Published in: Proceedings of the 10th annual INCOSE conference, July 2000, Minneapolis, USA

Enabling Changes in Systems throughout the Entire Life-Cycle – Key to Success ?

Armin P. Schulz, Ernst Fricke, Eduard Igenbergs

Division of Astronautics
Technical University of Munich
Boltzmannstrasse 15, 85748 Munich
Germany
A.Schulz@lrt.mw.tum.de, http:\\www.lrt.mw.tum.de

ABSTRACT

In the past decades the world has been changing in almost every aspect. Systems development is facing rapidly changing and increasingly global environments in markets, competition, technology, regulatory and societal systems. Systems to be delivered must be designed not only to meet customer or market needs, but increasingly to meet requirements and constraints of systems sharing its operational context and throughout their entire lifecycle. The design of a system must provide for a continuous evolution of its architecture either by upgrading a system already in service or releasing a new version.

Based on these key challenges imosed on development systems, this paper will evolve the idea of incorporating changeability into a systems architecture. Flexibility, agility, robustness, and adaptability as four key aspects of changeability will be defined and described. Design principles to enable flexibility, agility, robustness, and adaptability within integrated systems are proposed and described. A measurement of a degree of implementation of certain aspects of changeability as well as cost associated with it are addressed with metrics. A basic process outlining and guiding an application of the framework described concludes this paper. Examples from varying industries will illustrate the applicability and implementation of selected principles.

Thus this paper spans a view from **why**, **when** and **where**, **what**, and **how** changeability has to be incorporated into a systems architecture.

INTRODUCTION

During the past decades several global factors tremendously shifted our world to a more complex one in almost every aspect. Based on Wenzel et al. (1997) and Schulz & Fricke (1999), three aspects are major drivers of development systems within the future:

- Dynamic Marketplace
- Technological Evolution
- Variety of Environments

Dynamic Marketplace. A high number of new markets is emerging rapidly , while existing markets are changing. Staying ahead of competition requires high responsiveness in terms of supporting late design decisions of a system architecture to narrow down the time gap between design freeze and system delivery. The ability to address evolving customer and market needs by implementing late changes into a system architecture will affect a systems success throughout its entire lifecycle. That is architectures must not only incorporate the ability to be changed easily and rapidly within late design phases but also when already being in service. Additionally, an increasing trend towards individualization requests for an individualization of mass products. Therefore the degree of variety supported by standard product platforms and their derivatives becomes crucial for commercial success.

Automotive industry provides an example, where luxury car manufacturers statistically produce no car twice due to a high number of variants imposed by a wide range of equipment selected by the individual customer.

Technological Evolution. The fast evolution of all our systems (Ring & Fricke, 1998) driven by a half life of technologies significantly shorter than system life cycles[1] or even system development cycle times leads to further problems for system architectures. Functions of systems are evolving rapidly within a system life cycle, in terms of number and performance of available functions. Or as Iansiti (1998) puts it '... *many business environments no longer have a stable technology base, instead it is novel, changing rapidly and unpredictable* ...'. Therefore, steady insertion of new technologies is necessary to keep a system competitive. Processes, organizational structures, tools or methods also have to

[1] The gap between technology half-life and system lifecycle is in particular driven by an increasing percentage of system functionality being based on electronic and software

be adapted accordingly.

For example the wireless application protocol (WAP) has been recently introduced into the mobile telecommunication market to offer mobile internet access. Older cell phones do not support this protocol, but by installing a software update, short message service (SMS) may be used to achieve mobile internet access.

Variety of Environments. The composition of those systems increasingly relies on components based on rather diverse technologies and origin (i.e. mechanical and electrical hardware, embedded software, etc.). An integration of all those systems results in more complexity and integrity in all systems. Additionally, those complex systems are embedded into a higher system, that is they are part of a system of systems (e.g. a single satellite within a communication constellation, a car within cooperative traffic management, a cellular phone in varying national networks). As pointed out before also these system are subject to a dynamic marketplace and an increasing technological evolution. Since the dynamics of our economy is increasingly ruled by the logic of networks (Kelly, 1998) the elements of the overall system are highly interrelated and affect each other.

