UNIVERSITY OF ZIELONA GÓRA
Faculty of Management
and Faculty of Mechanical Engineering
In association with the Design Society

FOURTH INTERNATIONAL SEMINAR AND WORKSHOP
7th - 9th October 2004
Zielona Góra - Rydzyna
POLAND

Engineering Design in Integrated Product Development
Management of Design Complexity

COMPLEXITY MANAGEMENT IN THE DESIGN OF DISTRIBUTED MANUFACTURING EXECUTION SYSTEMS

M. JENKO
Department of Control and Manufacturing Systems
University of Ljubljana
Slovenia
marjan.jenko@fs.uni-lj.si

Keywords: component-built, evolutionary, distributed manufacturing modeling, emergent response, low-bandwidth communication, manufacturing execution system, mutation, queuing model, reusable components, software reuse, statistical simulation, work system

***Abstract**: A Manufacturing Execution System (MES), an Enterprise Resource Planning (ERP) system and a Supervisory Control and Data Acquisition (SCADA) system make up the information and decision-making infrastructure of a modern company.*

Collaboration between SCADA and ERP system historically depended on the individuals who served as a link between production and management. MES enhances the collaboration between managing and manufacturing functions. MES adds new functionality to production management, and adds refinement to ERP and SCADA interfaces.

MESs are traditionally built in client-server configurations. This paper presents a new concept in a MES design and presents an extension of MES functionality. The proposed concept addresses, first, the distributed nature of modern manufacturing and second, the need for mutations in manufacturing. The extension in functionality is the inclusion of micro-planning, supported by statistical simulation, into the MES framework.

1. INTRODUCTION

The concept of using computer systems to assist in the production operation has been around since the early 1980's. Initially it was named Computer-Integrated Manufacturing (CIM), but as the scope of these systems increased and some standard products become available, the Manufacturing Execution System (MES) definition evolved. It is about a system that helps everyone in production *to execute the plan*. MESs range in scope but at minimum they all make production information available in real-time to all those who need it.

A modern MES implementation is technically a link between the real time, event driven shop floor control systems and the transaction order driven higher level system, i.e., the ERP system, Figure 1. MES guides, initiates, responds to, and reports on production floor activities as they occur. The resulting rapid response to changing conditions, coupled with a focus on production efficiency and quality, drives value-added production floor operations and processes. MES provides a level of detail and real-time control that is impossible to achieve with an ERP system only. Built-in real-time micro-scheduling minimizes non value-added activities. It is geared to keeping optimal work-loads on machines at all times and thus helps to achieve high production throughputs.

An instantaneous implementation of a purchased MES is not possible. A plant-wide MES can not be purchased as a finished product. It must be built to the requirements of each facility. Such a system is traditionally expanding its functionality over a period of years, incorporating existing equipment, and being modified to integrate new equipment as and when the equipment becomes installed.

Properly designed and maintained, MES provides a distributed database, sensors for tracking production processes, structures for decision-support and communication, that all collaborate in providing data in a secure and timely fashion to all major functional

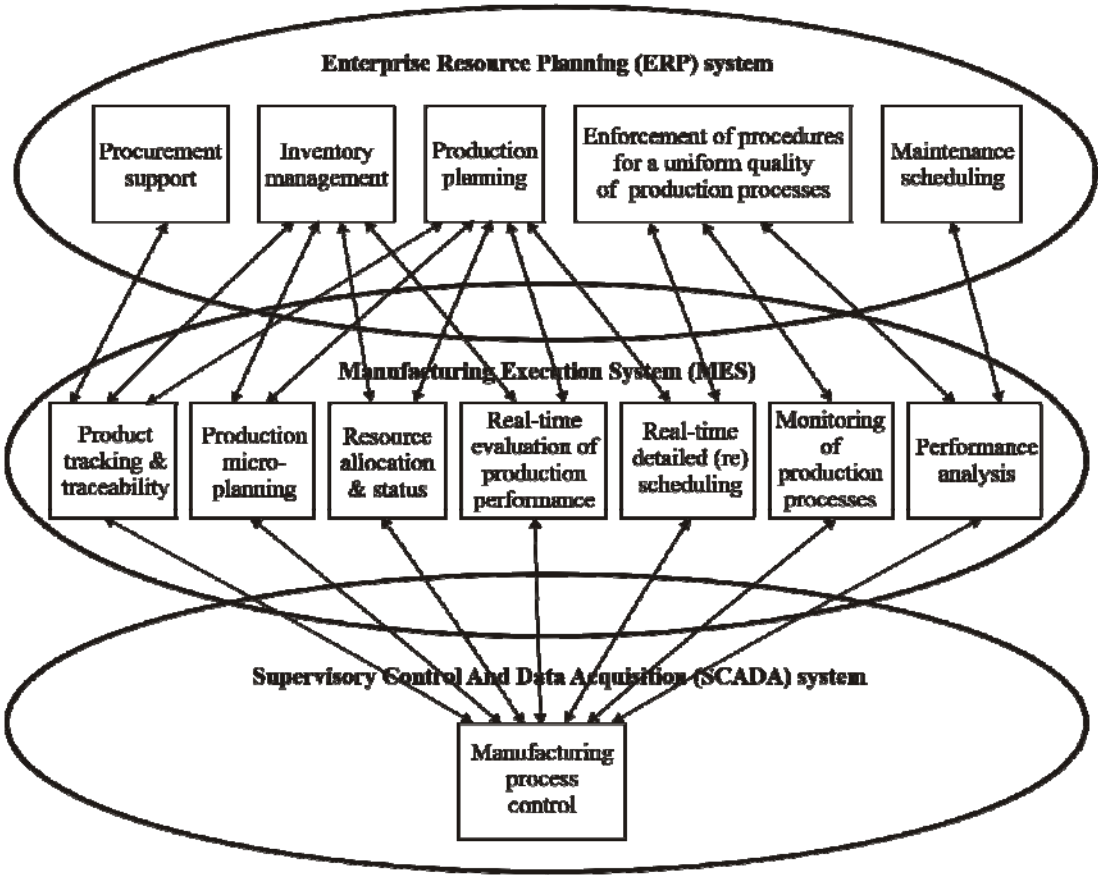

Figure 1 *MES is a link between the ERP and shop floor control systems*

areas, i.e., to process control, production management, commercial and technical applications.

