AN INTELLIGENT DESIGN ENVIRONMENT FOR CHANGEABILITY MANAGEMENT - APPLICATION TO MANUFACTURING SYSTEMS

Benkamoun, Nadège; Kouiss, Khalid; Huyet, Anne-Lise
Université Blaise Pascal, France

Abstract

Design for system changeability and reusability has been sought by engineers from several disciplines. It has lead to the emergence of numerous strategies and paradigms. Especially in conceptual design phase, when knowledge about requirements, design problem and system specifications is incomplete, the future for effective changeability is already at stake.
This work presents knowledge related to changeability strategies as well as enablers, namely modularity, interfacability, changeability and reusability ontologies. It is illustrated by examples of manufacturing system design. The established formalism leads to a formal organization of the required functionalities for changeability management; the paper presents an intelligent design environment for changeability management. Its collaborative architecture is based on two concurrent and continuous processes: designing changeability and leveraging on it during the whole system (re)design lifecycle. Dedicated agents cooperate together, so they offer an intelligent and distributed design environment. As a result, designers are assisted to adopt a systemic design approach to analyse, design, evaluate and maintain changeability.

Keywords: Systems engineering (SE), Design management, Design engineering, Collaborative design

Contact:
Nadège Benkamoun
Université Blaise Pascal, Clermont-Ferrand
Institut Pascal - MMS
France
nbenkamoun@ifma.fr

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

The evolution of large-scale complex systems over long lifetime has pushed designers to study changeability as a life-cycle system property (De Weck et al., 2011). Changeability can be defined as the degree to which a system is able to adapt to changing circumstances. As far as industrial systems are concerned, highly changing and fluctuating production contexts have lead manufacturing systems communities to follow the same path. The paradigm of Reconfigurable Manufacturing System (RMS) was defined in the 90's (Koren et al., 1999) as a manufacturing system whose structure is able to react and adapt to change by physically changing its components through adding, removing or modifying machine modules. Its focused flexibility on demand through re-configuration phases was distinguished from the Flexible Manufacturing System (FMS) paradigm, where generalized flexibility is built-in a priori (ElMaraghy, 2009).
Beyond all the discussions about the different changeability types and enablers, changeability in design has rarely been treated in a systemic way for manufacturing systems. Indeed, manufacturing system design is often addressed under specific design sub-problems - resource requirements, resource layout, material flow, buffer capacity etc. (Chryssolouris, 2005) - themselves often surrounded by specific assumptions or restrictions. But before design sub-problem formulations, design projects have first to deal with high-level description requirements leading to solution principles for the whole system. This preliminary phase - called conceptual design - is characterized by incomplete and imprecise knowledge about system environment and specifications. However this phase is very challenging as it is during conceptual design that the impact of decisions is the highest (Wang et al., 2002). Therefore changeability management should be studied during the preliminary design phase in order to take full advantage of it.
This articles falls into systemic engineering design approach category in adopting an integrative holistic view of large-scale and complex systems. Our work aims at 1- formalizing and organizing knowledge for changeability management during system (re)design activities and 2- proposing an intelligent design environment to assist designers with changeability management - namely the architecture of a knowledge-based system - according to its defined functionalities.
Section 2 introduces principles and knowledge for designing system changeability. Then, Section 3 presents the architecture of an intelligent design environment for changeability management. Its collaborative organization is justified, the interacting agents are described, and their functionalities and supporting knowledge are detailed.

2 SUPPORTING KNOWLEDGE FOR SYSTEM CHANGEABILITY

Designing a system which is able to cope with changing contexts and requirements, pre-requires formalizing and structuring knowledge about changeability strategies. Architectural characteristics and principles to make the system adaptable and changeable through new system configurations, variants or instances are reviewed. For the rest of the paper, we define domains of definition for studying system's elements changeability.

2.1 Architectural strategies for changeable systems

2.1.1 Product variety strategies

Variation and diversity of customer needs in product design has caused the emergence of variety management strategies (ElMaraghy et al., 2013). For product or system design, leveraging on commonalities has been crucial to survive economically in a context marked with change and variety. Architecture design strategies for reuse have been developed (e.g. product family, product platform, modularity and commonality in product architecture) (Jiao et al., 2007) but also strategic processes for reuse as product line engineering or product family engineering. Considering a manufacturing system as a product with its varying production context requirements, product variety management principles are directly applicable to manufacturing systems.

