serve as springboards for gaining a competitive advantage. The Design Creation and Validation service applies CPDA's structured methodology to the evaluation of new products and processes as well as to current projects in client organizations. A critical focus, the emerging technology of knowledge engineering with templates and rule-based architectures focuses on delivering the needed tools into the hands of product developers to capture knowledge, and to formalize its use. The use of direct geometry access and manipulation, data translation technology, XML alternatives, and JT options are also assessed for their ability to deliver interoperability across the diverse and disparate business and technical applications.

**Design/Simulation Council*:*** The Council promotes a standard framework employing common terminology to integrate and optimize the diverse and divergent specialist activities currently fragmenting design efforts. CAE must fully integrate with design, up front, to close the chasm between design and analysis. Analysts must actively participate continuously in design decisions and enter the mainstream. The impending breakthrough in CAE will rest on knowledge reuse, process capture, and streamlining.

**PLM Integration / Product Definition:** A top-down view provides a conceptual framework for collaboration across different product development perspectives, bridging customer needs, systems engineering and tradeoffs, design solutions, and fulfillment and manufacturing. Integration and interoperability in complex PLM environments pose substantial hurdles. The rapid transition to cross-enterprise collaboration, at all levels of design and supply, intensifies the pressure on existing, inwardly focused IT architectures to support and enable new modes of doing business.

**Product Value Management:** Tools and reference models must be established and supported for mapping and optimizing enterprise processes in product development. The highest current priority applies to mechatronics to capture and relate all relevant requirements to development tasks across three disciplines: mechanical, electrical, and software design. In addition, considerable effort has been consistently applied in managing the challenge of product variants across sales and marketing, manufacturing, and development. Indeed, complex offerings of configurable products and bundles proliferate across many manufacturing industries.

Collaborative Product Development Associates was formed by the PLM research team of D.H. Brown Associates, Inc. (DHBA).

# *Addressing Future Automotive Needs with Model-Driven Systems Development*

*Unprecedented Shifts Mandate New Approaches in Product Development and Reshape PLM*

*Peter S. Robison, Director of Automotive PLM Solutions, Global Automotive Industry, IBM*

*Peter Robison holds responsibility for global strategy and direction of IBM's Global Automotive Industry as it relates to the Product Lifecycle Management domain, with specific focus on the Embedded Systems aspects. This includes alignment of capabilities across IBM's Industrial Sector, relationships with partners, and leadership of teams pursuing complex opportunities. Since joining IBM in 1995, Peter has helped automotive clients improve their NPI performance and has developed partnerships between IBM and independent software vendors. The efforts extended to automotive suppliers and the emerging markets of Central and Eastern Europe, as well as to Asia. In 1999, Peter was part of the management team that established IBM's Product Lifecycle Management direction. Peter also established IBM's 'Virtual Product Innovation' Competency addressing competitive innovation processes.*

*With over twenty years' experience in the automotive industry, Peter previously held responsibility for leading the Trim and Hardware Engineering department for the Rover Group and served as Body Engineering Chief Engineer for the whole of Rover Group (MG, Mini, Rover, and Land Rover). Peter started with engine design and analysis at Perkins Engines Company.*

*This report is derived from Peter Robison's keynote presentation at CPDA's PLM Road Map™ 2008 held September 23 & 24 in Detroit, Michigan.*

## EXECUTIVE SUMMARY

A study recently completed of the automotive industry involving discussions with 125 executives world wide raises major issues that will shape the future of PLM.[1] The highest priority issue, technology progress, will continue as a key industry enabler. Sustainability questions jump to the top of the list as well, as demand shifts towards hybrid and electric systems. Consumer concerns registered the largest percentage gain in the list, and extend to the issue of corporate social responsibility – doing the right thing for society as a whole. Overall, the discussions often focused on three very specific areas: the intelligent vehicle, the challenges of meeting the needs of sophisticated consumers, and the demand for dynamic operations in terms of vehicle connectivity.

[1] Sanjay Rishi, Benjamin Stanley, and Kalman Gyimesi, "Automotive 2020 – Clarity Beyond the Chaos," IBM Global Business Services, IBM Institute for Business Value, http://www-07.ibm.com/shared_downloads/6/IBM_Automotive_2020_Study_Clarity_beyond_the_Chaos.pdf

Five imperatives that automotive companies must address emerge from the 2020 study:

- Simplify complexity to maintain the agility to offer customers what they want.
- Focus on the development of personal mobility solutions as consumers will purchase "transportation services" with flexible access to different types of transportation.
- Transform the retail process – customers do not really like going to dealerships. They do a lot more shopping online, raising the need to configure vehicles online.
- Global execution represents a key to success.
- Partner extensively – this becomes a critical strategic alternative.

Mechatronic product development has added a new level of complexity to product development as companies strive to integrate across the software, electronics, and mechanical development domains for their products. Product development struggles to handle the complexities that we face today, and that complexity will intensify. This complexity must be addressed successfully to meet customer expectations and to make dynamic and rapid business decisions.

In terms of physical and logical coupling, automotive has traditionally involved relatively high coupling on mechanical systems and real-time software. Consider the concept of a connected vehicle, which must communicate vehicle-to-vehicle, or vehicle-to-infrastructure on road congestion issues or traffic. This kind of vehicle becomes more and more a system of systems, and raises the need to think about the larger environment. Not only are the developers managing the systems within the vehicle, but they are also linking the vehicle to other systems as well. Additionally, developers use a wide range of tools and techniques to develop and design the product across multiple disciplines. All these factors and more contribute to the trend of increasing complexity.

Product Lifecycle Management (PLM) faces challenges as it evolves from product engineering to systems engineering. Methodologies, solutions, and technologies must be available to evaluate complete system models and simulations. PLM must support collaboration, data reconciliation, and integration across disparate and dispersed disciplines.

Today, there is no integration between the physical modeling, the systems level modeling, and the software level modeling. The integration is poor as well between the functional behavior modeling and the simulation of dynamic systems. Methods, tools, and standards must support the control and flow of information between disciplines, and synchronize design efforts.

Model-driven systems development simplifies the complexity by managing the development process across the multiple disciplines and various tool sets at progressively deeper levels of abstraction. It begins with a high level understanding across collaborative disciplines of the system's context in terms of usage. At each level, it concentrates on understanding the full context of the system at that level, on understanding the collaboration required, and on understanding how the functions are distributed to achieve the system goals while meeting the defined constraints. Only those details appropriate to each specific level should be considered. Too much detail should not be used for any level that could encourage a decision prematurely in the process.

Model-driven product development also simplifies complexity by decomposing the system to model an increasing range of viewpoints as the level of detail increases. By effectively breaking down the complexity of a full system into specific levels, and decomposing the system to model multiple views, decisions are phased across the development lifecycle. As the level of detail rises with the collaboration of more and more design disciplines, the decomposition of the model delivers the appropriate detail relevant to each group without burying any of them with extraneous information.

