Decision Making in Manufacturing and Services
Vol. 1 • 2007 • No. 1–2 • pp. 137–152

A Distributed Decision-Support System for Virtual Prototyping

Thomas M. Tirpak*, Lawrence E. Lach**,
Weimin Xiao***, Juan M. Lopez****

Abstract. Virtual Prototyping (VP) is a data-driven design process that promotes both knowledge reuse and innovation. High-profile applications in the automotive and aerospace industries have demonstrated its potential to significantly reduce prototype cycles, time to market, and total product cost. This paper addresses VP as a specialized application of Decision-Support Systems, and discusses common requirements for engineering design tools, as well as requirements specific to the design of electronic products, such as mobile phones. Motorola Labs' test bed for VP is introduced in terms of its open, agent-based architecture utilizing Java CORBA. One of the key principles of the VP System is the reuse of expert knowledge across multiple engineering domains. This is highlighted via several use cases, showing that the system can function not only as an Intranet-accessible repository of model services but also as an integral part of decision-making within the native CAD environment.

Keywords: Distributed Decision-Support, Virtual Prototyping, CAE.
Mathematics Subject Classification: 68U35, 68U07, 90B50.
Received/Revised: 03 December 2006/20 April 2007

1. INTRODUCTION

Especially in high-tech industries, innovation and time-to-market both play a significant role in a company's ability to compete in the marketplace. Thus, it is necessary to provide product development teams with tools that guide decision-making and collaboration. For almost four decades, the field of Computer-Aided Engineering (CAE) has developed along with, and as a driver for, advances in computing and networking technologies. CAE, and its better-known cousin Computer Aided Design (CAD), are

* Motorola Home & Networks Mobility Business, Schaumburg, U.S.A.
E-mail: T.Tirpak@Motorola.com (Corresponding Author)
** Motorola Labs Physical Realization Research Center, Schaumburg, U.S.A.
E-mail: Q12020@email.mot.com.
*** Motorola Labs Physical Realization Research Center, Schaumburg, U.S.A.
E-mail: AWX003@email.mot.com.
**** Motorola Labs Physical Realization Research Center, Schaumburg, U.S.A.
E-mail: AJL068@email.mot.com.

two activities within the larger framework of Product Lifecycle Management (PLM), whose goal is to enable companies to quickly and profitably develop and support products satisfying the requirements of their customers.

A study conducted by the U.S. National Academies (National Research Council, 2004) characterized Advanced Engineering Environments (AEEs) as integrated computational systems and tools that facilitate design and production activities within and across organizations, which may include:

— Design tools such as computer-aided design (CAD), computer-aided engineering (CAE), and simulation.

— Production tools such as computer-aided manufacturing (CAM), manufacturing execution systems, and workflow simulation.

— Program management tools such as configuration management, risk management, and cost and schedule control.

— Data repositories storing integrated data sets.

— Communications networks giving participants inside and outside the organization secure access to data.

To address future needs for AEEs, the National Academies have proposed a Framework for Virtual Manufacturing. It should be noted that software tools are not currently available for many required product development activities. For other activities, tools are readily available or emerging, but they are not interoperable or are used inefficiently. This framework includes Virtual Prototyping as a means to implement product verification and validation activities.

According to Sandborn (2000), there is general agreement that a virtual prototype is a model that can be evaluated against system requirements prior to major design or manufacturing investment. At the most basic level, the term "Virtual Prototyping" refers to the creation and use of simulation models to guide product development decisions. Research and applications of Virtual Prototyping, however, have addressed a wide range of computer-assisted, data-driven activities throughout the lifecycle of a product, i.e., from the initial idea for a product to its support in the marketplace, and have sought improvements in concept exploration, requirements definition, design, and design validation. One aspect of Virtual Prototyping, namely design visualization or Digital Mock-ups (DMUs), has received much attention, because of its benefits for evaluating early product concepts and for assembly planning, e.g., detecting potential collisions/interference of parts and validating the integration of subsystems.

As noted by Thomke (2003), computer-aided design experimentation has enabled companies to "front-load" their development processes and thereby reduce cycle times. Virtual Prototyping (VP) has been used widely in the automotive and aerospace industries. For example, Durstewitz (2002) describes a virtual collaboration environment for aircraft design, which includes modeling, visualization, and shared workspace tools. Murphy (2001) addresses the many types of simulation that are performed during the lifecycle of an aerospace product, and the manner in which digital prototypes can support product development decisions. Park (2005) presents an e-Engineering

framework being developed for automotive suspension module design, based on intelligent software agents, Internet/Web, workflow, optimization, and Product Data Management (PDM).