2.1.2 Modularity as a change enabler

Modularity is considered as a great enabler for reuse strategy among different disciplines. A modular architecture enables changeability by easily adding, substituting, and removing predefined modules. In system engineering, it is often considered as a major system property and even a prerequisite for changeability; in software engineering reusable and evolvable solutions are made possible with segmenting the code into objects and services. In product design engineering, modular and flexible product platform are very trendy. And in manufacturing engineering, modularity and interfaceability consist of the two main characteristics of a RMS.

2.1.3 Modularity design principles

In all of these disciplines, module design follows two main principles. First, in order to be inter-changeable, intra-module cohesion should be high and inter-modules coupling should be low. Second, as modularity induces increasing development cost for standardizing interfaces, the degree of granularity has to be balanced between required system variants distinctiveness (i.e. modularity) and planned commonality (i.e integrability). In other words, designers have to differentiate between necessary flexible assets and integrated system structure. For that, levels of reusability and adaptability have to be defined. A formalization of these two principles will be presented as some required functionalities for a design environment presented in Section 3.

2.2 System modules representation

2.2.1 Physical and rationale domains

System modules can cover multiple viewpoints. For instance, modules can be defined in the physical domain, from high level description (e.g. robot, transportation system, Automated Guided Vehicles-AGV, jigs, machine profile) to detailed level (path trajectory, trolley dimensions, handling robot capabilities, conveyor speed, PLC program). But in reality, level distinctions are application-dependent or domain-dependent. In any case, a physical module is always first defined according to the requirements it is related to. Describing a module according to the functionalities, the requirements or largely, the intention it answers is more meaningful than an operational or physical description. Design rationale approaches share the same motivation. Instead of only specifying assets by their operational specifications, design rationale intends to describe the intention behind the choices all along the design process. Especially for a changeability perspective, reusability is not only about physical modules capabilities (e.g. physical interchangeability) but also the range of requirements and functionalities the system can cope with. Therefore, we defined system modules in two domains: the physical domain and the rationale domain.

2.2.2 Modules types: blocks and requirements

Consistently with the systemic viewpoint of this paper, a representation formalism of conceptual designed elements needs to be understood and made available by multi-disciplinary designers. This representation formalism for design rationale has to be domain-independent. Domain-independent representation approaches to record and reuse design rationale have been reviewed by (Regli et al., 2000). As an example of systemic design rationale for manufacturing system engineering, Cochran et al. (2001) have developed the Manufacturing System Design Decomposition (MSDD) based on the axiomatic design approach (Suh, 1998). It integrates relative design disciplines (i.e. information systems, manufacturing strategy, supply chain, human work system design, facility design process, equipment design and product design) and trace the manufacturing system design process in terms of relations between design objectives (i.e. Functional Requirements) and solutions (i.e. Design Parameters). In systems engineering approaches, requirement engineering and requirement management tools also aim to trace requirements as implicit design decisions. The modelling language SysML includes a requirement diagram and relationship formalisms between requirements and allocated blocks (Friendenthal et al., 2011).
Falling into these approaches, we differentiate two module types: requirements in the rationale problem domain and structural blocks in the physical solution domain. A requirement is defined as a statement that specifies a need, a condition, capability that should be achieved in the system. A structural block is a modular unit of structure that physically defines the system.

2.2.3 Relationship formalism

Structural modules and requirement modules are highly-related to each other. Analysing coupling in physical domain, in rationale domain but also between both domains is necessary. A new formalism for typing the different kind of relationship is presented in Figure 1. Requirements are represented in square and structural blocks in circles. It establishes relationship types between requirements (i.e. business, functional, non-functional, behavioural, interface or constraint type requirements) and realizing blocks (i.e. physical, logical, or hybrid type). Some are inspired from the SysML relationships formalism, but this one brings much more details about their meanings. First, a requirement can arise according two relationship scenarios: 1- it <refines> another requirement, it brings more details and updates the old requirement version; 2- it <derives> from a requirement (e.g. functional decomposition) or from a structural block, meaning that a technical choice has influenced its formulation. A structural block can relate to a requirement in three ways: 1- it <satisfies> a requirement, meaning that the requirement source has directly caused the existence of the block in the system; 2- it is <specified by> a requirement that acts as a specification, it constitutes instructions for the later development phase; 3- it <allocates> to a requirement, which means that the requirement has a dependency link as it influences, or concerns an existing block. Last, structural blocks can also have physical links as inheritance, composition or association. It can be used in rationally deriving associated requirement to the related blocks. An illustration of rationale and physical relationships between requirement modules and structural modules is given in Figure 2.