Model-driven systems development manages product requirements as a critical step in the early stage to make sure that the product meets customer's expectations. By concentrating on the full breadth of views in collaboration across the major disciplines involved, it fully considers multiple viewpoints early in the process, which proves far more effective than the depth-first functional decomposition traditionally employed.[2]

Model-based development starts by describing the actual usage of the product being designed, rather than the design features. The system designed must meet real needs. The usage may involve combinations of various features or services. A comprehensive definition of the usage of the product then provides the basis to derive the necessary features and functions. The focus helps to meet the full range of all capabilities needed, and avoids targeting unnecessary functions.

Given the scarcity of skills, people cannot waste time hunting for information. It would be even worse if they found the wrong information and then used it. We must link the tools and use the automation available in those tools. Data will reside in multiple repositories, as there is no need to maintain all the data in one place. But several key pieces of that data do need to be associated and synchronized. The IT environments will be heterogeneous in serving the multiple repositories. While efforts may be made attempting to build PLM environments as a single integrated source, the systems still have to work with partners and suppliers. They have to work with the rest of the enterprise as well, so the enterprise PLM system will be heterogeneous.

[2] L. Balmelli, D. Brown, M. Cantor, and M. Mott, "Model-Driven Systems Development," *IBM Systems Journal,* Volume 45, No. 3, 2006, http://www.research.ibm.com/journal/sj/453/balmelli.pdf ; Brian Nolan, Barclay Brown, Dr. Laurent Balmelli, Tim Bohn, and Ueli Wahli, "Model Driven Systems Development with Rational Products," International Technical Support Organization, February 2008, http://www.redbooks.ibm.com/redbooks/pdfs/sg247368.pdf .

Another aspect of model-driven systems development relates to whole vehicle simulation. While simulations and analyses address specific aspects of the design quite well, trying to run a vehicle through a duty cycle or a simulation that includes stability control systems interacting with one another presents a very different series of challenges. The work here with a Japanese company concentrates on the notion of whole vehicle simulation. It also addresses "super real-time" targets of a tenfold improvement in performance over "real time."

A framework developed through research in Japan with one of the major OEMs and a first-tier supplier helps to lay out the context for model-driven systems development. First, business processes in terms of workflow, organization, and resources must be recognized. The program must cover the context of the requirement management system in understanding and decomposing the requirements. It must reconcile test and simulation results with requirements to validate the design well ahead of committing to physical production. It clearly must link to the PDM system to pull in a lot of information, but it also must link to the ERP system for costs.

Another effort with a German automotive firm views the product development process in a functionally oriented way, to validate and fully assess new innovation before applying it, in order to avoid recalls or warranty problems. Functional modeling also drives their reuse strategy by defining the functional areas that must be reused between different platforms. Viewing the vehicle program in terms of the functions that it needs to perform may promote reuse by making it much easier for people to find information. When people find a component, they often do not know whether it performs adequately for their particular purposes. The functional model improves the odds that people will pick a component or assembly and validate that it suits their need.

A final aspect relates to the linkage of the architecture to support business decision making. Once systems go into the vehicle, a change in the architecture may impact the interaction of the systems with one another. The important data resides in various repositories, and needs to be accessible and integrated to support decision making. With the model-based architecture, the IT environment is part of the model-driven approach. The business architecture also links leading decision makers even though the work is performed in the engineering department. Many critical decisions are made outside of the engineering department and the business decision makers are not necessarily fully up-to-date with the tools that the engineers use. They still need to be able to understand the evolution of the project, and they interact with the project leaders to make correct decisions. Overall, the scope of model-driven systems development fully encompasses both the IT environment and the business architecture.

These capabilities are not fully available today, ready to buy off-the-shelf. But the key aspects of model-driven systems development do address the needs of the future. The model-driven environment will emerge over the next few years. For context, one of the German car companies has recognized that their model-driven environment will involve a five-year journey. The transition does not happen overnight and no one should throw everything away that is already in place. The approach provides the critical linkages, and synchronizes the critical data in the various repositories and toolsets already in place.

## TABLE OF CONTENTS

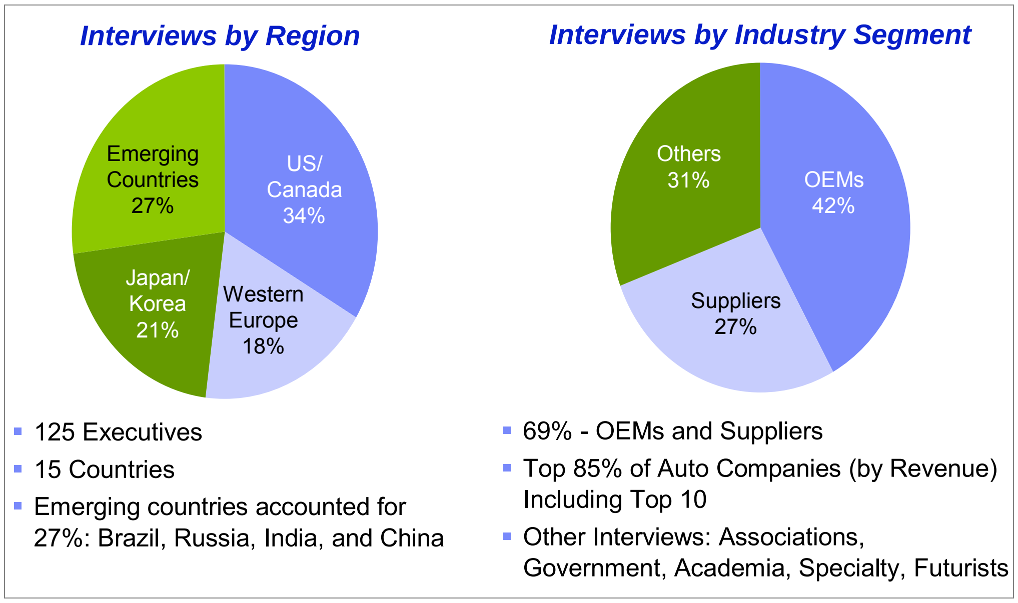
# *Addressing Future Automotive Needs with Model-Driven Systems Development*

*Unprecedented Shifts Mandate New Approaches in Product Development and Reshape PLM*

## TRENDS DRIVING THE AUTOMOTIVE TRANSFORMATION THROUGH 2020

A study completed recently of the automotive industry raises major issues that will shape the future of PLM. Discussions with 125 executives identified their perceptions of trends from now through the year 2020.[3] The group split about evenly both across the globe and between OEMs, suppliers, and others. (The "other" category includes a range of organizations such as academia, engineering and industry groups, and consultancies.)