Danesi (2006) discusses a product development methodology including co-design and distributed design. A digital mock-up (DMU) is at the center of communication and interaction. Li et al. (2004) present a Complex Product Virtual Prototyping Environment that includes a collaborative design support platform, model depository with environmental models, product model database, virtual prototyping engine, and visualization environment. Applications involve electronic, mechanical, control, software, etc., subsystems, and the approach utilizes a collaborative grid simulation platform.

Research and applications of VP have also focused on hardware-software co-simulation and validation of code on sets of simulated integrated circuits (ICs). Such models can provide critical insights regarding what product functionality should be implemented as hardware and what should be implemented as software. For example, Belanovic (2004) discusses a system for automated generation of virtual prototypes to guide the design of embedded systems.

Selko et al. (2006) divides the cycle time for product design as: imagining what you want, finding the right information and interpreting it correctly, and using design tools. They discuss a workflow-driven user interface that helps to quickly find and reuse existing parts and systems, reuse industry and corporate standards, capture and reuse expert knowledge, and capture and reuse engineering processes. The Federated Intelligent Product Environment (FIPER) `http://www.fiperproject.com` and the Enterprise Accessible Software Application (EASA) `http://www.easa.aeat.com/index.htm` are two examples of toolkits for capturing engineering processes and integrating supporting services, such as design simulation and optimization. In essence, these are Business Process Modeling (BPM) tools specifically configured to manage the workflows and computerized resources of engineering and product development organizations.

This paper overviews the development of the Motorola Labs Virtual Prototyping System as a test bed for investigating methods and software for evaluating and optimizing designs, and exploring design innovations. Section 2 discusses some of the challenges of designing electronic products and summarizes the requirements for the VP System. Section 3 presents the open system architecture that was developed, based on a collection of software agents and the Java Common Object Request Broker Architecture (CORBA). Section 4 explains the operation of the VP System as an Intranet-accessible set of design evaluation services. Section 5 addresses the use of the system as an automated design-checker, launched from within a CAD tool. Section 6 offers some concluding remarks and recommendations for future work.

2. REQUIREMENTS FOR A VIRTUAL PROTOTYPING SYSTEM

Building on many years of experience in modeling mechanical and electrical systems, as well as manufacturing process, e.g., (Tirpak, 2002), Motorola Labs formed a working group in 1998 with the charter of creating "a data-driven design process that

enables innovation and reduces the number of prototype cycles, time to market, and total product cost while meeting the quality, performance, reliability and value needs of our customers and markets." This section of the paper highlights the need for next-generation engineering tools, as initially characterized by the core team, as well as their recommendations for developing a Graphical User Interface (GUI) to support decision makers. Challenges specific to electronic product development and the overall goals for the Virtual Prototyping System are likewise discussed.

2.1. OPPORTUNITIES TO IMPROVE ENGINEERING TOOLS

Product development teams typically use a number of different software tools, some of which are internally developed and some of which are licensed from third-parties. A 1999 survey found that within Motorola, a variety of tools were used for electrical design, e.g., Mentor Board Station and Cadence, for mechanical design, e.g., Pro/Engineer, and for Design-to-Manufacturing (DTM) information management, e.g., Tecnomatix Unicam. There was also a significant number of internally developed databases with manufacturing/component information, e.g., the Aladdin Product Cost Estimation Tool (PCET), and product performance models, e.g., the Green Design Advisor (GDA).

Based on a review of the engineering design software and tools in Motorola's development environments, the following issues were identified:

— Engineering software tools are usually very complicated:
  - They solve complex engineering problems in great detail;
  - They require very detailed input, or a large data set, in order to solve the problem;
  - They are expensive with respect to license, administration, and maintenance costs;
  - They are not designed for a layperson to use. Even a trained engineer might find them difficult to use.

— Engineering software tools are usually developed by someone other than the end user:
  - They typically address a single engineering domain, such as component layout, DTM processing, manufacturing cost estimation, environmental assessment, etc.;
  - They typically do not communicate with other tools;
  - They typically require frequent software updates. However, the tools cannot or should not be updated by the end-users because of licensing, domain expertise, resources, and/or economic reasons.

— It is not practical to:
  - Install all the tools on every design engineer's computer;
  - Ask the engineer himself/herself to exchange data between the tools;
  - Ask the engineer to master all the tools;
  - Ask the engineer to understand all the other engineering domains outside of the area of her/his expertise.

In addition to the above issues with tools, internal benchmarks also highlighted four issues, occurring to a greater or lesser degree within certain organizations' product development processes, which contributed to longer-than-necessary cycle times:

— Inadequately defined product requirements.
— Insufficient or inappropriate design reuse.
— Design conflicts between different subsystems.
— Late identification of design issues, e.g., in Accelerated Lifecycle Testing (ALT).