There are several MESs available on the market, especially for the high volume production where only a state-of-the art organization of production gives a competitive edge.

MES solutions for a geographically distributed production environment are at a less mature stage. Relevance of standardized interfaces and protocols for building a distributed MES is recognized as important. Beyond that it is up to the designers of a particular system to build working solutions. This paper presents an ICT framework, that, when adhered to, yields an *evolutive distributed* MES. The proposed framework for a MES design is an answer to challenges of modern adaptive distributed manufacturing, which is, by nature, loosely coupled and flexible. Each player in the adaptive manufacturing value chains needs to be prepared to deal with a changing mix of trading partners, depending on which virtual product coalition they are participating in. Proposed MES framework addresses the issue of *change management* in a MES for a distributed manufacturing via sets of functional components that are stored in a components repository and autonomously distributed to different geographic places as needed.

Production planning is usually understood as a work of a deterministic nature. This work is traditionally performed by tools, built into ERP systems, that use data from a database to sum-up various production, storage, transport times and costs. These are than displayed in Gantt charts for different sequences of manufacturing activities. Experts are needed to schedule activities of different production orders that are executed simultaneously on shared manufacturing resources. The proposed framework first, introduces micro-planning on a production floor and second, addresses the non-deterministic nature of production processes with help of statistical simulations of production activities.

2. BACKGROUND

2.1. Modern distributed manufacturing systems

Several paradigms for the analysis and synthesis of modern manufacturing, i.e., of modern means of production, relocations, and storage have emerged in the last decades. The best known are holonic [1], fractal factory [2], bionic [3], and Complex Adaptive Manufacturing Systems (CAMS) [4]. All of these paradigms achieve their objectives by the adaptation

of the manufacturing systems to ever-changing requirements. It is in the parting scheme and in the decision-making processes that they differ mostly from each other. Progress in refining decision-making processes is being reported steadily.

2.2. Distributed Monitoring and Control Systems

The art of distributed software design is in the structuring of the application into self-sufficient entities with well-defined interfaces. Each entity itself is a small application. A good partitioning strategy enables writing software of great complexity. Actually, C++, which is the mother of all object-oriented languages, was introduced for solving one of the most difficult distributed problems in the fifties -- the optimization of a telephone switching network. AT&T founded this work, which produced the C++ compiler as a by-product [5]. Such a network has a close resemblance to a distributed manufacturing environment. Both systems consist of self-sufficient entities (switches in networks and different processes in manufacturing).

Efforts founded by the Advanced Research Project Association (ARPA) resulted in the creation of the Internet. This opened the potential for, first, effective information dissemination, and, second, for a huge upgrade from object-built applications to distributed object-built applications. New distributed programming technologies emerged in the last decade. The relevant ones for application programming are: Component Object Request Broker Architecture (CORBA) [6, 7, 8, 9, 10]; Distributed Component Object Model (DCOM) with its derivative COM+, and .NET. Distributed programming technologies use communication technologies, built into Operating Systems (OS), and internet infrastructure (routers, firewalls, interconnecting hardware).

2.3. Statistical simulations in manufacturing

Statistical, as opposed to deterministic simulation, takes into account scheduled activities and random events, calling for statistical modeling, such as late deliveries of material, customers' modifications of orders, machine breakdowns, sickness, etc. Can such modeling significantly help to improve manufacturing practices?

Let us put manufacturing activities into a perspective on a large scale of Newtonian mechanics. Biology or sociology might also be instructive, but are not within the author's competence.

At one end of the scale is the *fundamental* level of the theory. Here, one studies, for example, the motions of a few mass points interacting with each other within well defined initial and boundary conditions. The problems may be technically difficult, but they are well defined and solvable, at least numerically.

At the other extreme is the *statistical* level. Here, the number of particles is so large (approaching Avogadro's number) that the motions of individual particles can no longer be followed - nor would there be a justification or possibility to do it. Only some statistical concepts adjusted to respect conservation of energy remain. These concepts are temperature, entropy, pressure, etc.. The net effect is that classical mechanics, by forgetting individual components and concentration on their statistical behavior, gives rise to thermodynamics - a totally new paradigm.

The first, fundamental level is concerned with individual elementary systems free of random behavior; and the second, statistical level ignores completely the individuality of the parts to concentrate on their collective effects. Manufacturing belongs to an *intermediate* level, where both individuality and statistics are important. At this level, the number of items in production and machines is large enough to justify statistics, but not large enough to render the individual pieces irrelevant. Hence, their individual properties as well as their interactions modified by random events must be considered simultaneously. Just about the only practical approach available for the analysis of such complexity is computer simulation.

A structure of simulator's internal design matches a structure of a manufacturing process. Discrete event type simulation technique and dynamic build-up of objects, which correspond to machines and items in production, are the software design approaches that result in mapping of manufacturing activities into simulation. Statistical behavior is introduced on a level of an object, i.e., for machines, operators and items in production. All these are described by parameters with different statistical distributions [11].

Some commercial tools exist for simulation of manufacturing activities. Arena is the most popular one. However, it is aimed at simulation at large scale such as supply chains, logistics applications and improvement of business processes.