Thus each system is affected by changes and the evolution of its embedding system and/or the systems within its operational context. Hence system architectures also need to incorporate the ability to adopt towards changes within its environment.

Key Challenges. Eventually these three drivers of future system development lead to two key challenges which have to be met within the design of system architectures.

(1) system architectures have to incorporate the ability to be **changed easily** and **rapidly**
(2) system architectures have to incorporate the ability **to be insensitive** or **adaptable towards** changing **environments**

It is moreover necessary to reduce the time gap between freezing properties, functions, features and architectures of a product and the product market delivery to a absolute minimum. In a study on Toyota's development system Ward et al. (1995) come to the conclusion that delaying design decisions is a major success factor for Toyota's superior performance in development time and quality. Ward and his co-authors call that the 'Second Toyota Paradox'. However, this approach being recognized within Toyota's development is more related to the implementation of the system architecture design process, incorporating the principles introduced and will be elaborated in future work.

ARCHITECTING FOR SYSTEM EVOLUTION ?

Evolve Idea. The current practice is to prevent and front-load changes, being based on the experience that late changes – and therefore any evolution – of systems are very costly. This is described by the 'Rule of Ten' stating that with each subsequent program phase the implementation of a change becomes ten times more costly and therefore recommends to prevent or front-load changes (e.g. Boehm, 1981, Clark & Fujimoto, 1991). But in today's fast moving business environments this is simply not enough, as preventing change and evolution is a serious obstacle to technological evolution and endangers competitive advantage due to technological leadership. With regard to front-loading Wheelwright & Clark (1992) point out that even the best forecasters and most clever downstream engineers will encounter unexpected changes in design.

Hence, companies have to deliver systems incorporating an architecture which supports changes throughout its entire life-cycle. Being part of higher systems (system of systems), as well as being part of a human response system (markets, customers and their needs), the ability to adapt towards changing environments is a further requirement. This is supported by different other authors (e.g. Adaptive Systems Group 1998, Dove 1999, Iansiti, 1998, Kelly, 1998, Open Systems Joint Task Force, 1998). In particular Kelly (1998) pointed out that instability and imbalance are the norm in today's economy and therefore systems optimized to a single design point will not last very long. But not only delivered systems have to have those abilities. Companies itself need to incorporate these abilities within their entire development system (i.e. development processes, test and verification, manufacturing, etc.).

There is a set of distinguishing characteristics, for which type of system architectures an incorporation of changeability should be considered or should not be considered. Steiner (1998) introduced a set of distinguishing features for enduring architectures, which is in the context of system evolution comparable to the approach the authors have taken. Basically incorporating changeability within a system architecture is required for systems, which

- are subject to a dynamic (that is rapidly and strongly changing) marketplace with varying customer base and strong competition
- have a long lifecycle compared to cycle times of technologies driving major quality attributes (i.e functionality, performance, reliability, etc.)
- are highly interrelated with other systems sharing their operational context

- require high deployment and maintenance cost

SOLUTION APPROACH

Four Aspects of Changeability. As an answer to the challenges imposed on today's development system the approach 'Design for Changeability', which has been first proposed by Fricke (1999) is introduced. Basically four aspects of changeability are distinguished:

- Flexibility
- Agility
- Robustness
- Adaptability

These four aspects describe a systems ability to cope with changes within itself or its environment (Schulz & Fricke, 1999). A set of varying design principles is used to enable each aspect of changeability within architecture design. While some principles only support a single aspect, other principles support several aspects. A different mix of practices supports the implementation of the principles and is partially domain specific. Finally, a set of metrics is needed to evaluate and control the design of a system architecture regarding changeability.

Flexibility and Agility. *Flexibility* represents the property of a system to be changed easily, that is low effort and without undesired effects. *Agility* represents the property of a system to implement necessary changes rapidly. Flexibility is a prerequisite to achieve agility, i.e., agility is an evolutionary level of flexibility (Figure 2).