2.2. USER INTERFACES FOR PRODUCT DEVELOPMENT TOOLS

One of the goals of engineering tools is to increase the level of automation of design activities. Nevertheless, within current development environments, human designers still make important decisions, and will continue to do so in next-generation tools. Thus, the computer-human interface is a critical part of any VP System.

A Graphical User Interface (GUI) for VP must support and exploit the distributed nature of the decision-making processes for engineering and design. Consequently, the GUI must facilitate both hierarchical and concurrent decision-making. This can be accomplished through a GUI that provides ordered interactions with a hierarchical view of the entire design. In general, there are five stages of interaction:

— Specifying the Design Domains and Level of Detail
— Selecting Analyses Based on Available Data
— Viewing the Computed Design Metrics
— Identifying Design Opportunities
— Guiding Product Management

An analysis of design processes at Motorola and other companies identified three principal failure modes in decision-making: Wrong Knowledge, Missing Knowledge, and Fundamentally Incompatible Knowledge. Correspondingly, a VP system and its GUI need to support human decision-makers with respect to all three of these modes for each of the five stages of interaction (as listed above).

2.3. CHALLENGES OF DESIGNING ELECTRONIC PRODUCTS

An engineering director once inquired, "If it is possible to develop a digital mock-up of a whole airplane, which includes a phone on each seat, why is it so difficult to develop a full digital mock-up of an entire mobile phone?" The answer lies in:

(1) the types of engineering design domains represented in the mock-up,
(2) the coupling between these domains,
(3) the level of detail required for the model to meaningfully guide decisions.

Whereas an airplane has millions of parts, full digital-mockups are typically limited to the mechanical and geometric aspects of the design. In contrast, mobile phones have significantly fewer components, but they have a high degree of interaction, due to the physics of electromagnetic energy propagation and the nearly infinite possibilities of software-defined functionality.

Designing and developing handheld electronic devices such as mobile phones is a process that requires engineering expertise from multiple domains, e.g., mechanical, electrical and radio frequency (RF). The Motorola Labs team found that separate groups of experts are typically responsible for separate portions of the product design for each domain. Decisions are made in each domain that can affect other design domains, and potentially result in design conflicts, leading to multiple prototype builds. Stories of complex, multidisciplinary design trade-offs include modifying the shape of a metal ornament on a mobile phone housing which was affecting the RF performance of the product's antenna.

Another example was the phone for which the clock frequency of the microprocessor interfered with the refresh of data on the Liquid Crystal Display (LCD). Mechanical examples include adjusting the placement of integrated circuits (ICs) on the keypad driver circuit board, to avoid excessive deflection from keypad actuation.

2.4. DESIGN GOALS FOR THE VP SYSTEM

As discussed earlier in this paper, Virtual Prototyping (VP) is a strategy for faster time to market by enabling product development with fewer prototype builds. Motorola Labs engineers took a tactical approach to VP by addressing how to enhance collaboration among engineering tools for multiple design domains, perform multi-disciplinary optimization (MDO), and provide engineers with meaningful suggestions for how to improve product designs. Thus, the design goals for the VP System included:

— Making available to product designers, within their "native" design environments, i.e., CAD tools, the expert knowledge and simulation models that exist throughout the company.

— Providing knowledge to designers regarding how the decisions they make in their own design domain affect the rest of the product design.

— Developing and maintaining a scalable, plug-and-play network of simulation models via a distributed network of interfaces, services and infrastructure.

— Enabling designers to efficiently search significant portions of design spaces.

Figure 1 presents a logical view of a VP System, which includes domain-specific tools for design, simulation, and optimization. These are represented by the large ovals, with the titles: Electrical, Cost, Quality/Reliability, etc. In addition, the system includes models representing multidisciplinary performance trade-offs. These are depicted as small circles labeled K_i, and require the codification of tacit knowledge typically found in the heads of experts in an R&D organization (Tirpak, 2005). The third aspect of the system is the communication integration layer. The highlighted parts of Figure 1 depict the knowledge flows for one possible scenario. In this case, the designer utilizes a domain-specific tool for electrical design, as well as multidisciplinary performance trade-off models for component costs (K_2), quality/reliability (K_3), manufacturability (K_5), and mechanical design (K_6).