2.4. Reusable Software Components

Object oriented compilers are a result of tedious work. These compilers give means to materialize the encapsulation concept that is fundamental in the design of a reusable component software. Object Management Architecture (OMA) [12], and open system architecture for CIM (CIMOSA) have been suggested in the last decade to facilitate the static integration of reusable components. Reuse-centered developments have been performed in the design of production control systems [13, 14, 15, 16, 17] and in business systems [18]. Research in ontology for

designing embedded system applications from reusable components are reported by Jenko et al. [19].

Reusable software components are used in a form of source code components and in a form of compiled components. Source code components give the freedom to trim functionality easily and to switch easily among target platforms. Compiled components are used in a process of software linking. Both source code components and compiled components are supposed to be built into an application before its first usage.

Components prepared for dynamic linking (inclusion on demand when the application is running) are not yet very common. Operating systems take advantage of such locally available components to use fewer hardware resources at a time. Programming specialists, with such components, can maintain and upgrade real-time systems that need to operate in a non-disruptive manner.

A dynamic inclusion of remote (as opposed to local) software components is presented in this paper as means to build a practical MES that supports distributed manufacturing in the open environment. In our particular case, it is distributed manufacturing of different electronics modules that is monitored and controlled with help of the intranet and internet environment.

3. EVOLUTIVE COMPONENT-BUILT MES DESIGN

The presented MES monitors and controls entities of a distributed manufacturing process.

Object oriented compilers, internet infrastructure and software technologies for collaboration of remote objects gave us the infrastructure to build such a MES.

Statistical simulation on a level of micro-planning on a production floor is built-in into the presented MES to improve understanding of different production scenarios, bottlenecks and to result in simulation-supported decisions.

Proposed formalization of a MES design in a form of reusable software components gives us means to built a MES that reacts to changes in a production environment, i.e., it has evolutive properties. This design is also scalable, since increased system complexity results only in bigger system and in increased information traffic, and not in increased complexity, i.e., in increased functionality of an elementary building block, which is a software component.

Complex Adaptive Manufacturing System (CAMS) paradigm [4, 20] gave us a concept for structuring a manufacturing process into self-sufficient entities. The CAMS paradigm divides a manufacturing

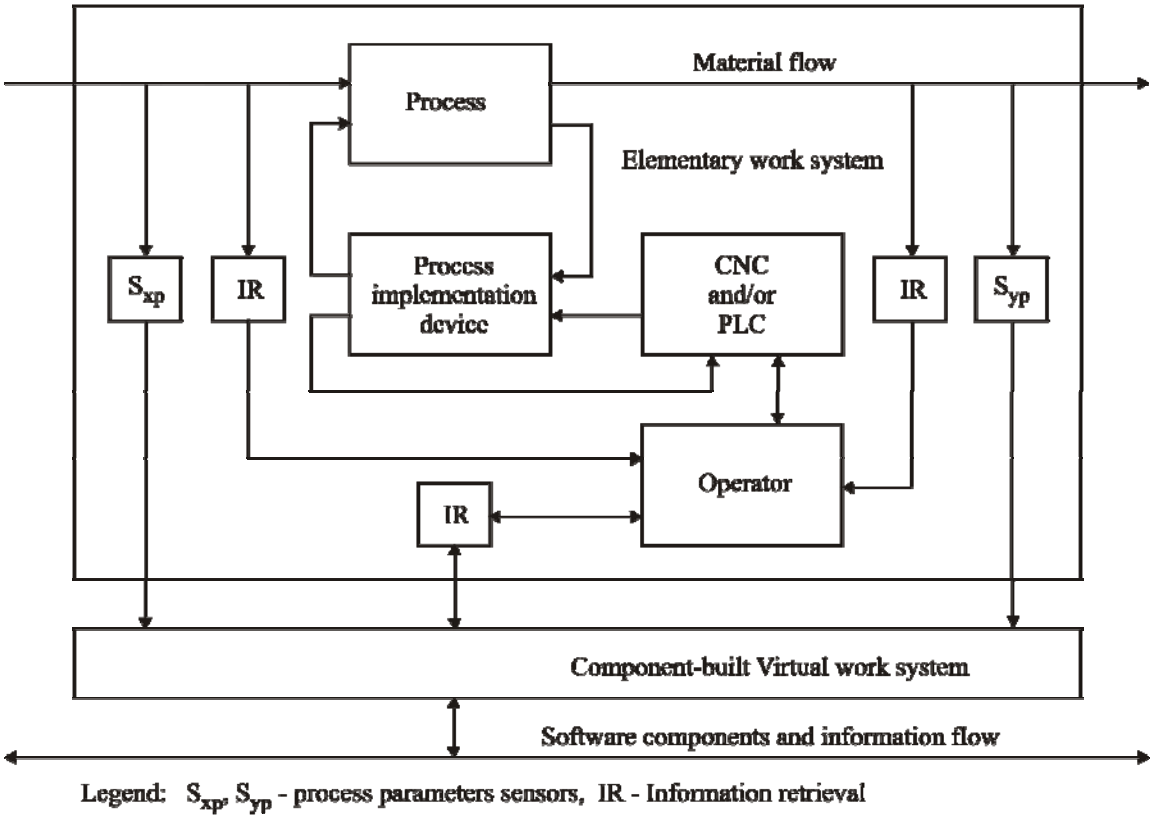

Figure 2 *Elementary work system in CAMS paradigm coupled to a component-built virtual work system*