Robustness and Adaptability *Robustness* characterizes systems, which are not affected by changing environments, that is robust systems deliver their intended functionality under varying operating conditions without being changed. Taguchi (1993) and Clausing (1994) have performed extensive research in the area of robustness within systems. *Adaptability* characterizes a systems ability to adapt itself towards changing environments to deliver its intended functionality. Robustness is a prerequisite to achieve adaptability, i.e. adaptability is an evolutionary level of robustness (Figure 2).

Solution Framework. The Design Principles are enablers for the realization of changeability. Among the principles being identified as characterizing flexible, agile, robust, and adaptive systems, a distinction is made between *Basic Principles*, supporting all four aspects of changeability, and *Extending Principles*, supporting only selected aspects of changeability and also having interrelations (Schulz & Fricke, 1999). The matrix in Figure 2 indicates which principle is contributing to what extent to each aspect of changeability. Although examples presented in this paper are mainly covering product systems, the authors believe the proposed principles to be applicable to any type of system (i.e. processes, organizations, etc.). Whatever type of system is under analysis, the design of its architecture is based on a system's elements, their attributes in terms of functions and properties, and their (inter-) relations. The degree to which a single principle is incorporated within a system architecture may be measured using various types of metrics.

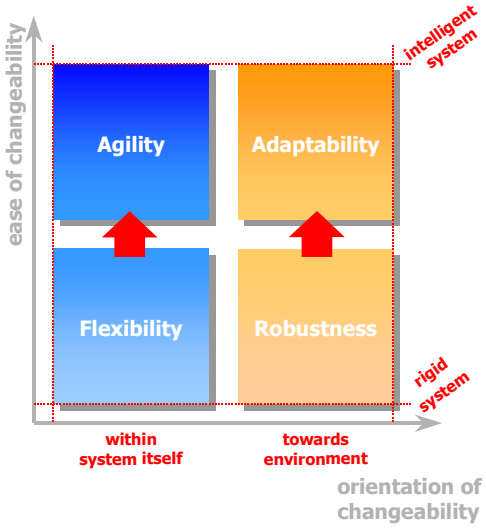


Figure 1 The Four Aspects of Changeability

The principles have been derived of various research projects and a number of sources in literature like Adaptive System Group (1999), Altshuller (1984), Dove (1999), Fey (1998), Fricke (1999), Maier (1998), Negele (1998), Open Systems Joint Task Force (1998), Rechtin (1991), Rechtin and Maier (1997), Suh (1990).

THE BASIC PRINCIPLES

Ideality/Simplicity. This principle is derived from the basic Pattern of Evolution in TRIZ[2], that all system evolve towards increasing ideality. Ideality in this context is defined as the ratio of a systems sum of useful functions against a systems sum of harmful or undesired effects. Based on that principle an ideal system consists of only useful functions, which may be interpreted as establishing small, simple units/elements with a minimized number of interfaces (loose coupling among and strong cohesion within modules) within an architecture. Moreover, an ideal system makes use of already existing resources and applies principles of

[2] comprehensive material on TRIZ and the patterns of evolution is contained in Altshuller (1984), Fey (1998), and Schulz and Clausing (1998)

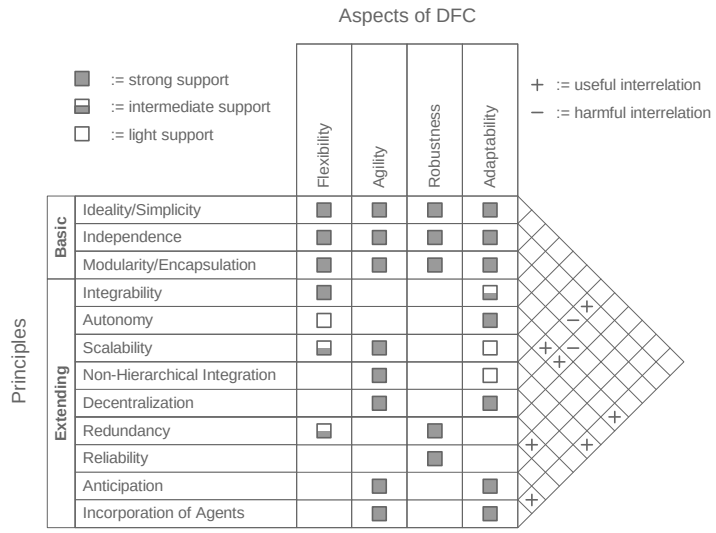


Figure 2. The Aspect-Principle-Correlation Matrix (Schulz & Fricke, 1999)

design streamlining. A similar approach can be found in Function Analysis (Akiyama, 1991), where the number of secondary functions, i.e., functions only supporting primary or main functions is to be minimized.