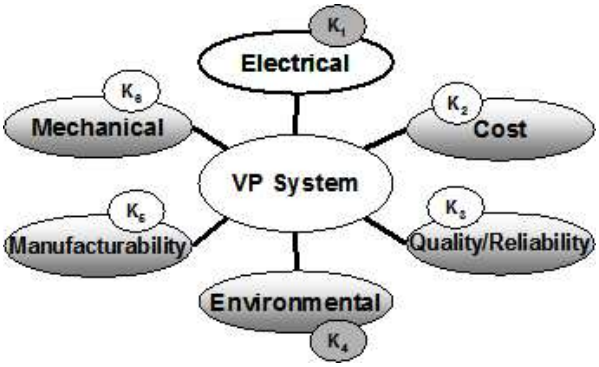

Fig. 1. Knowledge Management View of Virtual Prototyping

3. SYSTEM OVERVIEW

The Motorola Labs VP System is an application delivery system that hosts a network of models and servers for evaluating and optimizing product designs. It includes a total of about twenty models from the electrical, mechanical, board-layout, and other domains. Utilities for identifying optimal designs via Multidisciplinary Optimization (MDO) and Automated Concept Exploration (ACE) have also been implemented. These services accept inputs through the VP System client's Graphical User Interface (GUI) built from Extensible Markup Language (XML) templates. The VP System client can be launched as a Java application. A command-line interface also supports batch-mode execution of defined analysis scenarios.

The remainder of this section addresses the Common Object Request Broker Architecture (CORBA) which provides communication between the VP System server, the dedicated domain model servers, and the VP System client. Sections 4 and 5 provide examples of the VP System in operation.

3.1. DISTRIBUTED AGENT ARCHITECTURE

Figure 2 presents a high-level architectural view of the VP System in terms of its interfaces, infrastructure, and services. The depicted configuration involves five different computers connected via the Intranet. The VP System client software, which is installed on the designer's local computer, communicates with multiple VP Servers, each of which manages a set of services.

Model services are the most common type of service. They can include practically any type of computational model, representing design knowledge. Optimization services implement multiple algorithms to solve optimization problems whose formulation is encoded via the Optimization Interface Layer (OIL) for one or more model services. Database services, as their name implies, manage queries to Structured Query Language (SQL) databases. Expert system services implement a Rule Based System (RBS) that processes "facts" contained in not only the VP Scenario but also

the outputs from one or more model services. It should be noted that each of the services is called via a corresponding Java CORBA agent. It is possible to launch services that are local to the computer running the agent software, as well as services on remote machines.

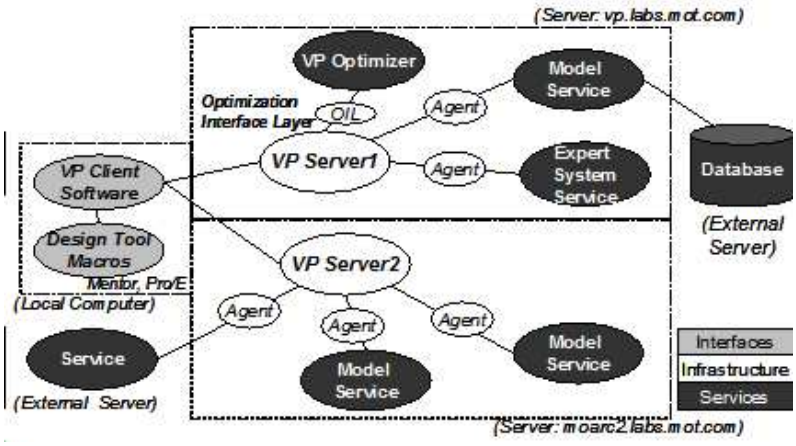

Fig. 2. VP System Architecture

Each VP Server implements an Action Manager, which interacts with the user via information, e.g., VP Scenarios, received from the VP client, and communicates with existing engineering domain services, such as the Green Design Advisor (GDA). Connectable software services are the building blocks of the VP System. The Java CORBA Agent framework provides connectivity. The elements of the agent defined by the Interface Definition Language (IDL) are:

```
module AgentApp {
  interface Callback {
    void agentCallback(in string sMsg);
  };

  interface Agent {
    string callAgent(in Callback callback, in string sMsg);
  };
};
```

The above agent IDL program states that the client-side program calls the server-side program through a standard method, i.e., callAgent; the server side program can use the agentCallback method to trigger a client-side event; and the server-side program returns the results of the operation.

The benefits of using an agent are:

— It provides a unified interface for all the engineering domain services;
— It provides a unified data representation format for all the services.

The VP System's data representation format has three elements: Item, ItemList, and ItemSet. An Item is a name-value pair with an extra pointer to an ItemSet. An ItemSet is a two-dimensional table that can be converted to an ItemList. An ItemList is a list of Items. The ItemList can convert itself to a string, or convert the string back to an ItemList. Software based on this type of data structure has the advantage of being able to change its data representation without any changes to the CORBA interface.