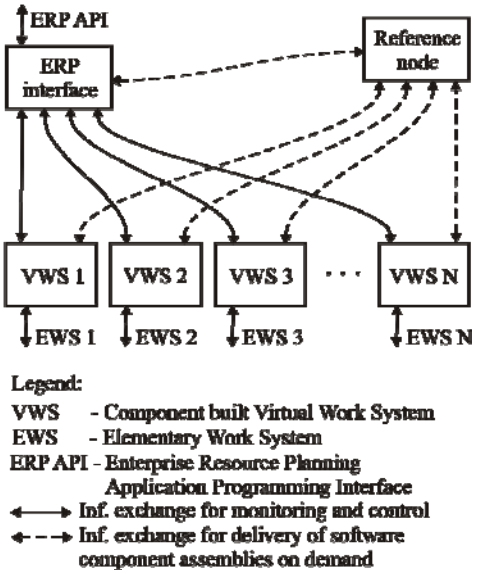

Figure 3 *Information exchange for monitoring, control and delivery of software components on demand*

process into elementary processes that can be analytically described. To implement an elementary process, CAMS paradigm introduces an Elementary Work System (EWS). EWS consists of a work process, of hardware elements that are sufficient to implement a work process, of a work process identification that enables process control and its optimization, and of a human operator, Figure 2 [4]. EWSs do have the capabilities and the competence to perform particular manufacturing operations. They are conceptualized as autonomous units and the indivisible building blocks of a manufacturing system. EWSs' representatives in the information world are Virtual Work Systems (VWSs). These were introduced as the progress in ICT was beginning to promise the possibility of effectively building such structures and inter-VWS information exchanges.

VWSs, EWSs representatives in the information world, are operating in computers that are wirelessly connected to the intra-net (embedded Linux and WindowsCE3.0 on rugged Tablet PCs and on embedded PCs).

3.1. MES evolution via emergent response of a VWS

VWSs are built from functionality components. The novelties in the proposed component-built distributed system for production monitoring and control are:

- a reference node, i.e., functionality components repository is introduced, Figure 3. Here, reference components are stored and maintained. The reference node is on a component-dedicated platform..

- functionality components are automatically downloaded from a reference node, installed into the VWS, and immediately used to boost or modify VWS functionality when needed.

Inter- VWS information exchange, and intra- VWS functionality, enables a distributed monitoring and control. And exchange of component requests for component assemblies enables VWS emergent response. Whenever an component-built VWS receives data that it is not equipped to process, it searches for a required functionality in the reference node, Figure 3. The component assembly, that encapsulates required functionality, is downloaded and installed into the VWS in the most unobtrusive manner. The VWS immediately uses the new functionality to produce a response to the request that triggered the whole activity. On a system level, such a component assembly exchange gives means for system evolution and mutations.