The principle of ideality/simplicity directly correlates to the information axiom (second axiom) in axiomatic design as introduced by Suh (1990). Suh defines an 'information content' incorporated within each system architecture, which is basically representing the degree of complexity needed to describe the architecture. A system architecture with a low information content is thus less complex, that is 'simpler' and therefore to prefer or to aim at. Suh also provides an approach to measure the information content and thus to evaluate alternative architectures.

Independence. This principle is derived from the axiomatic approach to design as introduced by Suh (1990). According to the first axiom in axiomatic design each system function or functional requirement has to be satisfied by an independent design parameter. A design parameter is representing the physical embodiment of a functions solution, that is i.e. a physical principle, a parameter, a component, etc. Independence of design parameters means, that changing a design parameter does not affect any related design parameters and thus not the proper operation of related functions. Suh (1990) distinguishes three degrees or levels of independence, which are defined as coupled, decoupled, and uncoupled. The relation between system functions and design parameter and their degree or level of coupling is displayed using design matrices. Capturing the properties of an architecture with respect to independence in that type of matrices is basis for evaluating alternative architectures applying metrics.

Modularity/Encapsulation. Building a system architecture that clusters the system's functions into various modules while minimizing the coupling among the modules (loose coupling) and maximizing the cohesion within the modules (strong cohesion) yields great benefits. In general two basic types of modularity may be distinguished, vertical and horizontal modularity. While horizontal modularity represents the clustering of a systems functions in different modules within a common layer, vertical modularity represents a layering of an architecture. According to Tate (1999) three types of modularity[3] may be distinguished, which basically correlate to the different type of architecture under consideration within the various phases of the design phase (e.g. problem architecture, operational architecture, functional architecture, physical architecture, etc.).

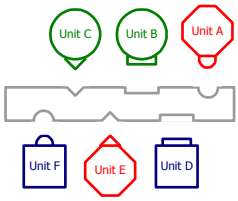
Modular architectures support reuse of elements, modules or even entire sections of an architecture with a certain scope of functionality and defined interfaces. Ease of exchanging and adapting modules or layers is facilitated incorporating self-sufficient, distinct, and not intimately integrated units. An implementation of platform concepts is possible as well as the use as a reference architecture for system evolution. Robustness is greatly enhanced since the impact of changes or noises is limited or isolated within the modules or layers.

SELECTED EXTENDING PRINCIPLES

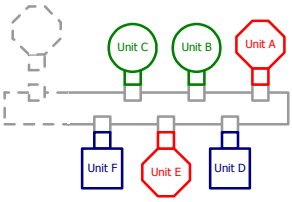
Integrability. This principle is key to achieving flexibility and adaptability. Integrability is characterized by compatibility and inter-operability applying generic, open, or common/consistent interfaces. Compatibility and inter-operability are necessary in a rapidly changing environment built of multiple interrelated systems. These abilities are even enhanced by implementing only mature and robust functions delivering a constant range/degree of functionality and thus ensuring stable interfaces independent of the environment. Concerning the principle of intgrability there is a strong correlation to the work being pursued in the area of open systems which is mainly driven by the DoD and related

[3] Tate (1999) distinguishes between resource (ease of manufacturing), interfacial (independence of system modules), and operational (range of operational variety) modularity

industries (Open Systems Joint Task Force, 1998). The principle of integrability is in particular critical for architectures in the context of system of systems, that is architectures having strong interrelations with systems sharing its operational context. This perspective is also supported by Maier (1998).