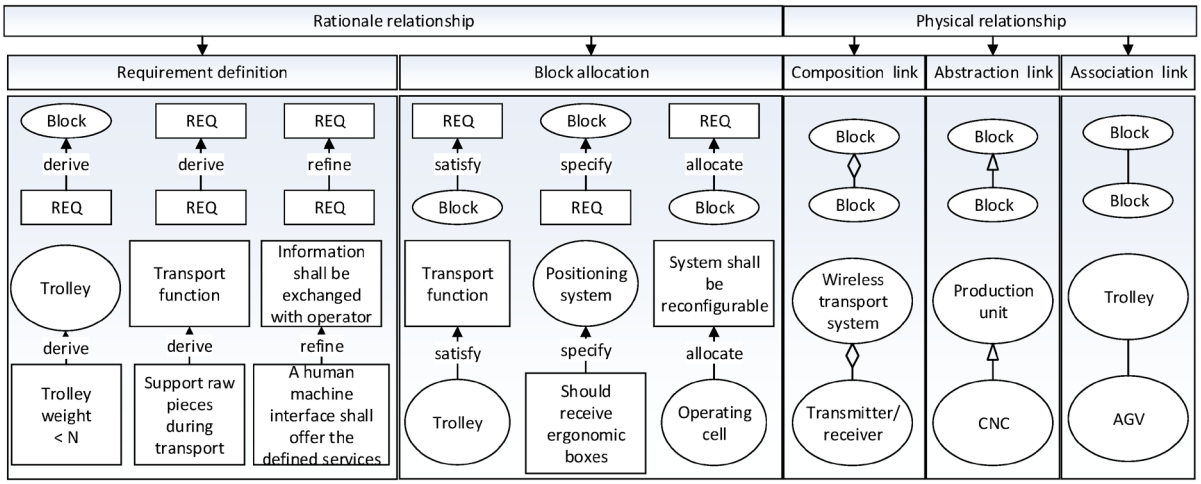

Figure 1. Project elements relationships in rationale and physical domains

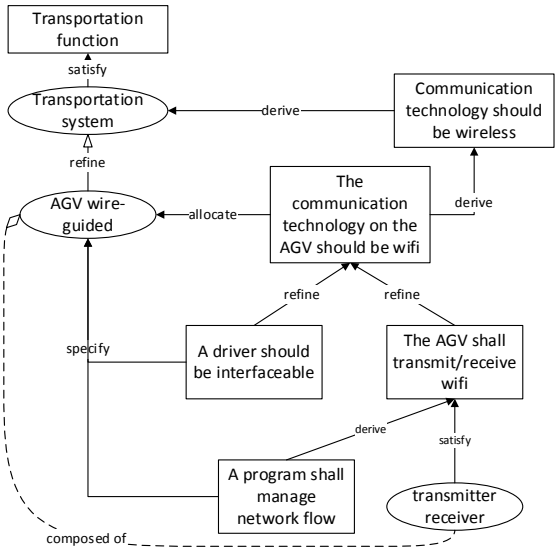

Figure 2. Illustration of relationships between requirements and structural blocks

This formalism is a preliminary work to enable future analysis of project consistence, dependencies, and thus later analysis of modularity in rationale domain. Design projects are often too complex for system engineers to even think about module possibilities for reuse, especially when knowledge has not been formalized or organized. Changeability and reuse in system architecture would be possible only if the designer is assisted with a supporting tool for changeability design and management. Therefore, the proposition extends the research area of Design Theory and Methodology (DMT) initiated by (ElMaraghy et al., 1989) to support intelligent systems; an intelligent design environment contribution with methodological principles kernel is presented to overcome the apparent complexity and guide designers through changeability design and management projects.