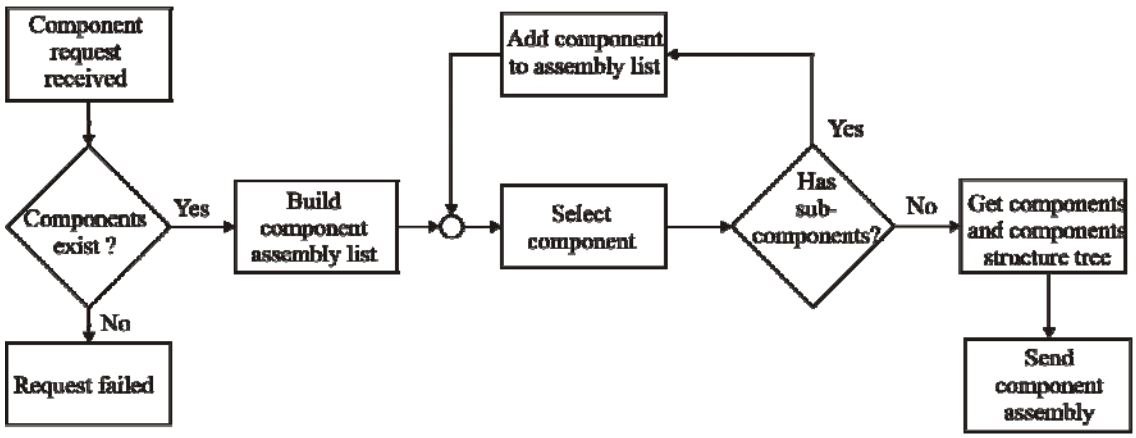

Figure 4 *Building-up a functional component assembly*

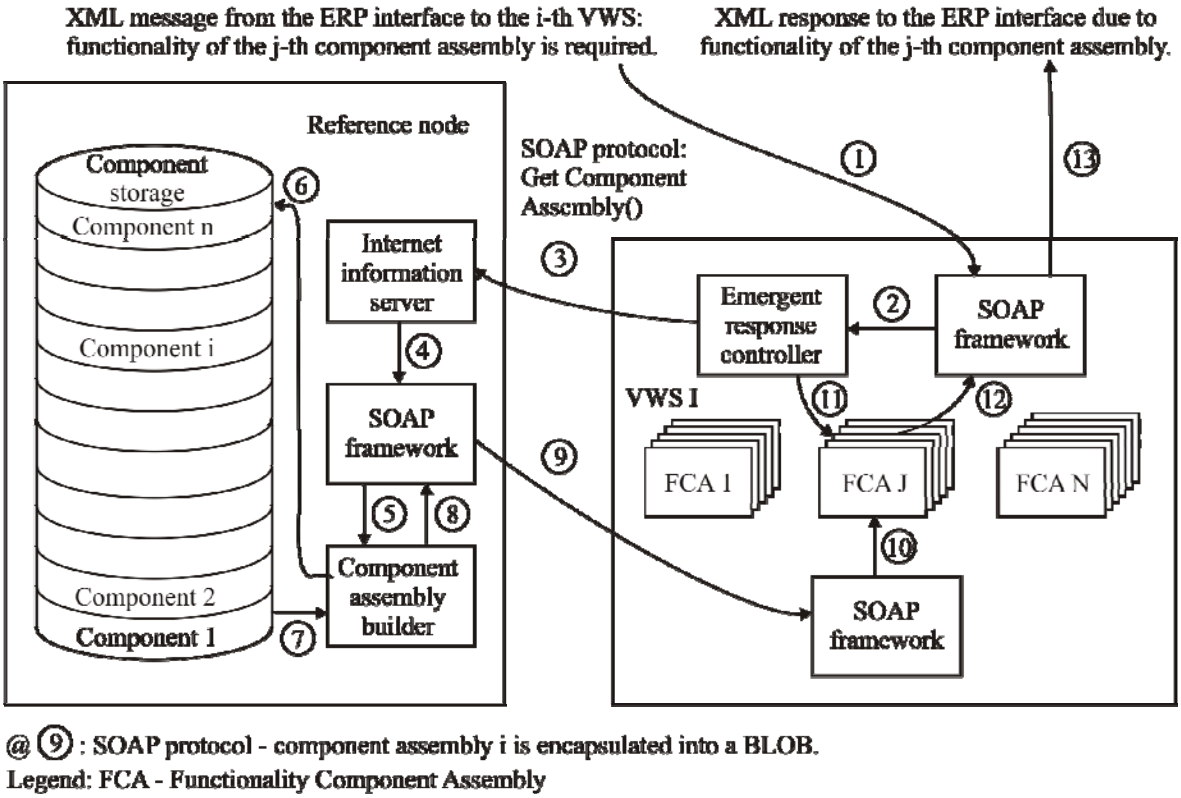

Figure 5 *Activities involved in an VWS emergent response*

Figure 4 presents a flowchart of activities that are involved in the creation of a component assembly in a reference node. First, a VWS sends a request for a new functionality to the reference node. It is a responsibility of software developers to simultaneously release component assemblies that send requests, and those that send replies to new requests. Therefore, the condition that a requested component would not be available should not be allowed to occur.

In the reference node, first a component assembly list is generated from information on components and on intra-component dependency. This information is stored in a database on components. Then, components from the assembly list are sequentially extracted from a database. They are packed together with the assembly list into an XML document. The data is sent to the VWS that requested new functionality, Figure 5. In this case, it is a Functionality Component Assembly J, that is needed in a VWS I to process the incoming message. The VWS unpacks the document from the reference node. Then, the VWS uses information from a dependency list to autonomously install inter-dependent components in a proper sequence. New

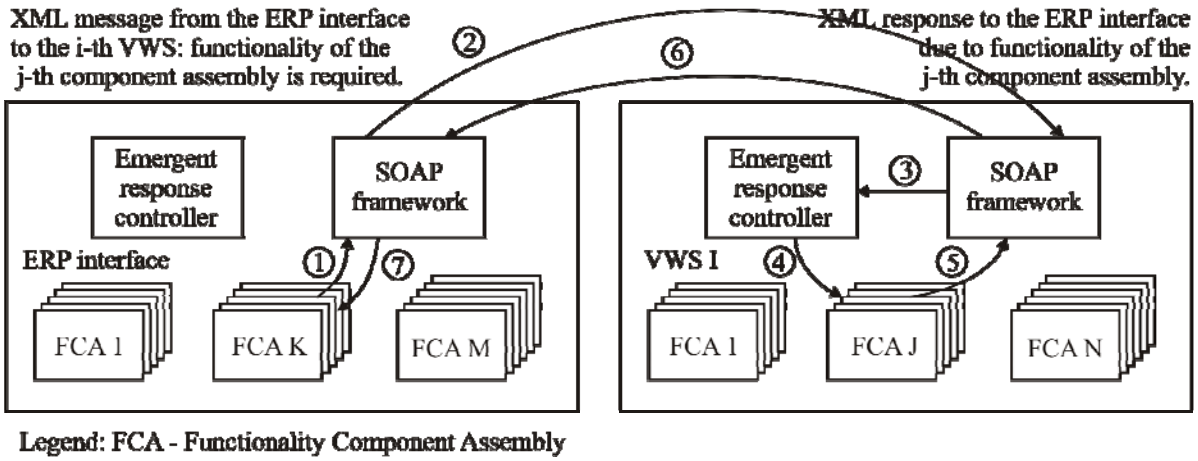

Figure 6 *Activities involved in the communication after the particular component assembly download*

functionality is installed and immediately available to process the message that triggered the whole activity of component assembly build-up, downloading and installation. Figure 6 shows subsequent information exchange with the involvement of Functionality Component Assembly J in the VWS I.

4. EVOLUTIVE COMPONENT-BUILT DISCRETE EVENT TYPE SIMULATOR FOR MICRO-PLANNING

In Taylor's paradigm, the "project" remains the same over a long period of time. For example, a fixed project might be "populate and solder a set of printed circuit boards (PCBs) and finally assemble the boards into an electronics module of a product A". The components are then continually being replenished from the outside, component placing and soldering stations continually produce populated PCBs, the kitting station continually replenishes the queue of PCBs, and the assembly EWS continually processes the PCBs into an electronic module according to the same processing instructions, or programs.

In adaptive manufacturing, however, the job orders and Work Flow Specifications (WFSs) change over short periods of time. For example, the projects might be "produce electronics module for 10000 units of a product A", followed by "produce electronics module for 2000 units of a product B", etc., with likely return later to the production of a module for a product A. Clearly, the input queues,