(a) proprietary interfaces
(all units have specific interfaces and are not interchangeable)



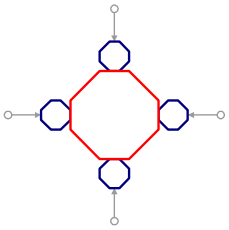
(b) generic and common interfaces
(all units share a common interface and are interchangeable)

Figure 2. Principle of Integrability

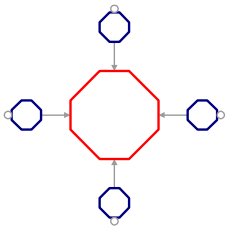
As an example for integrability web-clients may be mentioned. These clients serve as a generic interface for information retrieval independent from the hardware platform they are running on. To cope with the dramatically increasing number of Electronic Control Units (ECUs) automotive industry introduced bus communication among the ECUs to provide common interfaces for at least groups of ECUs. A new ECU could be added with regard to the already existing interface. This is also an example for scalability.

Decentralization. This principle is key to agility and adaptability. Based on loose coupling and strong cohesion a decentralized distribution of control (see autonomy), information, resources, attributes, and properties within the system architecture strengthens the capability of the system to rapidly adapt itself towards its environment and to respond autonomously to changing requirements. Thus necessary decisions are made at the point of 'best knowledge and information', while all knowledge and information has to remain accessible throughout the entire system for decisions on system level. This also enables the allocation of attributes or properties to the most appropriate location within the system.

A critical aspect within distributed control is to ensure consistency of objects/units throughout the entire system, that is this principle may have a harmful interaction with the principle of integrability.



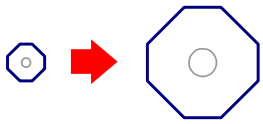
(a) centralized system
(all resources located at a single place)



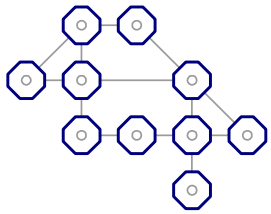
(b) decentralized system
(resources distributed to appropriate locations)

Figure 3. Principle of Decentralization

Within modern fly-by-wire flight control systems elevators, rudders, or spoilers are no longer steered by force of the pilot but by actuators located directly at the elevator, rudder, or spoiler and acting on stimulation by electric signals.



(a) single element scaled by up-/downsizing
its characteristic parameters



(b) increase/decrease number of multiple identical
elements linked to provide a scaled architecture

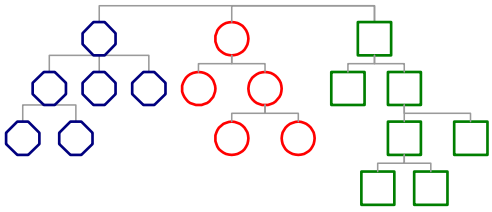
Figure 4. Principle of Scalability

Scalability/Self-Similarity. This principle is a key to flexibility, agility and adaptability. Based on elements independent from scale (fractals), architectures may be scaled upwards or downward. Basically there are two different ways of approaching scalability. First several identical elements of the architecture may be linked together to provide scaled performance or functionality. Second a single element of the

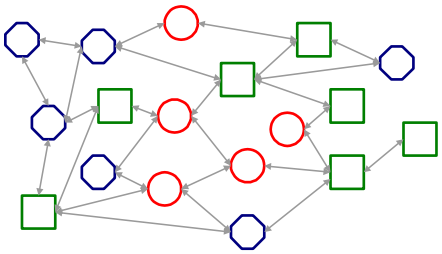
architecture may be scaled by up-/downsizing its characteristic parameters. As a basis the system architecture has to provide the necessary capability for an unrestricted increase or decrease of total unit population within the system.

Typical examples for scalability besides bus communication in today's cars' on-board networks (see integrability) are expandable launchers using additional boosters for higher payload into orbit (e.g. Ariane 4, Titan IV, etc.).

Non-Hierarchical Integration. This principle is key to agility and adaptability. Non-hierarchical integration is characterized by linking object/units across the total system, that is with no respect to any type of modularity or encapsulation. This ensures a direct, flexible, and fast communication, negotiation, or interaction among objects/units. It also covers the evolution of new and the destruction of obsolete links among objects/units, partially self-organized by objects/units based on autonomy and incorporation of agents.