3 ARCHITECTURE OF AN INTELLIGENT DESIGN ENVIRONMENT FOR CHANGEABILITY MANAGEMENT

Organizing and formalizing expert knowledge into intelligent systems is essential for assisting designers to cope with complex and changeable problems. The intelligent design environment proposed in Figure 3 was meant to act like a design assistant; thus, designers keep a central role by interacting and initiating the different system functionalities. The architecture of the intelligent design environment is presented under a collaborative architecture viewpoint. It enables distributed design environments for the various actors. Each agent is represented with its main functionality and its supporting knowledge.

3.1 A collaborative design environment

Changeability management has to be viewed under two complementary viewpoints: design for changeability (i.e. creation of modules and corresponding interfaces) and leveraging on changeability capabilities (i.e. taking benefits of the developed and invested modules and interfaces). To establish changeability as a lifecycle system property, changeability management has to be concurrent and even collaborative with the design project of the system. As design is a distributed and asynchronous problem, the system is based on a collaborative architecture. The design environment in Figure 3 entails these three concurrent processes, represented as three macro-agents (MA1, MA2, MA3) collaborating together. A second level of collaboration takes place between the agents within MA1 and MA2. Collaborative modules –or agents – work together to solve problems thanks to their communication skills and their own capabilities for solving problems (Shen et al., 2003).

3.1.1 Blackboards architecture

The collaborative design environment is organized as a blackboard architecture (Shen et al., 2003). The main blackboard is a data repository for information about system modularity and interfaceability capabilities. It is related to a second blackboard that includes knowledge about system architecture in rationale and physical domains. Surrounding them, knowledge sources (KSs), namely the three macro-agents, are permitted to communicate and interact with them while they operate. This architecture is relevant for continuous design process and continuous analysis of system elements modularity and interfaceability.

3.1.2 Artificial intelligence potential support

Intensive research has been undertaken to apply artificial intelligence (AI) to design. Without focusing too much on this topic, we can list some supporting AI techniques that would fit into the proposed architecture in Figure 3. Communication protocols (e.g. the Knowledge Query and Management Language – KQML), negotiation protocols (e.g. the contract-net protocol), retrieval knowledge mechanisms for efficient blackboard storage management, semi-automation of problem-solving methods and inference mechanisms for knowledge-based systems (Studer et al., 1998) are some examples. In industrial domain, the increasing complexity of products to manufacture, and thus of manufacturing systems, has led to more and more applications of distributed intelligence artificial to decisions problems (Kouiss et al., 2002). Bakhtari and Bartsch-Spörl (1994) have identified AI technology potential functionalities for design requirements as negotiation assisting in conflict and version management, assessing design quality, support with relevant information, or help for innovation.