In the AgentClient class, the constructor AgentClient has been programmed to connect to the CORBA naming service through the Java-CORBA convention. The function connectToAgent has been programmed to find the server-side agent program with the Java-CORBA convention. The function callAgent can be used directly in its standard format to call the server-side agent by its agent name, or as the starting point for a more complex sequence of operations.

At the time the VP System architecture was defined, the Java language standard did not implement the CORBA Event Service; therefore, a CallbackListener class and its associated classes CallbackServant and CallbackHandler were designed to add server-side "event-push" to the agent framework, e.g., for monitoring the tasks being completed by an agent. The customized program to respond to server-side call-back events would start from the function callbackEvent.

3.2. EXAMPLES OF CLIENT AND SERVER PROGRAMS

Figure 3 depicts the class diagram of the server side and client side programs used to integrate one of the VP System's model services, namely, the printed wiring board cost estimation service.

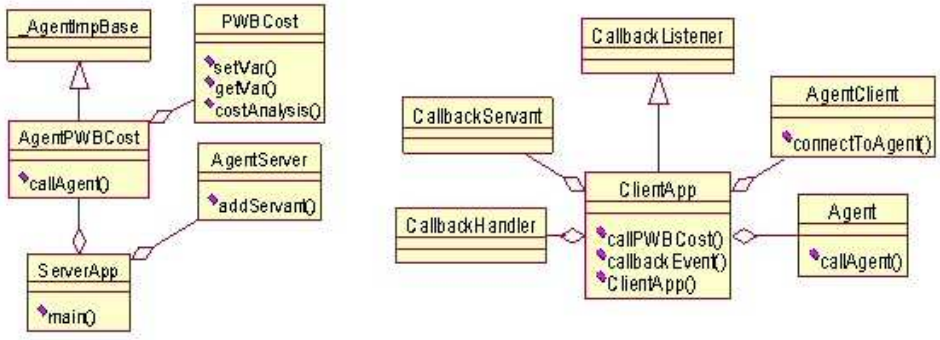

Fig. 3. Example VP Server and Client Class Diagrams

The PWBCost class uses CFA (finishing cost per area, $\$/m^2$), CLA (cost per layer per area, $\$/m^2$), CPTH (cost per plated through hole, \$), CTEST (cost of test, \$), D0 (defective density, $1/m^2$), NLAY (number of metal layers), NPTH (number of plated through holes), LB (length of PWB, m), and WB (width of PWB, m) to determine

the manufacturing cost of the PWB. The PWBCost class also calculates the cost sensitivities related to those variables at their current values.

3.3. BENEFITS OF THE JAVA CORBA ARCHITECTURE

The agent framework greatly simplifies the integration of software tools for various engineering domains within the VP system. Software tools can be developed, tested, and installed on different computers and locations by different groups. Then, the tools can be connected to the agent framework through the AgentServant, and be ready to provide services. Thus, model services can be developed collaboratively by engineering domain experts.

The agent framework reduces the degree of dependency of the software design on the application interface, which is typically a major concern in object-oriented programming. It provides a server-side event-push that can be used to write distributed event-message based applications. The event-push can also be used to send an acknowledgment to the client with intermediate results. This feature is very important for engineering software tool integration, since complex engineering analyses tend to run for quite some time, and their computational progress needs to be monitored.

The agent programs are compact, simple, and highly reusable. They rely only on the Java standard without other vendor-specific software dependencies. In summary, the framework provides an economical and unified solution to connect and reuse individual engineering domain software tools.

4. DECISION MODE I: "CALCULATOR"

This section of the paper discusses the operation of the VP System as an Intranet-accessible collection of design evaluation and optimization services, from which the user selects one or more services, inputs data to their GUIs, and runs them.

4.1. RUNNING A SINGLE DOMAIN MODEL

Figure 4 shows the Scenario Manager screen of the VP System, which is displayed immediately after a successful logon to the system. It should be noted that electrical and mechanical CAD files, if necessary for the model service(s) to be run, can be linked to the scenario via the Shared Files screen. If one wishes to run a single domain model service, one can access it by clicking on the service tree on the left side of the screen. The service tree can be configured via the Options screen, e.g., when one regularly runs a specific set of services and does not wish to see the other available services in the tree. Running VP Scenarios with multiple domain model services will be discussed in Section 4.2.