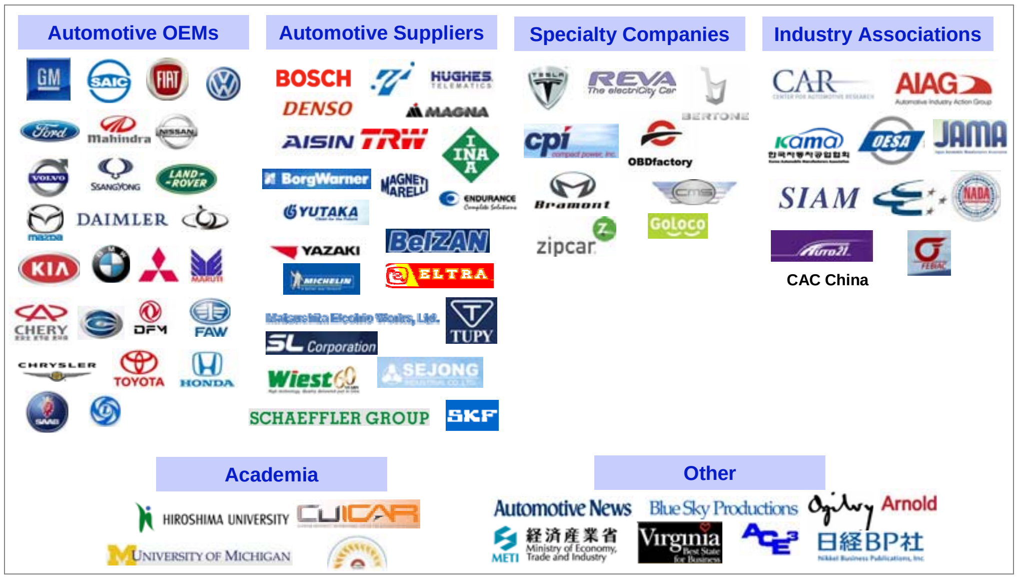
*FIGURE 1*
*Interviews were Global and Comprehensive*



*Courtesy IBM Corporation*

[3] Sanjay Rishi, Benjamin Stanley, and Kalman Gyimesi, "Automotive 2020 – Clarity Beyond the Chaos," IBM Global Business Services, IBM Institute for Business Value, http://www-07.ibm.com/shared_downloads/6/IBM_Automotive_2020_Study_Clarity_beyond_the_Chaos.pdf

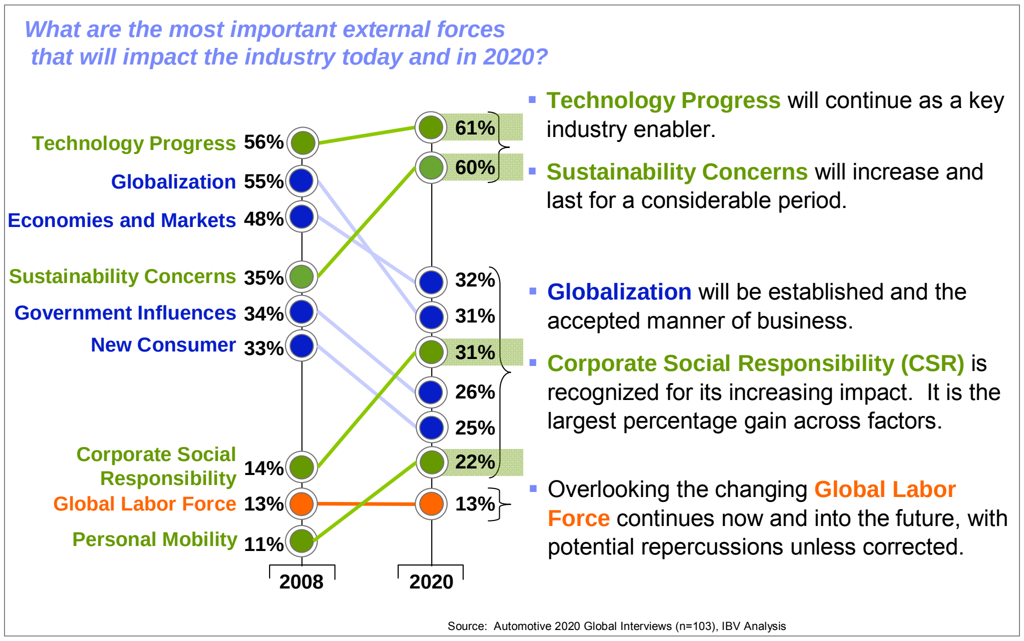
*FIGURE 2*
*Industry Leaders, Governments, Associations, Academicians, and Emerging Innovators were Active Participants*



*Courtesy IBM Corporation*

The highest priority identified was technology progress. This will continue as a key industry enabler. Sustainability concerns jumped to the top of the list as well as demand shifts towards hybrid and electric systems. The largest percentage gain expected by 2020 in the list, consumer concerns, extended to the issue of corporate social responsibility – doing the right thing for society as a whole. The integrated enterprise must shift the profile of the company to be more responsible in the world of interdependent ecosystems. Globalization dropped moderately as a priority, in the view of the respondents, between now and 2020. But that perception probably derives from the fact that many firms are already global, and globalization simply represents the norm. Economies and markets dropped in priority but remain a concern, possibly reflecting the tradeoffs from the pressures of the shakeouts that have already taken place, versus the challenges that continue.

*FIGURE 3*
*Priority Issues of Today Will Have Experienced a Major Shift by 2020*