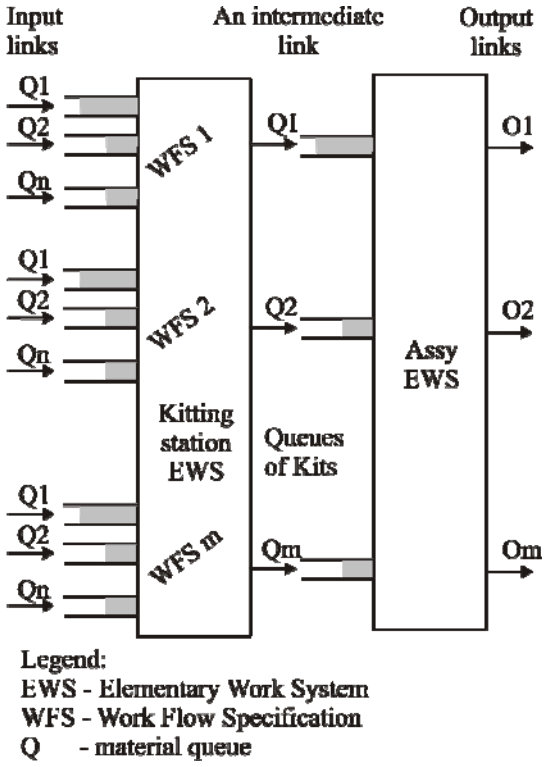

Figure 7 *Subsystem for the assembly of electronics modules*

kitting procedures and queues of PCBs are project-specific. Not only because the components and EWS programs are different, but the customers might be different too.

In such scenarios of adaptive manufacturing a statistical discrete event type simulation supports

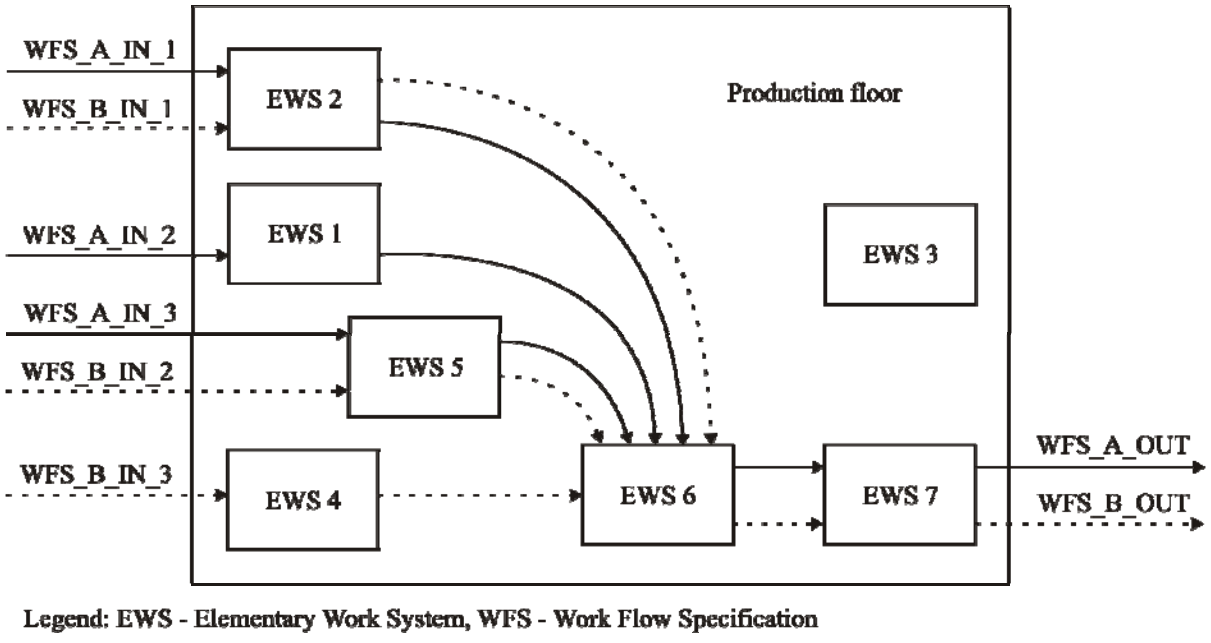

Figure 8 *Two WFS sharing the same resources*

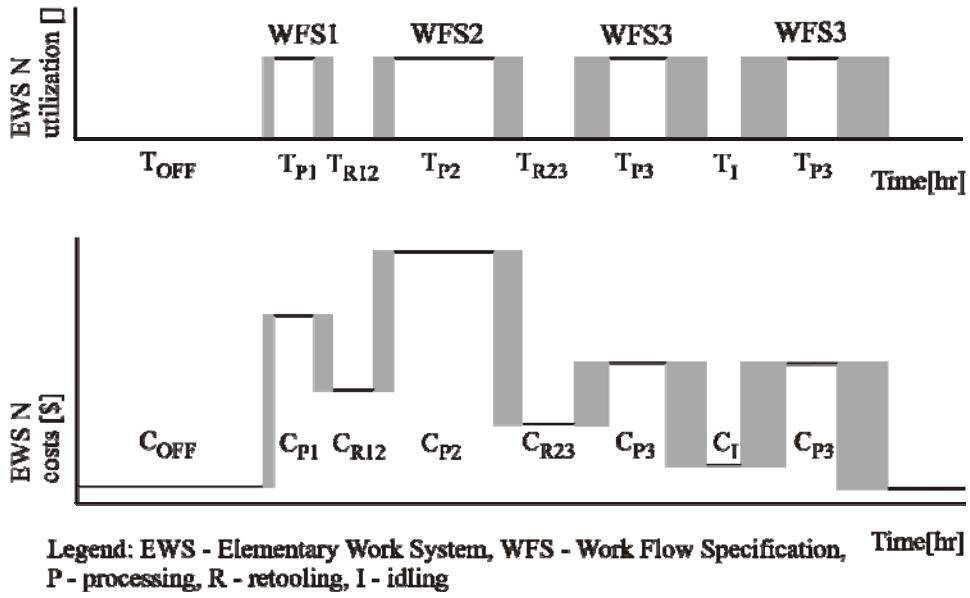

Figure 9 *Times and costs for an EWS N*

decisions in micro-planning. Let us define a production floor as a set of EWSs and links among them. EWSs correspond to different production phases, and links correspond to transport paths that material pieces need to cross between production phases. For example, the subsystem for the assembly of electronics modules is in Figure 7. The model for the production floor, consisting of EWSs and links where different job orders and WFSs take place simultaneously on a large scale is in Figure 8. Here, the material flow defined by the application A is represented by solid lines, the material flow defined by the application B by dashed lines. In this example, EWS 3 is not used by either application - but might be by other applications at other times. EWSs 1, 2, 3, 4, and 5 perform placing of electronics components onto a PCB and soldering them, EWS. 5 is a kitting station and EWS 6 performs the assembly of electronic modules from different PCBs.