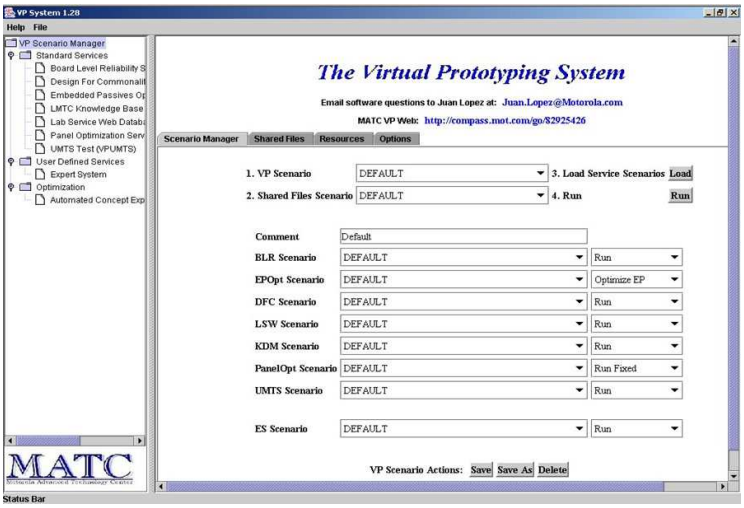

Fig. 4. Scenario Manager of the VP System

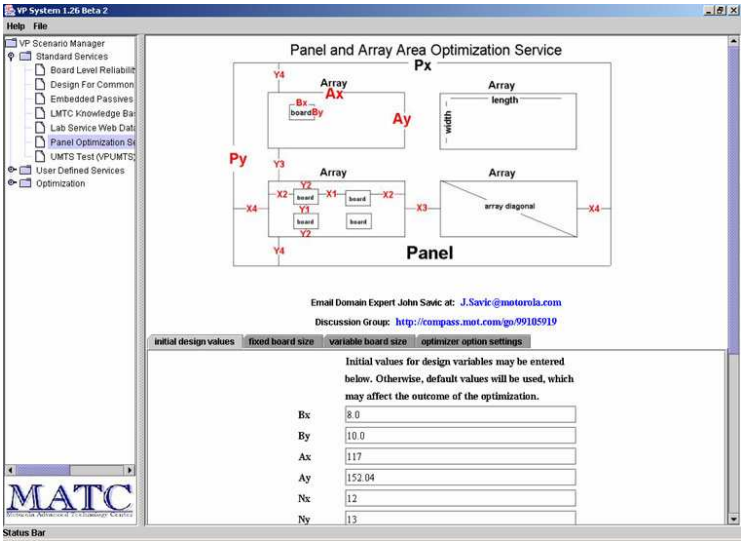

Fig. 5. Panel Optimization Service Screen

Figure 5 shows the main screen for the Panel Optimization Service. The user may modify the board size information and optimization settings, displayed in the edit-fields on this screen. One of the options is to allow the board dimension information from an electronic CAD file in Mentor neutral format, if one has been specified in the Shared Files portion of the VP Scenario, to override the values provided on the Panel Optimization Service screen.

Inputs for this service include: Fixed or Variable Printed Wiring Board (PWB) Size, Panel Size, Bounds on Array Sizes, and the Minimum Inter-Board Spacing. When the user clicks the "Run" button at the bottom of the Panel Optimization Service screen (below what is actually shown in Fig. 5), the service simultaneously optimizes the layout of PWBs within arrays, and the layout of arrays within a panel, so as to maximize the total number of PWBs per panel, thereby minimizing their manufacturing cost.

Figure 6 shows the service report screen, which lists the outputs, including: PWB Size, Configuration of PWBs within an Array, Configuration of Arrays within a Panel, and the Total Number of PWBs per Panel. For example, one of the modules that was optimized for Motorola's Mobile Devices Business, resulted in a 28.8% increase in boards per panel and potential 22.4% cost savings.

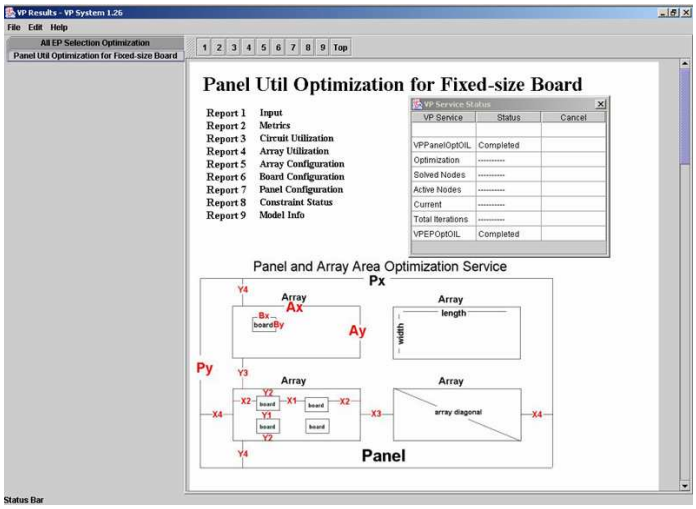

Fig. 6. Panel Optimization Service Report

4.2. RUNNING MULTIPLE DOMAIN MODELS

As previously mentioned, it is possible to run more than one model service at a time. This can be accomplished by defining a VP Scenario, as shown in Figure 4. The VP Scenario includes the Shared Files, e.g., CAD file, the scenarios (included the GUI-based inputs) for the individual model services, and their respective run modes. The sequence of steps is as follows:

— Select model services
— Load shared files
— Configure model services
— Save model scenarios
— Save VP scenario
— Run VP scenario
— View collection of evaluation reports

There are two main benefits of running an entire VP Scenario rather than a single service. First is that the user can easily run a "standard" set of analyses, e.g., for manufacturability. For example, this may consist of running four services for: estimating part cost, calculating commonality metrics with respect to a platform design, identifying any parts that have had issues in the manufacturing facilities, and identifying any parts for which mechanical lab test results are available.

The second benefit is that multiple, dependent attributes of a design can be evaluated and/or optimized. For example, one may wish to use the Embedded Passives Optimization service to identify the parts that should be realized as embedded devices, rather than discrete components, and then use the Panel Layout Optimization service to determine the layout of panels and arrays with sufficient area for the discrete components. In this case, the VP Server follows its precedence rules for which service to launch first, and which services must complete their runs before other services can be launched. It should be noted that the Expert System Service is launched only after all other model services have successfully completed. The small window located in the upper-right part of Figure 6 displays the current status of each service.

5. DECISION MODE II: "SPELL-CHECKER"

This section of the paper discusses the operation of the VP System as part of a designer's work within the "native environment", e.g., Mentor or Cadence for electronic CAD, and Pro/E for mechanical CAD. The use case for this mode of decision support mirrors that of spell-checkers commonly available within word processing programs. It is envisioned that as a designer is completing a design, he/she would periodically check the performance of the design with respect to the knowledge available in the VP System's model services. Likewise, at specified milestones in a product's lifecycle, e.g., "solution lockdown" and "manufacturing introduction", it would be necessary to verify that a design meets certain performance criteria.

5.1. LAUNCHING THE VP SYSTEM

Using the macro languages available in Pro/Engineer and Mentor Board Station, simple macros have been created to launch the VP System to evaluate the CAD file with which the designer is currently working. The macro first exports the working design to an open, text-based format, i.e., as the Standard for the Exchange of Product Model Data (STEP), and saves it to a temporary folder within the directory structure of the VP System installation on the designer's local computer. The macro then launches the VP System in batch-mode to evaluate the STEP file using a predefined VP Scenario, e.g., "MENTORSTEP (AJL068)", which specifies the model services to be run.

Thanks to the macros, launching the VP System to evaluate Mentor and Pro/E designs is as simple as typing the key sequence "vp()" from the user's keyboard, while working in the CAD tool environment. Different macros may be defined to launch a different VP Scenario for each type of analysis milestone, e.g., "solution

lockdown", etc. For each scenario, a set of model services is run, and the VP Expert System subsequently uses its rule set to compare the output metrics from multiple model services against established performance thresholds for the design. It should be noted that although the RBS is very easy to use, its creation requires considerable knowledge of the product requirements in order to define the conditions for "Errors", "Warnings", and "Suggestions", and to author corresponding explanations.

5.2. REVIEWING THE EXPERT SYSTEM REPORT

Figure 7 shows the top portion of the report from an Expert System Service which has been implemented according to the VP System's conventions for "spell-checking a design". The full report contains the number of "Errors", "Warnings", and "Suggestions", as well as brief messages with advice regarding what to change in the design. It is clear that this design "passed the test" since its status is "OK". The VP System's Report Screen provides the Expert System Service report along with the individual reports for all the constituent model services.

Metrics

Tag	Value	Value2	Value3
NumErrors	x	x	0
STATUS	x	x	OK
NumSuggestions	x	x	3
NumWarnings	x	x	1

Fig. 7. VP Expert System Design Check

The following is an example of how engineers used the "spell-checker" feature. The VP System was launched via macro from within Mentor Board Station and run for two model services. For each part number in the bill-of-materials (BOM), potential quality issues were identified in a database of more than 850 reports on materials, ALT, and other tests performed by an analytical lab in Motorola's Tianjin, China factory. Likewise, the mentor neutral files were automatically loaded into the Embedded Passives Device Selection service, and recommendations were generated regarding additional parts that could be realized as embedded parts, rather than discrete components.