(a) hierarchical integration
(inflexible and slow communication across system)



(b) non-hierarchical integration
(interaction interfaces on same level ensure fast and flexible communication across system)

Figure 5. Principle of Non-Hierarchical Integration

A critical aspect within non-hierarchical integration is establishing links among objects/units within different modules or layers of the system architecture, that is this principle might have a harmful interaction with the principle of modularity/ encapsulation.

Any type of cross-functional or integrated product team represents a typical example for non-hierarchical integration. Moreover the world wide web is another example for rapidly changing but direct communication based on standardized and common interfaces (i.e. TCP/IP Protocol).

PRACTICES AND METRICS

Practices. The enabling principles introduced in the paragraph above merely indicate which characteristics system architectures should incorporate in order to be changeable. But are these characteristics actually implemented within the design of a certain architecture? The need to answer this question lead to the definition of a variety of practices supporting the implementation of single principles. Due to the scope of this paper only few simple examples shall outline the general idea.

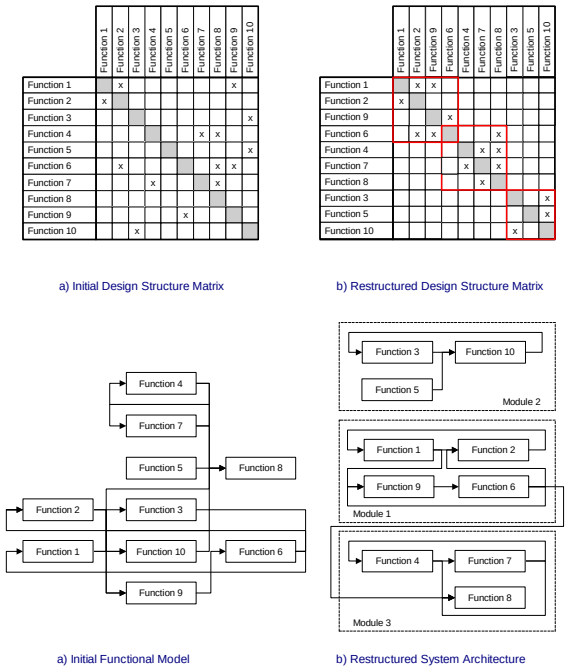


Figure 6 Defining Modularity using the Design Structure Matrix

Within the principle of modularity a well-known practice can be introduced. A clustering of system objects/units into modules is supported by the design structure matrix, which has been first introduced by Steward (1981) and is illustrated in Figure 6. Design structure matrices capture the existence and type of interrelations among a system's elements. Various algorithms depending on the objectives are applicable to restructure these matrices in order to achieve a modular/encapsulated architecture (Browning, 1998). Based on the resulting clusters within the architecture platform concepts are applicable to incorporate rather static functions, that is functions not imposed to rapid technological evolution and highly dependent on specific market segments, into a common platform,

serving as a basis for a flexible product family. Rather dynamic functions, that is functions imposed to rapid technological evolution and highly dependent on specific market segments, will be incorporated into derivatives of the platform.

Measurement. It is not necessarily useful to implement changeability into a system architecture to its full extent, that is implement all four aspects of changeability throughout the entire system architecture. Basically the architecting process should be guided by the question: **Where** in the system architecture do I need **what type of** and **how much** changeability?