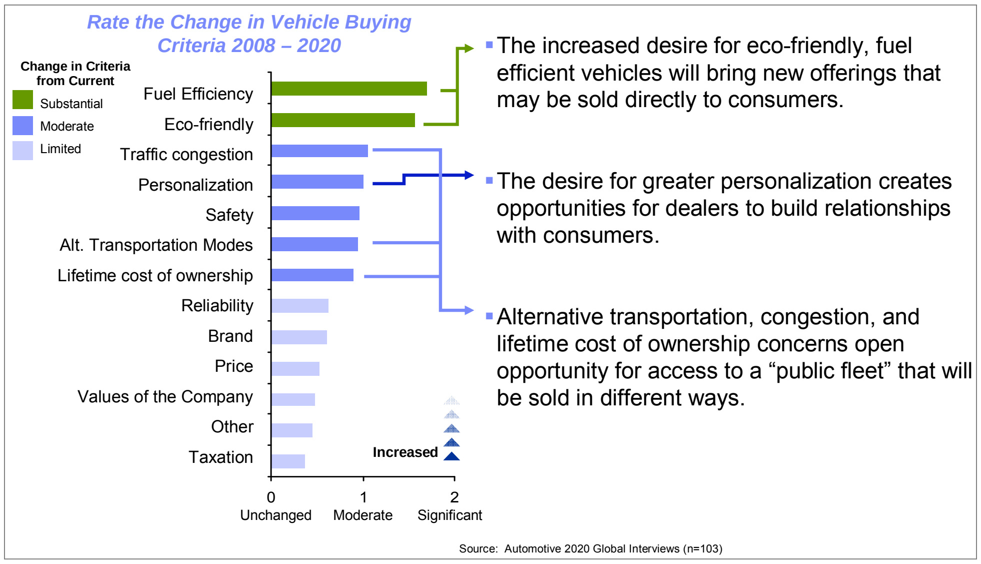
*Courtesy IBM Corporation*

## THE CHALLENGES OF PLEASING THE RISING SOPHISTICATION OF CONSUMERS

Overall, the discussions often focused on three specific areas: the intelligent vehicle, the challenges of meeting the needs of sophisticated consumers, and the demand for dynamic operations in terms of vehicle connectivity. With the intelligent vehicle the issue arises of customers enjoying the experience of the car. Most of us spend a lot of time in the car, often sitting in traffic jams. So people had best enjoy their time spent in the vehicle. The ability to respond to a rapidly changing dynamic environment becomes especially critical with the growing sophistication of customers. They may know more today about the products overall than many actually supplying those products. These customers get on the Internet, and they access all the positive and negative blogs out there about the products. They know what they want and they are adopting the idea of mobility. Rather than owning a number of cars, mobility presents the concept of certain types of transportation for specific types of activities or a particular time of the year. Going on vacation raises a very different set of transportation needs than commuting a few miles back and forth to work.

Consumers often look at items that are unrelated to the vehicle features. The services around the product or the brand become more important, as well as the increased desire to be eco-friendly and to support greater personalization. Alternative transportation, congestion, and lifetime cost of ownership concerns open consumer support for access to a public fleet. "Slugging," as an "in" phrase, represents a form of hitching a ride with people standing at particular locations looking for a lift. Active safety measures will help ensure that the vehicle does not get into an accident in the first place, and not just save the occupants when it does. Driver assistance may consider not only the fastest or the shortest route, but the most eco-friendly route. Data fed into a hybrid powertrain might consider an incline coming up and reserve energy in a kinetic recovery system. Remote diagnostics may cut down on the number of times someone has to take their vehicle to a dealership for repairs. The software might be downloaded into the vehicle.

*FIGURE 4*
*Sophisticated Consumers Will Challenge Auto Companies with Superior Product Knowledge and Buying Criteria Often Unrelated to Vehicle Features*



*Courtesy IBM Corporation*

## THE NEED FOR MODEL-DRIVEN SYSTEMS DEVELOPMENT TO ADDRESS RISING COMPLEXITY[4]

The growth of software in the vehicle, the disparate development tools utilized across disciplines, and the difference in lifecycle between the hardware and the software in the vehicle, create major challenges for the automotive industry. The majority of the people who responded to our study still have a lot of room for improvement.

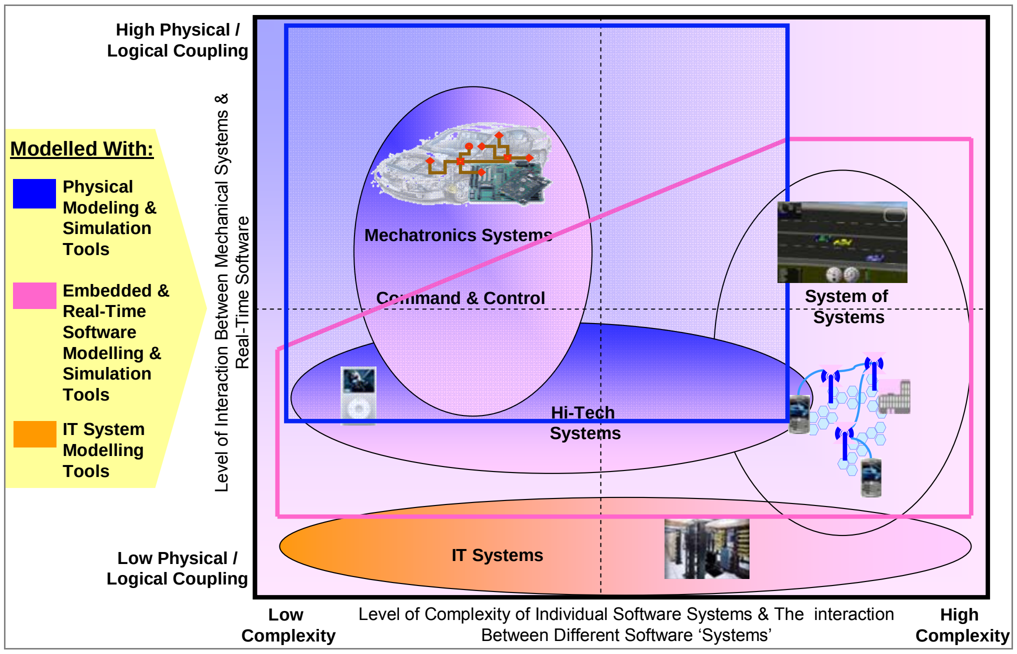
Mechatronics content continues to increase significantly in the final products. Best-in-class products rely more and more on the integration of mechanical, electrical, and software components. Mechatronics has added a new level of complexity to product development as companies strive to integrate across the three domains.

Compounding the challenge, there are also many forms of systems for product development. Each expert domain has unique characteristics in terms of standards and technology, skills required and culture, the development process, and constraints. The IT systems and telecommunications systems incorporate computer networks, middleware, and software applications as exemplified in PLM and collaborative environments. Highly coupled systems involve both physics and software with a mix of digital and analog logic. The command systems for embedded control or real-time systems involve multiple disciplines in development.

Large-scale systems-of-systems in aerospace and defense face increasing complexity as well. These systems-of-systems involve major interactions between multiple IT systems, with integrated logistics systems, telematics, and intelligent transportation systems. The whole environment of a mission being pursued may have to be covered, whether the mission involves a single fighter plane or whole battlefields. In each of these systems, the domains served have their own characteristics that compound the complexity of any solution.

[4] L. Balmelli, D. Brown, M. Cantor, and M. Mott, "Model-Driven Systems Development," *IBM Systems Journal,* Volume 45, No. 3, 2006, http://www.research.ibm.com/journal/sj/453/balmelli.pdf ; Brian Nolan, Barclay Brown, Dr. Laurent Balmelli, Tim Bohn, and Ueli Wahli, "Model Driven Systems Development with Rational Products," International Technical Support Organization, February 2008, http://www.redbooks.ibm.com/redbooks/pdfs/sg247368.pdf .