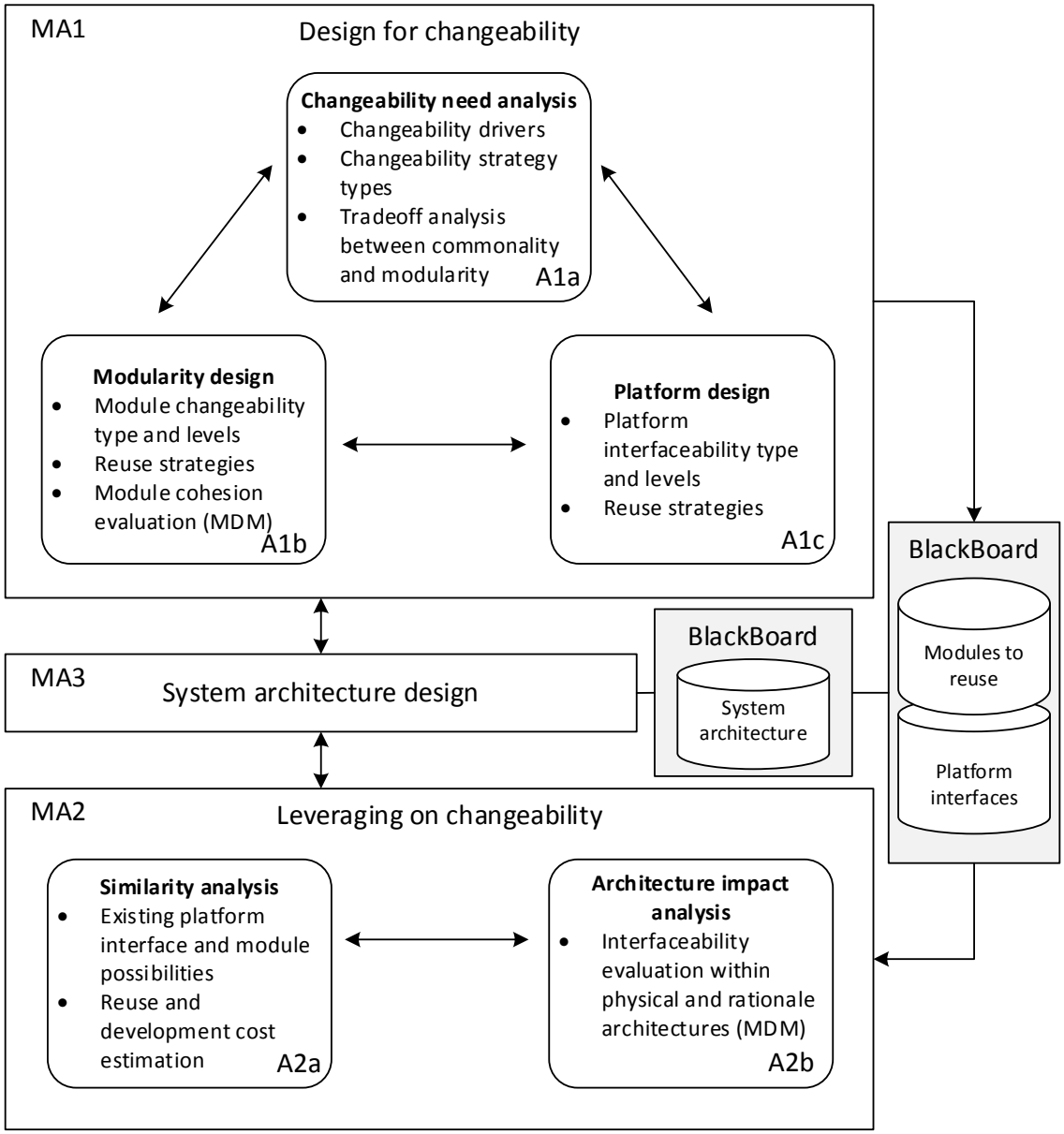

Figure 3. Architecture of the intelligent design environment for changeability management

3.2 Design for changeability – MA1

The first macro-agent (MA1) aims to support design for changeability. Changeability need analysis (A1a) first initiates the design process. As modularity interferes with platform interfaceability, module (A1b) and platform interface (A1c) design processes are tackled concurrently.

3.2.1 Changeability need analysis (A1a)

Before any changeability management process, changeability must be designed at the outset. A mindset that predicts design scenarios about plausible unfolding futures (Rhodes and Ross, 2009) is required to evaluate the range and type for the required changeability. To support forethought about changeability, a design environment has to support the specification of changeability requirements by associated ontologies. Classifying changeability requirements according to identification of change initiator and specification of the related changeability strategy would encourage designers to think in an anticipatory way. For that, two requirement stereotypes have been formalized:

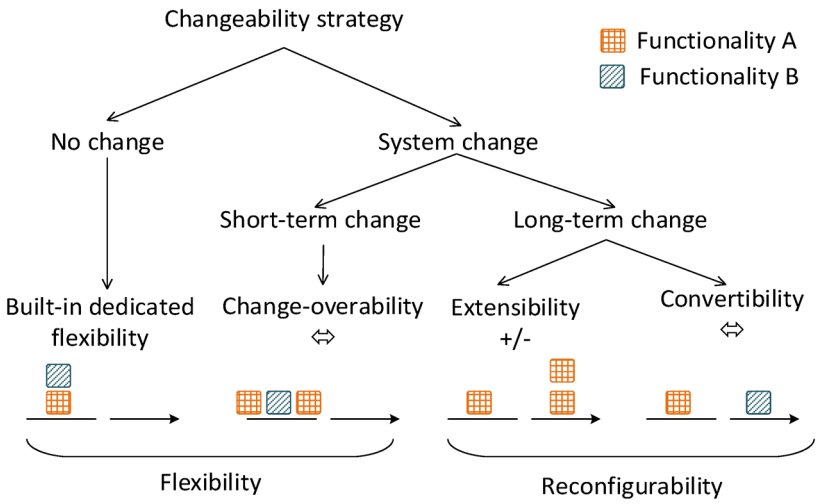

Figure 4. Changeability strategies

- Stereotype for changeability drivers. With this stereotype, the designer is encouraged to relate requirements to strategic change drivers. Changeability requirements can be anticipated by analysing the plausible change drivers. In manufacturing system design, change drivers can refer to product change (new product variant, new product family, new volumes), process change (manufacturing or logistic process), technology or standard change, environment change (new layout, new plant) or any strategic motivation for change.
- Stereotype for changeability strategy. In order to deal with new functionality or new volume, Figure 4 illustrates different changeability strategies that can be implemented. We first differentiate system changeability strategies that absorb a new change without requiring a new implementation (i.e. built-in modules), from the ones that require to implement physical change (i.e. module changeability). Within built-in module strategies, we differentiate modules dedicated to specific functionalities, from the ones that entail several functionalities - type swiss army knife. Module changeability strategies are differentiated according to their time scale for change: flexibility as planned change type "plug and produce" (i.e. short-term convertibility) or reconfigurability as hypothetical future change for system convertibility or extensibility.

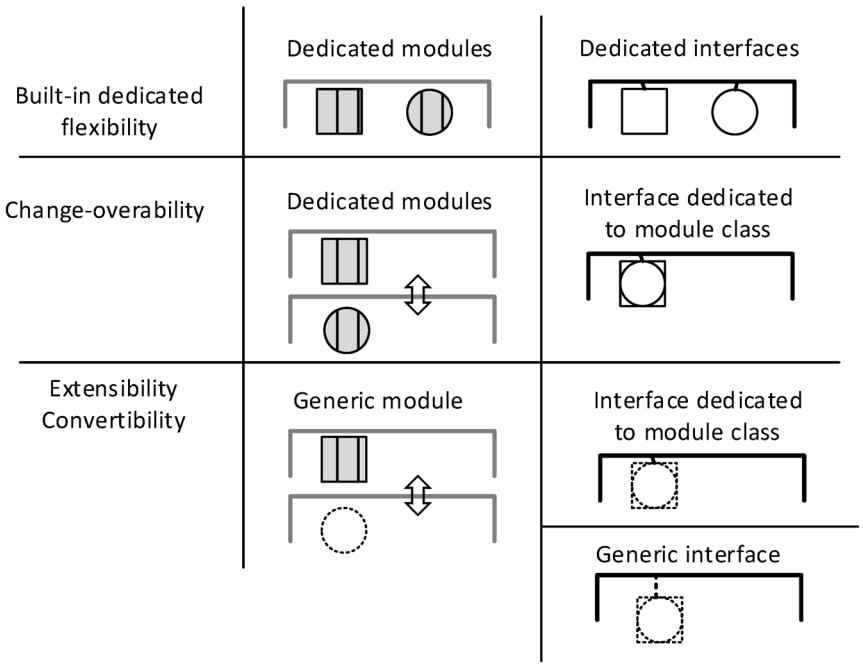

Figure 5. Module and interface types according to changeability strategies

3.2.2 Modularity and platform design (A1b and A1c)

Module changeability and platform interfaceability. The identified changeability strategies result in interface and module requirements for the system platform. Deciding between implementing dedicated modules (type "plug and play") and abstracting module types depends on the considered time scale, the completeness of future knowledge, the changeability strategy, or investment strategies. The different identified scenarios for changeability strategies (Figure 4) are now considered under module and interface needs (Figure 5).

- In case of no system change, modules dedicated to two functionalities will also require dedicated interfaces (e.g. a machine for drilling and a machine for turning).
- In the short-term change-overability scenario, the dedicated modules will require an interface dedicated to the range of modules it can be plugged to (e.g. a machine tool with a tool holder for different size of tools).
- Long-term changes call for abstraction of modules, as knowledge concerning the "shape" of the future module is incomplete and fuzzy. Therefore, interfaces are either dedicated to a module class – the interface being a constraint for the future module (e.g. machines shall communicate with Profibus standard) – or generic to a class of interfaces (e.g. platform requires a generic transportation functionality as input).