Job orders are the actual inputs to the manufacturing system and WFSs define the production process.

The essential simulation outputs are distributions of production times and of production costs, such as in Figure 9. Statistical data, concerning the utilization of EWSs, help to discover production bottlenecks and resource underutilization.

5. IMPLEMENTATION

It took one engineer-year (EY) of research to evaluate combinations of available technologies. It took another two EYs of effort to develop and code a real world design for a distributed MES consisting of an component-built ERP interface and VWSs, and of a component-built built-in statistical simulator for a micro-planning support.

Such a scalable project for the implementation of a distributed application needs to effectively address security, ease of maintenance, system extensibility, system mutations, and net bandwidth preservation. These issues become most critical to the success of a project when a distributed application spans the intra- and internet which is the case with distributed manufacturing.

Regarding security, existing security policies must remain intact, i.e., firewall and router settings must not be changed in order to make a new distributed application work. No additional ports are to be opened. A combination of a SOAP service, XML language, and HTTPS gives means to preserve security, since this combination allows passing information through firewalls and routers that are set to allow only web browsing, that is, are set for common internet usage. HTTPS itself has built in mechanisms for authorization, authentication, and data encoding.

Component-built VWSs, structuring functionality components into sub-components, managing component inter-dependency with the help of dependency lists, usage of .NET technology for intra-agent components activation, and focusing system maintenance and system development on a reference node only, makes the implementation of the presented MES feasible. Particularly, since:

- VWSs maintain and evolve by themselves. Only the reference node needs human maintenance.

- VWS implementation complexity is reduced. Structuring VWSs into sets of logically simple inter-dependent components makes coding a profession, not an art.

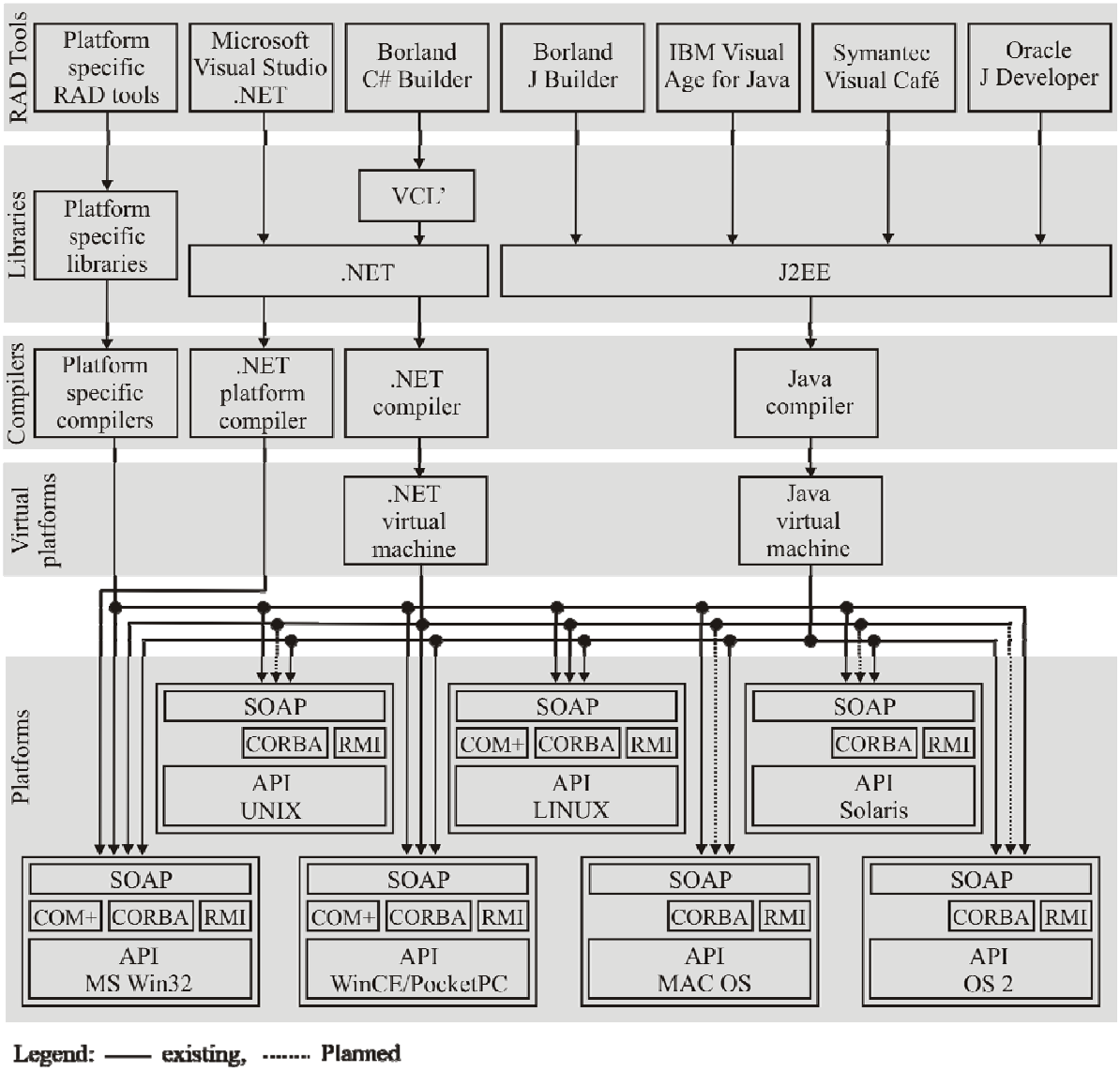

Figure 10 *Coupling of modern RAD tools, libraries, compilers, and platforms*

- New EWS inclusion is solved on a system level. An EWS, added to the production floor, is immediately equipped with a VWS, that is created from a template. The VWS then develops functionality through its life time.