*FIGURE 5*
*Positioning the Different Kinds of "Systems"*


High Physical / Logical Coupling
Modelled With:
Physical Modeling & Simulation Tools
Embedded & Real-Time Software Modelling & Simulation Tools
IT System Modelling Tools
Level of Interaction Between Mechanical Systems & Real-Time Software
Low Physical / Logical Coupling
Mechatronics Systems
Command & Control
Hi-Tech Systems
System of Systems
IT Systems
Low Complexity
Level of Complexity of Individual Software Systems & The interaction Between Different Software 'Systems'
High Complexity

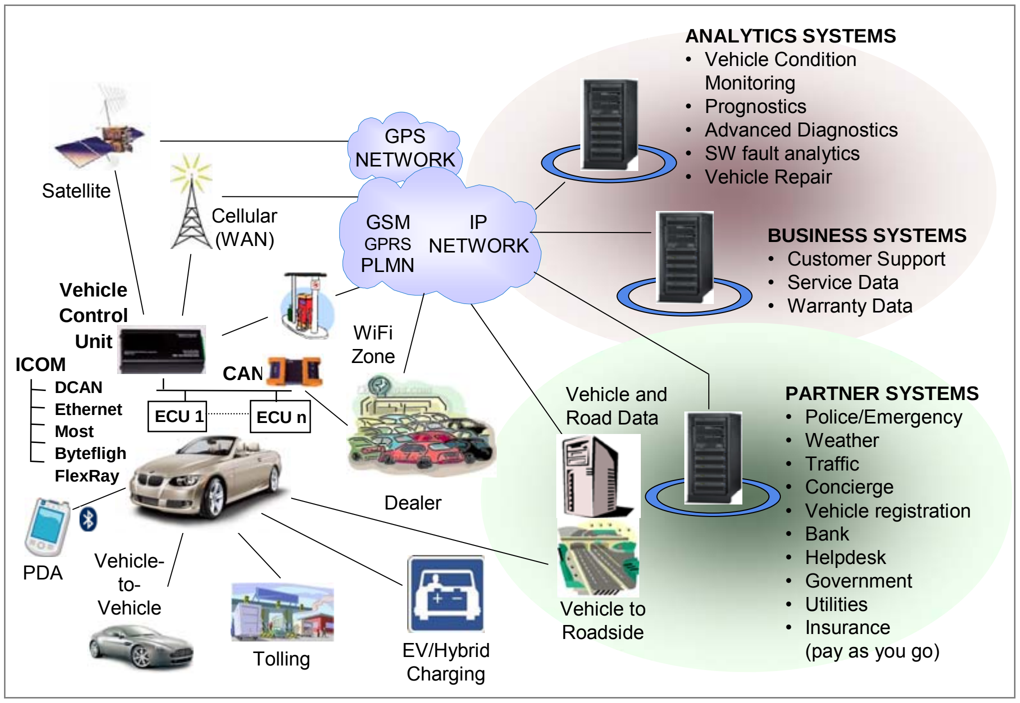
*Courtesy IBM Corporation*

Consider the broad range of systems in terms of physical and logical coupling, which forms the left-hand side of the plot above. That is the coupling between mechanical systems and real-time software. Along the bottom, the scale reflects the complexity of individual software systems and software-to-software interconnections. The chart positions several industries. Automotive has traditionally involved relatively high coupling on mechanical systems and real-time software. A high-tech iPod would be positioned on the left, and a communication system on the right.

Systems-of-systems reside more to the right-hand side. The automotive industry is moving to system-of-systems. As an example, consider the connected vehicle which includes not only the electronics but the typical architecture within the vehicle.

But now the vehicle must communicate vehicle-to-vehicle, or vehicle-to-infrastructure, on road congestion issues or traffic. Vehicle-to-satellite communications may also be involved. The vehicle becomes more and more a system of systems. Not only are the developers managing the systems within the vehicle, but they are also linking the vehicle to other systems. Additionally, they use a full range of tools and techniques to develop and design the product across a range of disciplines. All these factors add to the complexity of the overall environment, which reinforces the need for model-driven techniques.

*FIGURE 6*
*The Connected Vehicle: 'A System of Systems'*



*Courtesy IBM Corporation*

## ADDRESSING COMPLEXITY WITH MODEL-DRIVEN PRODUCT DEVELOPMENT

Model-driven product development directly addresses rising complexity by breaking down challenges into multiple levels of abstraction. It begins with a high level understanding of the system's context focusing at first on how the system is used. The level of detail and specificity deepens through a series of transformations that maintains traceability, and supports the coherence of the entire model. Only those details appropriate to each specific level are considered. This helps avoid making decisions too early in the process based on details that are likely to be changed or made obsolete later; such details are often derived from faulty, incomplete information.

Model-driven product development also simplifies complexity by decomposing the system to model an increasing range of viewpoints as the level of detail increases. For large-scale development programs, design takes place across multiple viewpoints concurrently, involving the distribution of functionality to various pieces of the system. The decomposition covers the full spectrum of disciplines involved in product development.

By effectively breaking down the complexity of a full system into specific levels, and decomposing the system to model multiple views, decisions are phased across the development lifecycle. As the level of detail rises with the collaboration of more and more design disciplines, the decomposition of the model delivers the appropriate detail relevant to each group without burying any of them with extraneous information.

The decomposition level refers to the depth of the phase in defining the structural hierarchy of the system. As an example, in reasoning at an early and high level about the enterprise, entities must be employed that are appropriate for that level of decomposition to resolve any issues at that level until it is appropriate to go to the next level of decomposition.

## FOCUS EARLY ON REQUIREMENTS

PLM faces challenges as it evolves from product engineering to systems engineering. Methodologies, solutions, and technologies must be available to evaluate complete system models and simulations. They must support collaboration, data reconciliation, and integration across disparate and dispersed disciplines.

Today, there is no integration between the physical modeling, the systems-level modeling, and the software-level modeling. The integration is poor as well between the functional behavior modeling and the simulation of dynamic systems. The methods, the tools, and the standards must support the control and flow of information across disciplines, and synchronize the design efforts.

We are seeing our top customers driving best practices in the core discipline of systems engineering. No longer concentrating on requirements in isolated disciplines, they are managing requirements across the whole development process including software, mechanical, and electronics domains. The same focus and rigor accorded to managing the bill of materials now drives the management of requirements as well. We see the need to move from a tool- or paper-based approach to model-based management. The issue does not concern management and documentation so much as the full validation of a design against tradeoffs and changes that impact requirements.

While the automotive industry has been very good in addressing high level requirements, the systems generally do not address the need for detailed reconciliations. Products today, such as a car, have thousands of detailed requirements that must be mapped to thousands of components, resulting in millions of possible mappings. In the face of this complexity, developers will not likely optimize across the full range, and end up limiting the level of integration.

Model-driven product development starts with the definition and understanding of requirements and moves on to design specifications. Linking requirements involves the management of every design change, interaction, and decision against its impact on requirements.