Reuse Strategy. Investing in module or interface development is justified by their required level of changeability, but also by their level of reusability. In product design, a product family (or product line) is defined as a group of products that share the same platform, namely the same group of assets. The main interest of product platform for design is the reuse of common assets to new product variants of the same family. However, other works support reuse strategy for any expert knowledge with a mere modules repository or a Knowledge Based System (KBS). For instance, Chalé Gongora et al. (2015) have proposed a new reuse strategy called "system and subsystems catalogue of building blocks" in which reuse does not only take place with product family assets, but also with project assets. Figure 6 offers a classification of system elements according to their level of reusability; it roughly distinguishes the fixed system core platform from potentially changeable modules.

- Fixed core platform: common core elements within any system variant of a system family.
- Platform modules: modules that have to be instantiated (or parameterized) in any system variant.
- Platform optional modules: available modules that can be used in any system variant without being common to all variants.
- Project specific modules: specific to a custom solution, but pointed out as being reusable in any project.

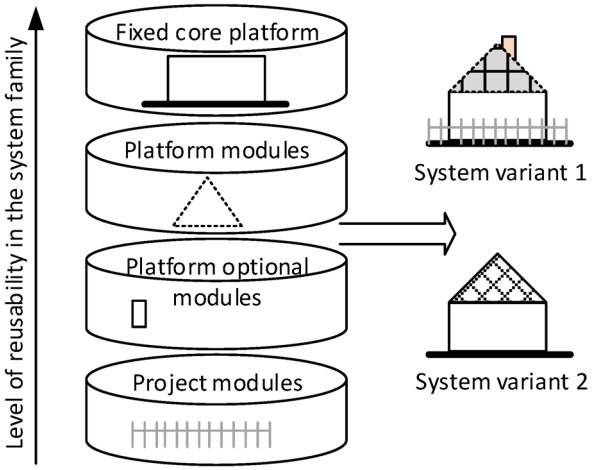

Figure 6. Classification of system elements according to their level of reusability

Module cohesion evaluation. To decide on an appropriate granularity level for modularity, intra-module cohesion should be maximal and inter-modules coupling should be minimal. For this purpose, tools to represent architectures and elements dependencies are necessary. Design structure matrix (DSM) is a tool that offers a representation of system dependencies between entities. Weights for different levels of dependencies can also be attributed. Multidomain matrix (MDM) is an extension of DSM modelling where elements dependencies are visualized simultaneously in different domain. Eppinger and Browning (2012) present various applications of DSM and MDM in three big domains: product, process and organization domains. We recall from Section 2 that modules can either be rationale requirements, or physical structural elements. Therefore modularity and interfaceability analysis should be carried out in the rationale and the physical domains. For this purpose, an MDM (Figure 7) is used to analyse system element interactions in rationale and physical domains.
Knowledge to represent these interactions can come from two sources: from design project relationships as it has been presented in Figure 2, but also from expert knowledge about interaction likelihoods.

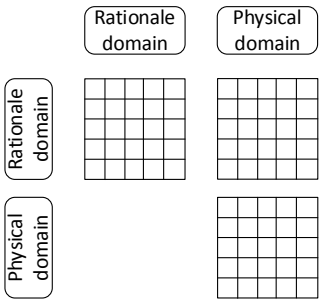

Figure 7. MDM for rationale and physical domains

3.3 Leveraging on changeability – MA2

The role of the second macro-agent (MA2) is to leverage on the existing system changeability capabilities. Knowledge bases with available interfaces and modules come from the central blackboard. They result of the changeability design process (MA1). Similarity between system requirements and potential reuse possibilities should be analysed and evaluated (A2a). Concurrently, reuse of existing modules and interfaces should also be studied under the change propagation consequences within the whole system architecture (A2b). MDM representation would also support this change propagation analysis.

Similarity analysis. Based on the classification of system modules in Figure 6, analysis of similarity shall follow the algorithm in Table1. Four types of actions (in bold) depend on the similarity of system requirements with existing system elements. From the less costly to the most costly scenario: 1-implement a platform module, 2-instantiate a module from the platform module class, 3-design a module class according fixed core platform constraints, 4-instantiate a module from the project module class. We focused on module reusability, but it should also induce decisions on interfaces reusability (c.f. Figure 5). Let us note that the similarity metric should be specific to the application.