- Net bandwidth is not wasted. New (or different) functionality is transported by internet only when needed, and only to the destination where it is needed.

- User's perception of MES responsiveness is practically the same in the case of VWS-emergent response, Figure 5, and in the case of normal communication to/from the VWS, Figure 6. The whole chorus of downloading a functionality component assembly, its registration, and activation takes less than a second on a distant Pocket PC connected to a reference node via a combination of intra-, and internet infrastructure. The limiting factor to response time is actual internet bandwidth. The bandwidth is sufficient when intra- nets are connected to a public infrastructure via cost-effective ADSLs.

- Intra- net security is not compromised. All information flow is performed via HTTPS protocol which includes its own security mechanisms. Firewall settings remain intact.

Figure 10 shows coupling of modern RAD tools, libraries, compilers and platforms that all can be used to build distributed evolutive component-built MESs.

6. CONCLUSION

Structuring monitoring and control processes for manufacturing into EWS-local entities, i.e., VWSs that communicate with the ERP interface, yields effective MES under static conditions. Structuring

VWSs into functionality component assemblies, that are instantly downloaded and installed on demand from a reference node, effectively addresses maintenance, and the evolution of a MES system. Structuring functionality components into sub-components, where components keep track of versioning and dependency lists, keep the VWS structure consistent. Statistical simulation on the production floor improves micro-planning. Proposed MES architecture addresses these implementation issues: ease of implementation, ease of maintenance, system security, capability of evolution, and preservation of net bandwidth. Modern technologies (SOAP/.NET, XML, and HTTPS) are being used to produce a MES that has the potential, by design, to grow, and the potential to evolve technically through the choice of the currently most powerful IC technologies.

REFERENCES

[1] Valckenaers, P., VanBrussel, H., Bongaerts, L., Wyns, J. "Holonic manufacturing systems", Integrated Computer Aided Engineering, vol. 4., no. 3, p. 191-201, 1997

[2] Warnecke, H.J., Die fraktale Fabrik, Revolution der Unternehmenskultur, Springer Verlag, 1993

[3] Ueda, K., "A concept for bionic manufacturing systems based on DNA-type information", Proceedings of 8th International Prolomat Conference, Tokyo, p. 853-864, 1992

[4] Peklenik, J. "Complexity in manufacturing systems", Manufacturing Systems, 24, p. 17-25, 1995

[5] Stroustrup B., The design and evolution of C++, Addison Wesley, 1995

[6] Huang C. Y., S. Y. Nof, Evaluation of agent-based manufacturing systems based on a parallel simulator, Computers & Industrial Engineering, vol. 43, p. 529 – 552, 2002

[7] Kim T. W., C. S. Ko, B. N. Kim, An agent-based framework for global purchasing and manufacturing in a shoe industry, vol 42, p. 495 – 506, 2002

[8] Kotak D., S. Wu, M. Fleetwood, H. Tamoto, Agent-based holonic design and operations environment for distributed manufacturing, Computers in Industry, vol 52, p. 95 – 108, 2003

[9] Shen J., S. Park, C. Ju, H. Cho, CORBA-based integration framework for distributed shop floor control, Computers & Industrial Engineering, vol 45, p. 457 – 474, 2003

[10] Yen B. P. C., Communication infrastructure in distributed scheduling. Computers & Industrial Engineering, vol. 42, p. 149 – 161, 2002

[11] Jenko M, Queuing simulation of distributed manufacturing systems, Proceedings of the International Conference on Flexible Automation and Intelligent Manufacturing (FAIM'01), Dublin, July 16-18, 2001

[12] Hutt A.T.F. (Ed.), OMG, Object Oriented Analysis and Design-Description of Methods, Wiley, New York, 1994

[13] Grabot B., P. Huguet, Reference models and object-oriented method for reuse in manufacturing control system design, Computers in Industry, vol. 32, p. 17 – 31, 1996

[14] Ryu K., Y. Son, M. Jung, Modeling and specifications of dynamic agents in fractal manufacturing systems, Computers in Industry, vol 52, p. 161 – 182, 2003

[15] Smith, J. S., S.B. Joshi, Reusable software concepts applied to the development of FMS control software, International Journal of Computer Integrated Manufacturing, vol. 5, p. 182 – 196, 1992

[16] Son Y. J., A. T. Jones, R. A. Wysk, Component based simulation modeling from neutral component libraries, Computers & Industrial Engineering, vol. 45, p. 141 – 165, 2003

[17] Verwijmeren M., Software component architecture in supply chain management, Computers in Industry, vol. 53, p. 165 – 178, 2004

[18] Worley J. H.,, G.R. Castillo, L. Geneste, B. Grabot, Adding decision support to workflow systems by reusable standard software components, Computers in Industry, vol. 49, p. 123 – 140, 2002

[19] Jenko M., N. Medjeral, P. Butala, Component-based software as a framework for concurrent design of programs and platforms – an industrial kitchen appliance embedded system, Microprocessors and Microsystems, vol. 25, p. 287 – 296, 2001

[20] Peklenik, J. (1992). "FMS - A complex object of control", Proceedings of 8th International Prolomat Conference, Tokyo, 1992, p. 1-25