⇒ Focus on Systems Engineering
To Help Manage Complexity Across Disciplines

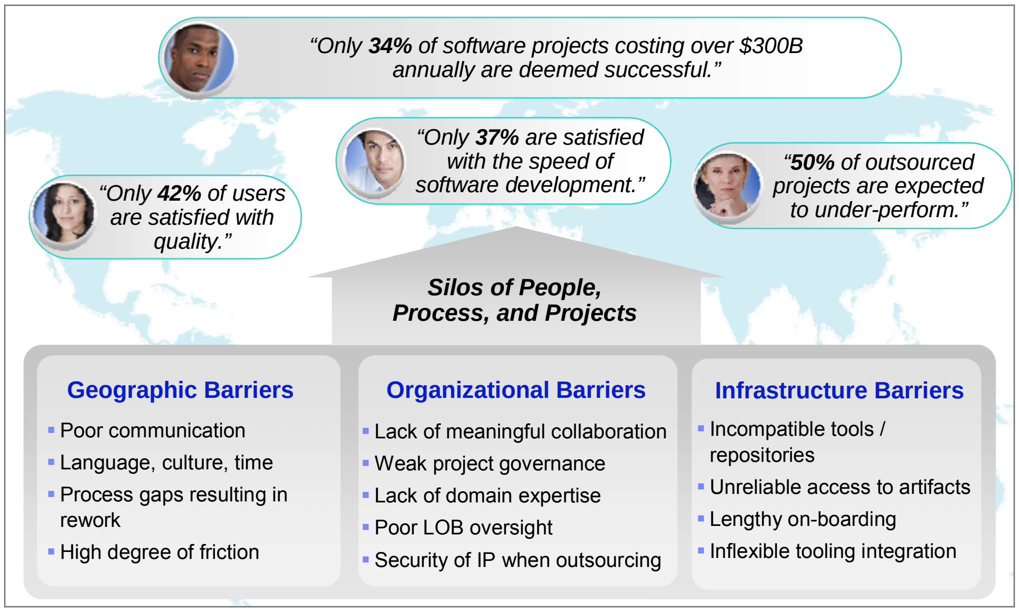
According to Embedded Market Forecasters…
"Model-Driven Development is used to more clearly analyze requirements, define design specifications, test systems concepts using simulation, and automatically generate code for direct deployment on the target hardware."

"A most important aspect of design outcomes is the closeness of final design outcomes compared with pre-design expectations."

The need arises to close the communication gap between those in marketing who are defining customer requirements and the engineering and manufacturing communities who have to develop and deliver the product. The solution must prevent customer-required features from falling through the cracks. Warranty costs, rework, and delays must be slashed. Several of the luxury brands in Europe had a disaster with failures in software and electronics for their vehicles. Typically in Europe, the companies now budget roughly 700 Euros for warranty costs during the warranty period. In fact, some of the luxury car brands have experienced actual figures at five times that level in the first six months on the market because of software and electronic problems. Clearly, a lot of lessons need to be learned in fully understanding, defining, and supporting requirements. Requirements management must not only define what the end customer wants, but it must reconcile the testing and simulation results to validate the design.

Software delivery presents a major opportunity to streamline the overall process using standards such as Eclipse and Jazz (open-source development and collaboration platforms). Currently, software development can involve as many as two hundred tools across the full cycle through the V-model. Today, it looks a bit like CAD and mechanical design did fifteen years ago. None of the tools are integrated; all involve manual interactions.

*FIGURE 7*
*Focus on Software Delivery to Help Differentiate Products while Controlling Costs*


"Only **34%** of software projects costing over $300B annually are deemed successful."
"Only **42%** of users are satisfied with quality."
"Only **37%** are satisfied with the speed of software development."
"**50%** of outsourced projects are expected to under-perform."
Silos of People, Process, and Projects
Geographic Barriers
Poor communication
Language, culture, time
Process gaps resulting in rework
High degree of friction
Organizational Barriers
Lack of meaningful collaboration
Weak project governance
Lack of domain expertise
Poor LOB oversight
Security of IP when outsourcing
Infrastructure Barriers
Incompatible tools / repositories
Unreliable access to artifacts
Lengthy on-boarding
Inflexible tooling integration

*Courtesy IBM Corporation*

Model-driven systems development (MDSD) transforms the process beginning with concept definition. It addresses business issues as much as engineering. It addresses structural issues in the organization and presents insight from diverse data sources early in design. This directly improves decision making. To understand a program's viability, the markets and volumes must be accurately targeted to meet a minimal internal rate of return. It's a business case.

Quite often, in evaluating a program's viability, very little hard-core engineering needs to be applied early in the effort. Indeed, hard-core engineering may be counter-productive:

> Engineers have a tendency to want to jump down to the details. So when they talk about a system for getting you to your destination, they are as likely to talk about problems with the air control software or the wiring of a piece of hardware as they are to talk about larger-grained issues. This can lead to confusion and errors – diving too deep too early causes integration problems and constrains a solution too early. Requirements are usually best understood in context; jumping levels leads to a loss of context.[5]

The start of a vehicle program does not involve a lot of deep-down engineering. MDSD concentrates first on the breadth of collaboration across multiple viewpoints as a more effective approach than a traditional depth-first functional decomposition. The process assures an effective distribution of responsibilities with the joint realization across collaborative design disciplines. The analyses and decisions are far more resilient in the face of inevitable change. Indeed, the whole focus of model-driven systems development on the front end concentrates on understanding the business criteria and context before pressing the button on a major investment of engineering time and resources, let alone manufacturing.

MDSD promotes a breadth-first analysis of functionality across the set of collaborating disciplines in product development. It starts at a relatively high level with use cases that lay out the sequence of events which describe the collaboration between the system and external actors to accomplish the goals of the system. The use cases specify the behavior required of both the system and external actors.

Model-driven systems development begins with an understanding of the system's context in terms of its usage. The context involves the people, entities, and other systems – or actors that interact with the system. It involves understanding the usage that delivers value, or the services required of the system, and the exchanges between the system and the external actors. Focusing on the way that context drives usage helps with the discovery of the requirements, which in turn ensures that the system meets the stakeholder needs.

[5] Brian Nolan, Barclay Brown, Dr. Laurent Balmelli, Tim Bohn, and Ueli Wahli, "Model Driven Systems Development with Rational Products," International Technical Support Organization, February 2008, http://www.redbooks.ibm.com/redbooks/pdfs/sg247368.pdf .

All too often, large systems are built based on requirements written by teams of people with varying ideas about the real needs, and a partial view of how the system may be used. Features may be elaborated without any realistic connection to how the system may actually be used. Those involved may even imagine the use of the system, but they write about requirements, features, and attributes. Those reading the documentation later create their own view of the usage, leading to miss understandings and wrong assumptions. The full understanding and usage of the system is not clearly defined and developed.

A comprehensive set of system usages serves as a base to derive the necessary features and functions. Those same usage-based models serve to design subsystems and components, while supporting traceability. Even the finest detail of an operation may be traced to the particular usage of the system.

## MODEL LEVELS OF ABSTRACTION AND OF DECOMPOSITION

The next stage addresses design and engineering tradeoffs to select the optimal solution. What systems, what existing technologies or carryover designs are available for this vehicle? What level of invention is needed for the product? Then the third stage addresses the range of customer needs, and segments that may be addressed to maximize the coverage of the market with variants and options.

Decomposition divides the system into elements that make up the system, and makes it easier to understand the way the systems meets the needs of the user. The divide-and-conquer approach defines a comprehensible set of elements, each of which addresses an understandable set of requirements. It addresses the limited capacity of humans to directly cope with the full complexity involved.