6. CONCLUDING REMARKS

Virtual Prototyping (VP) is a data-driven design process that promotes both knowledge reuse and innovation. High-profile applications to-date in the automotive and aerospace industries have demonstrated its potential to significantly reduce prototype cycles, time to market, and total product cost. Thus, it will play an increasingly more

important role in digital product development and Advanced Engineering Environments (AEEs).

This paper has addressed VP as a specialized application of Decision-Support Systems, and discussed common requirements for engineering design tools, as well as requirements specific to the design of electronic products, such as mobile phones. Motorola Labs' test bed for VP was introduced in terms of its open, agent-based architecture utilizing Java CORBA. One of the key principles of the VP System is the reuse of expert knowledge across multiple engineering domains. This was highlighted via several use cases, showing that the system can function not only as an Intranet-accessible repository of model services but also as an integral part of decision-making within the native CAD environment.

Using the VP System, the Motorola Labs team has completed a variety of preliminary design analyses for products, such as mobile phones and base stations. Typically, the best results were for focused domain applications, such as Embedded Passives (EP) module design, where the VP team worked directly with Motorola's EP experts to develop and apply models for simulation and optimization.

The key to design cycle time reduction lies not only in the tools but in the processes. Thus, the lessons from the VP System are guiding further work towards a One Pass to Production Process, with an aggressive cycle time goal for transforming product requirements into manufacturable designs. In other words, the first physical "prototype" is the actual product for the customer. Primary inputs include: product requirements, design history, and validated components (platform/reuse library). Primary outputs include: CAD files, a DMU for visualization, a bill of materials with costs, and a description of the design functionality/behavior. Controls include: Project Management methodology and assigned engineering resources. In this scenario, one would naturally validate the design with detailed simulations, i.e., Virtual Prototypes. Fundamental questions related to the multiple dimensions of workflows, engineering analysis, and knowledge sharing, are being addressed by research in the field of Computer Supported Cooperative Work (CSCW).

Acknowledgments

The authors would like to thank Jun Ma, an intern from Northwestern University, who worked on the VP System for four summers. Thanks also to the Motorola engineers who collaborated on design domain model service development and applications, especially Robert Croswell and John Savic, and to the managers who supported this project, including Iwona Turlik, Tom Babin, and Rupin Javeri.

REFERENCES

1. P. Belanovic, M. Holzer, B. Knerr, M. Rupp, G. Sauzon, *Automatic Generation of Virtual Prototypes,* Proc. of 15th IEEE Intl.Workshop on Rapid System Prototyping (RSP'04), 2004.

2. F. Danesi, N. Gardan, Y. Gardan, *Collaborative design: from concept to application,* Proc. of Geometric Modeling and Imaging – New Trends (GMAI'06). 2006.

3. M. Durstewitz, B. Kiefner, R. Kueke, H. Putkonen, P. Repo, T. Tuikka, *Virtual Collaboration Environment for Aircraft Design.* Proc. of the Sixth Intl. Conf. on Information Visualisation (IV'02). 2002.

4. B.H. Li et al., *The Recent Research on Virtual Prototyping Engineering for Complex Products.* Proc. of 8^{th} Intl. Conf. Computer Supported Cooperative Work in Design. pp 18–25. 2004.

5. C.A. Murphy, T.D. Perera, *The Definition and Potential Role of Simulation within an Aerospace Company.* Proc. of 2001 Winter Simulation Conf., pp 829–837. 2001.

6. National Research Council, *Retooling Manufacturing: Bridging Design, Materials, and Production*. The National Academies Press, Washington, DC. 2004.

7. S.W. Park, J.K. Lee, B.C. Shin, Introduction to Development of e-Engineering Framework for the Automotive Module Design. *Proc. of 12^{th} Intl. Conf. on Computer Supported Cooperative Work in Design*, pp 146–151. 2005.

8. Sandborn, B., *Virtual Prototyping Printed Circuit Boards,* The Board Authority, Dec. 2000, pp 6–10. 2000.

9. A. Selko, K. Versprille, T. Vanderhoof, R. Strand, 2006. *Deliver Innovative Products: Knowledge-Driven Digital Product Development,* Industry Week Webcast. Oct. 31, 2006.

10. S.H. Thomke, *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation.* Harvard Business School Press. Boston, MA. 2003.

11. T.M. Tirpak, P.K. Mohapatra, P.C. Nelson, R.R. Rajbhandari, A Generic Classification and Object-Oriented Simulation Toolkit for SMT Assembly Equipment. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 32 (2002), 1, 104-122.

12. T.M. Tirpak, Five Steps to Effective Knowledge Management. *Research-Technology Management.* 48 (2005), 3, 15–16.