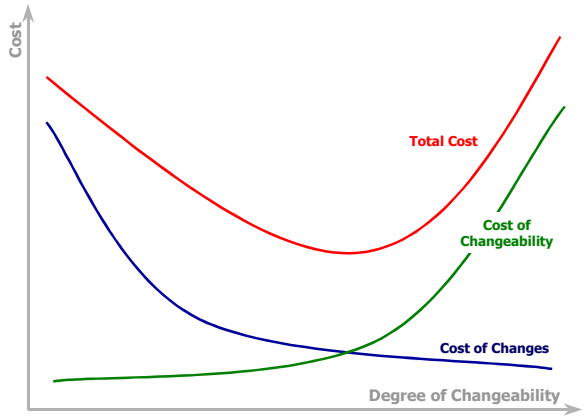


Figure 7 Degree of Changeability vs. Sources of Cost

Further, implementation of changeability might be accompanied by a certain effort in terms of money or time. The trade-off between the right price to pay or the right amount of time to spend to the expected benefit is a critical consideration. Therefore our current research focuses on advancing our knowledge in two critical areas:

First, the measurement of the degree to which certain principles enabling changeability have to be implemented within a system competing in a specific business environment. This area is mainly guided by the following questions:

- How much X (i.e. modularity) is incorporated?
- How much X (i.e. integrability) is required?

Second, the measurement of the impact an implementation of certain principles within a system architecture has. This area is guided by the following questions:

- What is the cost for implementing it ?
- What is the benefit for implementing it ?

A typical relationship between the cost imposed on a system architecture due to incorporating changeability and due to changes expected within its lifecycle is illustrated in Figure 7.

Cost of changeability within an architecture is typically imposed by higher design or manufacturing effort due to **incorporating** changeability, while cost of changes are typically imposed by a higher effort to change a system architecture at any time within its lifecycle due to **not incorporating** changeability. Both curves result in a distribution of the total cost of a system architecture depending on its degree of changeability. A certain range of changeability within the architecture results in minimized total cost. This window of opportunity is what should be aimed at during the design of a specific system architecture.

CONCLUSIONS

Benefits. To stay ahead of competition in dynamic environments it is inevitable to ensure sustaining superior system capabilities, i.e., offering state of the art systems throughout their entire lifecycle. Therefore, systems and their architectures have to offer changeability throughout their lifecycle not only within themselves but also towards their environments. To cope with these challenges a strategy ‘Design for Changeability’ is proposed, incorporating the following four attributes.

- Flexibility
- Agility
- Robustness
- Adaptability

System architectures characterized by these attributes will yield great enhancements. Technology insertion throughout the entire system life-cycle to ensure superior system capabilities and customized functionality is possible. Upgrade opportunities and the ease of customization leads to high attractiveness to customers or stakeholders. Rapid responsiveness to emerging and changing markets is facilitated by adapting the architectures accordingly based on modular and platform concepts. Reduced life cycle cost result from cross-platform integrability, reuse of units, modules, or architectures, while the impact of changes necessary to adapt the architecture throughout the entire lifecycle is minimized.

Discussion and Outlook. Although a wide range of benefits already has been achieved there are open issues which have not been addressed yet. A consistent framework building from strategies incorporating the four attributes of ‘Design for Changeability’, principles characterizing the properties of architectures meeting the four attributes, practices supporting the principle’s implementation, and metrics measuring and controlling the maturity of implementation has been proposed. The principles introduced are not considered to be comprehensive yet. Moreover case studies have to be performed ensuring applicability of the framework and its elements. The question what principles and practices

are to applied in which environment and context is still to be answered. Interrelations, both useful and harmful, among the strategies, principles, practices, and metrics so far have only been identified on a very high level and are a main subject to further research.

REFERENCES

Adaptive Systems Group, The Harlequin Group, http://www.harlequin.com/products/asg, 1998

Akiyama, K.V., Function Analysis – Systematic Improvement of Quality and Performance. Productivity Press, Cambridge, 1991

Altshuller, G., Creativity as an Exact Science. Gordon & Breach Science Publishers, New York, 1984

Boehm, B.W., Software Engineering Economics. Prentice Hall, Englewood Cliffs, 1981

Browning, T.R., Modeling and Analyzing Cost, Schedule and Performance in Complex System Product Development, Ph.D Thesis, Massachusetts Institute of Technology, 1998

Clark, K. and Fujimoto, T., Product Development Performance. Strategy, Organization and Management in the World Auto Industries, Harvard Business School Press, Boston, 1991

Clausing, D., Total Quality Development, ASME Press, New York, 1994

Dove, R.K., Design Principles for Highly Adaptable Business Systems, Paradigm Shift International, Taos County, NM, 1999

Fey, V., Theory of Inventive Problem Solving (TRIZ), The TRIZ Group, 1998

Fricke, E., Der Änderungsprozeß als Grundlage einer nutzerzentrierten Systementwicklung, Ph.D. Thesis. Technical University of Munich, 1999

Fricke, E., Gebhard B., Negele H., Igenbergs E., No Innovation Process without Changes, but... Proceedings of the 7th International Symposium of INCOSE, Los Angeles, 1997

Iansiti, M., Technology Integration, Harvard Business School Press, Boston, 1998.