System decomposition must model the system from a variety of viewpoints covering the full spectrum of disciplines involved in product development. For large-scale development programs, design takes place across multiple viewpoints concurrently, involving the distribution of functionality to various pieces of the system.

In effect, model-driven development addresses complexity through abstraction. Complex systems are modeled at different levels of specificity as the model undergoes a series of transformations. Each transformation adds levels of specificity and detail, maintains traceability, and supports the coherence of the entire model. Model levels group artifacts at a similar level of detail. The artifacts cover any item that describes the architecture, such as a diagram, matrix, or text document. The models are also customizable to the specific needs and terminology of a particular product development program.

Managing the level of abstraction involves the appropriate level of detail for each level. Two kinds of levels are involved – model levels and levels of decomposition. The model level refers to the phase of thinking. As an example, analysis models should be less detailed than design models. Too much detail should not be used for entities for a specific phase and level that would encourage or even force a decision too early in the process. The decomposition level refers to the depth of the phase in defining the structural hierarchy of the system. As an example, in reasoning at an early and high level about the enterprise, entities must be employed that are appropriate for that level of decomposition to resolve any issues at that level until it is appropriate to go to the next level of decomposition.

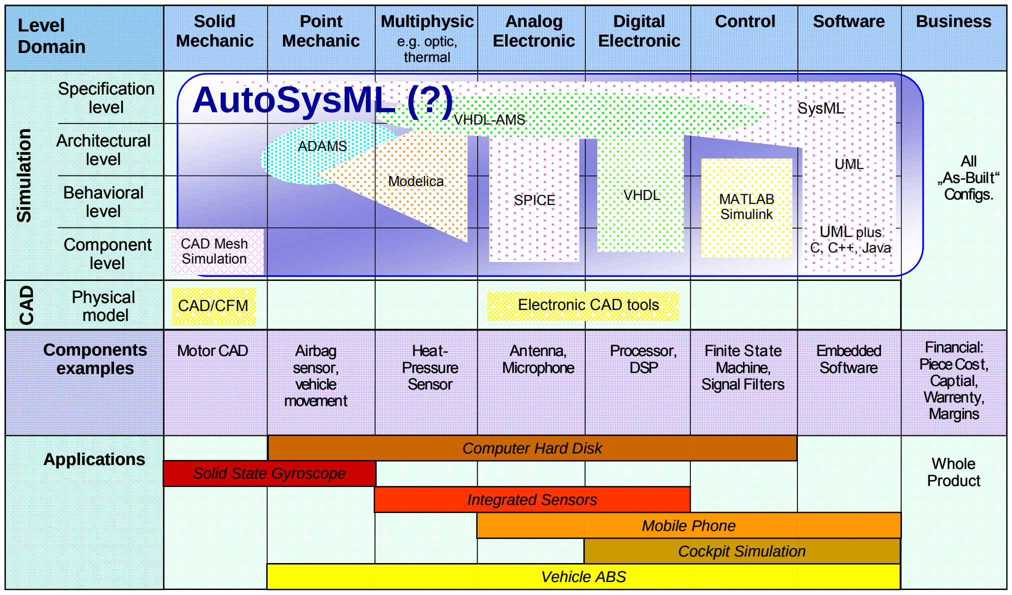
For each level of abstraction, or each model level and each level of decomposition, the same activities repeat. They target an understanding of the context of the system, an understanding of the collaboration required to achieve the goals, and an understanding of how function is distributed across form to achieve the system goals while meeting the set of constraints. Each progressive step in defining context, defining collaborations, and specifying the distribution of responsibilities across disciplines highlights ambiguities, uncovers problems, and provides the opportunity to correct mistakes early in the development process.

## SYSML REPRESENTATION FOR SYSTEMS AND STRUCTURE

One of the major challenges in terms of modeling is the proliferation of modeling languages that simulate the performance of the vehicle from a technical point of view. We at IBM are working with academia and several customers to try and model the financial side as well. The big ticket items in any product portfolio are those that make or break profitability, or risk a targeted launch date. Building in critical aspects of the business such as the configuration, financial variables, piece cost, capital utilization, warranty margins, and whole lifecycle costs, supports an understanding of the business viability by modeling a product at the early stage. We can then understand the business viability as well as the technical feasibility.

One of the areas being explored concerns the extension of languages like SysML to the automotive industry. As those who know languages such as SysML understand, they are not very good at pure mechanical design. They are also not very good at the aesthetics, which are also important in automotive. However, we may be able to actually extend them far enough to model the whole vehicle – not just systems in the vehicle, but the whole vehicle. That presents one area that we at IBM are focusing on.

*FIGURE 8*
*Positioning of Current Modelling Languages*



*Courtesy IBM Corporation*

SysML is good as a modeling language for representing systems and product architectures, as well as their behavior and functionalities. Starting with the early phases of design, it effectively supports the conceptual stage of development by directly representing the product features defined by the decomposition of customer needs. The conceptual stage should clarify how the product behavior is expressed through the interaction of its components. The formal description of the product at an early stage improves the understanding of the product requirements and how well they meet the customer needs. The features may be represented as requirements in model driven systems development, which in turn may be allocated to sub-systems and components for the product.

SysML constructs also support the description of the structure of the product in terms of "blocks" with dependencies expressed using constraints. As a basic structural element, blocks provide a discipline-agnostic tool for building systems. Blocks may represent any type of component of the system whether it is functional, physical, or human.

Three mechanisms describe product behavior, as interactions, as a state machine, and as activities. They can provide a unified behavior concept that may be orchestrated in a single, uniform, and complex model for the whole product.[6]

[6] Laurent Balmelli, "An Overview of the Systems Modeling Language for Products and Systems Development," *IBM Technical Report 2006,* rev.14, 10/2/06, p. 23, http://www.ibm.com/developerworks/rational/library/aug06/balmelli/ ; Sandy Friedenthal, Alan Moore, Rick Steiner, *OMG Systems Modeling Language (OMG SysML™) Tutorial*, Object Management Group, July 11, 2006, http://www.omgsysml.org/SysML-Tutorial-Baseline-to-INCOSE-060524-low_res.pdf .

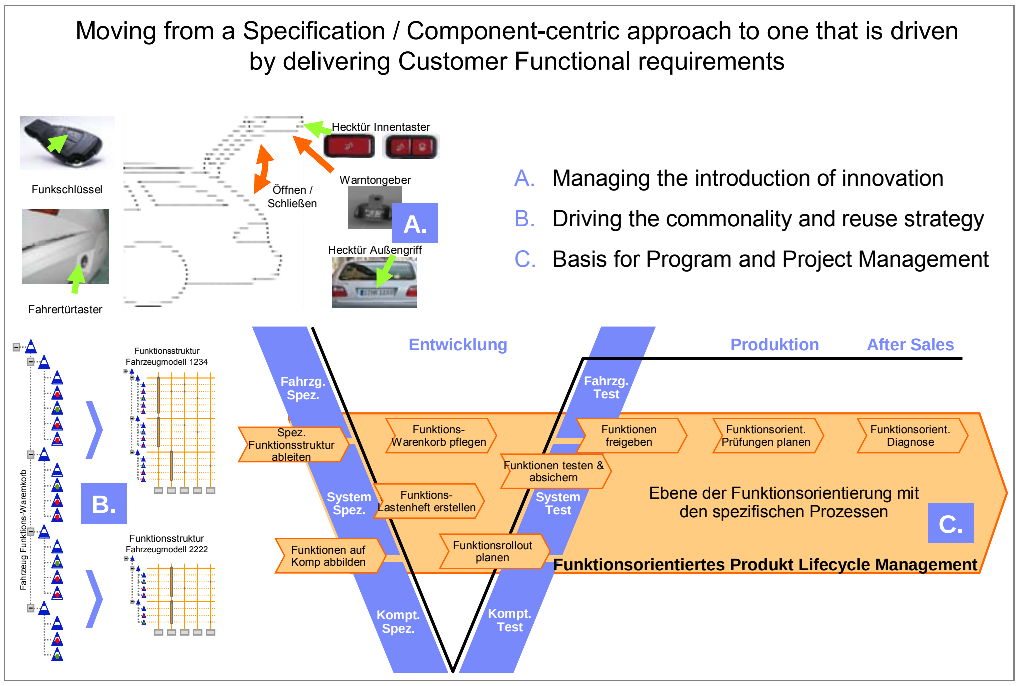
## JOINT EFFORTS WITH LEADING JAPANESE AND GERMAN CUSTOMERS

A framework developed through research in Japan with one of the major OEMs and a first-tier supplier helps to lay out the context for model-driven systems development. First, business processes in terms of workflow, organization, and resources must be recognized. The program must cover the context of the requirements management system in understanding and decomposing the requirements. It must reconcile test and simulation results with requirements to validate the design well ahead of committing to physical production. It clearly must link to the PDM system to pull in a lot of information. It also must link to the ERP system for costs.

Another effort views the product development process in a functionally oriented way with a German automotive firm. We will still need to automate the effort with CAD modeling systems, and leverage the knowledge. But looking at a vehicle program in terms of the functions that it needs to perform may promote reuse by making it much easier for people to find information. One of the issues of reuse as well is that when people find a component they do not know whether it performs adequately for their particular purposes. The functional model improves the odds that people will pick a component or assembly and validate that it suits their need.

This particular German OEM is implementing functional modeling to validate and fully assess new innovation before applying it to avoid recalls or warranty problems. The second aspect relates to driving their reuse strategy by defining the functional areas that must be reused between different platforms. Finally, they are actually using the functional modeling and the functional description to manage their product development and gateway process. While that represents a lot of work, the cost is fully justified because it avoids the failures in the marketplace and the lack of profit margin that too often plagues new vehicle programs.

*FIGURE 9*
*Functional Oriented PLM*



Moving from a Specification / Component-centric approach to one that is driven by delivering Customer Functional requirements

A. Managing the introduction of innovation
B. Driving the commonality and reuse strategy
C. Basis for Program and Project Management

Funkschlüssel
Fahrertürtaster
Öffnen / Schließen
Hecktür Innentaster
Warntongeber
A.
Hecktür Außengriff
Funktionsstruktur Fahrzeugmodell 1234
Funktionsstruktur Fahrzeugmodell 2222
Fahrzeug Funktions-Warenkorb
B.
Entwicklung
Produktion
After Sales
Fahrzg. Spez.
Spez. Funktionsstruktur ableiten
Funktions-Warenkorb pflegen
System Spez.
Funktions-Lastenheft erstellen
Funktionen auf Komp abbilden
Kompt. Spez.
Funktionsrollout planen
Funktionen testen & absichern
System Test
Kompt. Test
Fahrzg. Test
Funktionen freigeben
Funktionsorient. Prüfungen planen
Funktionsorient. Diagnose
Ebene der Funktionsorientierung mit den spezifischen Prozessen
C.
Funktionsorientiertes Produkt Lifecycle Management

*Courtesy IBM Corporation*

## WHOLE VEHICLE SIMULATION

Another aspect of model-driven systems development relates to whole vehicle simulation. While simulations and analyses address specific aspects of the design quite well, a problem starts to arise in truly simulating the whole vehicle. Consider ride and handling. The designers can check the body stiffness, the compliance of the chassis systems, the power steering and the hydraulics. But try and run a vehicle through a duty cycle or a simulation run that includes stability control systems – then have them interact with one another. That presents a very different series of challenges. The work here with a Japanese company concentrates on the notion of whole vehicle simulation. It also extends to "super real-time" targets of a tenfold improvement in performance over "real-time" to develop more products than before, probably with fewer engineers. Those products benefit from a shorter development lifecycle with the simulation of all the configurations of the product for the market. Super real-time supports far more configurations.

## LINKING BUSINESS DECISION MAKERS

A final aspect relates to the linkage of the architecture for business decision making. Once systems go into the vehicle, a change in the architecture may impact the interaction of the systems with one another. How do the various PLM and other tools provide the information needed to make decisions? The important data resides in various repositories, and needs to be accessible and integrated to support decision making. One approach would try to put everything into one enormous repository. That isn't going to happen any time soon. We also need to be able to connect the data and link it to decisions.

The business architecture links leading decision makers even though the work is performed in the engineering department. Many critical decisions are made outside of the engineering department, and the business decision makers are not necessarily fully up to date with the tools that the engineers use. They still need to be able to understand the evolution of the project, and they interact with the project leaders to make correct decisions. Overall, the scope of model-driven systems development fully encompasses the IT environment and the business architecture.

## SUMMARY CONCLUSION

To sum up, current practices are struggling to handle the complexity we are dealing with today to meet customer expectations. That complexity may be effectively addressed by breaking down the problems into multiple levels of abstraction, and decomposing the system into models addressing multiple viewpoints with deeper levels of detail. As the depth of detail increases, the specific needs of the full range of engineering disciplines are met, and issues resolved.

Starting at a high level and focusing on usage, managing product requirements represents a critical step at the early stage of development to make sure that the product meets customers' expectations. A requirements framework manages and facilitates synchronization for the development process across the various tool sets used in the organization.

We also need to be able to make dynamic and rapid business decisions even though critical data resides in multiple repositories. We do not need all the data in one place, but several key pieces of that data do need to be associated and synchronized. We may try to build our PLM environments as a single integrated source, but that still has to work with partners and suppliers. It has to work with the rest of the enterprise as well, so it will be heterogeneous.

These capabilities are not fully available today, ready to buy off-the-shelf. But the key aspects of model-driven systems development do address the needs of the future. The model-driven environment will emerge rapidly over the next few years. For context, one of the German car companies has recognized that their model-driven environment will involve a five year journey. The transition does not happen overnight and no one should throw everything away that is already in place. The approach provides the critical linkages, and synchronizes the critical data in the various repositories and toolsets already in place.