Kelly, K., New Rules for the New Economy, Viking Penguin, New York, 1998.

Maier, M., Architecting Principles for System of Systems, Systems Engineering, Vol. I, No. 4, 1998, pp. 267-284

Open Systems Joint Task Force, (OSJTF) http://www.acd.osd.mil/osjtf, 1998

Rechtin, E., Systems Architecting - Creating and Building Complex Systems, Prentice Hall, 1991

Rechtin, E. and Maier, M.W., The Art of Systems Architecting. CRC Press, Boca Raton, FL, 1997

Ring, J. and Fricke, E., Rapid Evolution of All Your Systems – Problem or Opportunity?, Proceedings of IEEE 17th DASC, Seattle, October 1998

Schulz, A., Fricke, E., Incorporating Flexibility, Agility, Robustness, and Adaptability within the Design of Integrated Systems – Key to Success?, Proceedings of the IEEE/AIAA 18th Digital Avionics Systems Conference, St. Louis, 1999

Steiner, Rick, Systems Architecture and Evolvability - Definitions and Perspective. Proceedings of 8th Annual Symposium of INCOSE, Vancouver, 1998

Steward, D.V., The Design Structure System – A Method for Managing the Design of Complex Systems, IEEE Transactions on Engineering Management, Vol. 28/3, 1981, p. 71-74

Suh, N.P., The Principles of Design, New York, Oxford University Press, 1990

Taguchi, G., Taguchi on Robust Technology. ASME Press, New York, 1993

Tate, D., A Roadmap for Decomposition: Activities, Theories, and Tools for System Design, Ph.D. Thesis, Massachusetts Institute of Technology, 1999

Ward, Allen, Jeffrey K. Liker, John J. Cristiano and Durward K. Sobek II, The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster, Sloan Management Review 36(3): 43-61, 1995

Wenzel, S., Bauch, T., Fricke, E., Negele, H., Concurrent Engineering and More – A Systematic Approach to Successful Product Development, Proceedings of the 7th International Symposium of INCOSE, Los Angeles, 1997

Wheelwright, S., Clark, K., Revolutionizing Product Development, The Free Press, New York, 1992

AUTHOR'S BIOGRAPHIES

Armin P. Schulz is Research Assistant and PhD candidate at the Division of Astronautics at the Technical University of Munich. He received his master's degree in aerospace engineering from the Technical University of Munich in 1998, after completing his master's thesis at the Center for Innovation in Product Development at MIT. His research focuses on technology development processes, systems design and architecting. He is a co-founder and currently secretary of the German Chapter of INCOSE.

Ernst Fricke is Assistant Professor at the Division of Astronautics at the Technical University of Munich. He received his master's degree in aerospace engineering from the Technical University of Munich in 1994 and his Ph.D. in Systems Engineering from the Technical University of Munich in January 1999. His research focuses on managing changes and evolution in product development. He is a co-founder and active member of the German Chapter of INCOSE.

Eduard Igenbergs is Professor of Astronautics

and Director of the Division of Astronautics at the Technical University of Munich. He has a Ph.D. and a masters degree in mechanical engineering. He had a dust counter experiment on the japanese HITEN spacecraft to the moon and a space debris experiment on board the German BREMSAT. Another dust counter experiment will fly 1998 to mars with the japanese Planet-B spacecraft. He developed the Munich Space Chair (MSC), a body restraint system on board MIR. He has been teaching and researching in the field of Systems Engineering for more than 15 years. He is an active member of the German Chapter of INCOSE.