Table 1. Similarity analysis algorithm

Input 1: *system requirement* Input 2: *fixed core platform*; *platform modules*; *platform optional modules*; *project modules* Output: *design decision for maximizing reusability*

If *system requirement* and *fixed core platform* are consistent
  If *system requirement* is enough similar to at least one *platform module*
  If one of the *platform module* is already instantiated Then **implement** *platform module*
  Else **instantiate a module** from the *platform module class*
  Else **design a module class** according *fixed core platform* constraints
Else If *system requirement* is enough similar to at least one *project module*
  Then **instantiate a module** from the *project module class*

4 CONCLUSION

The presented intelligent design environment enables changeability design from the outset. Designers are assisted by a collaborative design environment that integrates the different design viewpoints they have to go through. Its collaborative architecture highlights three concurrent processes; the underlying belief is that system architecture design process has to continuously collaborate with design for changeability (i.e. modularity and platform design) and with leveraging on changeability processes. Required knowledge for changeability (e.g. elements reusability levels, modularity and interfaceability strategies, changeability strategies and similarity analysis for leveraging on changeability) has been formalized and integrated to support the different system functionalities.
However, besides a contribution on integrating changeability principles within a unified framework, system changeability is for the first time not only seen under a physical viewpoint. System modules are defined in the rationale domain as requirements and in the physical domain as structural blocks.
Moreover, high potential for integrating AI approaches is offered by this paper. Semi-automation of knowledge retrieval for cohesion analysis, modularity or interfaceability design becomes possible. Application of negotiation mechanisms would also enhance collaboration support between the different stakeholders and designers. Finally, expanding knowledge bases with expert ontology from specific domain needs would greatly increase the applicability potential of this design environment.

REFERENCES

Bakhtari, S. and Bartsch-Spörl, B. (1994), "Bridging the gap between AI technology and design requirements", Artificial Intelligence in Design'94, Vol. 38 No. 1993, pp. 753–768.

Chalé Gongora, H.G., Ferrogalini, M. and Moreau, C. (2015), "How to Boost Product Line Engineering with MBSE", Complex Systems Design & Management, Springer International Publishing Switzerland 2015, pp. 239–268.

Chryssolouris, G. (2005), "The design of Manufacturing Systems", Manufacturing systems: theory and practice, Springer, pp. 329 – 461.

Cochran, D.S., Arinez, J.F., Duda, J.W. and Linck, J. (2001), "A decomposition approach for manufacturing system design", Journal of Manufacturing Systems, Vol. 20 No. 6, pp. 371–389.

ElMaraghy, H.A., Schuh, G., ElMaraghy, W., Piller, F., Schönsleben, P., Tseng, M. and Bernard, A. (2013), "Product variety management", CIRP Annals - Manufacturing Technology, Vol. 62 No. 2, pp. 629–652.

ElMaraghy, H.A. (2009), Changeable and reconfigurable manufacturing systems, London : Springer.

ElMaraghy, W., Seering, W. and Ullman, D. (1989), "Design Theory and Methodology", ASME First International Design Theory and Methodology Conference.

Eppinger, S.D. and Browning, T.R. (2012), Design structure matrix methods and applications, (Press, M.,Ed.).

Friendenthal, S., Steiner, R. and Moore, A. (2011), A practical guide to SysML: the systems modeling language, Elsevier.

Jiao, J., W. Simpson, T. and Siddique, Z. (2007), "Product family design and platform-based product development: a state-of-the-art review", Journal of Intelligent Manufacturing, Vol. 18, pp. 5–29.

Koren, Y., Heisel, U. and Jovane, F. (1999), "Reconfigurable manufacturing systems", CIRP Annals-Manufacturing Technology, Vol. 2 No. 48, pp. 527 – 540.

Kouiss, K., Gouarderes, E. and Massote, P. (2002), "Organisations distribuées des systèmes de pilotage", Fondements du pilotage des systèmes de production, Hermès Science Publications, pp. 80–117.

Regli, W., Hu, X., Atwood, M. and Sun, W. (2000), "A survey of design rationale systems: approaches, representation, capture and retrieval", Engineering with computers, Vol. 16 No. 3-4, pp. 209–235.

Rhodes, D. and Ross, A. (2009), "Anticipatory capacity: Leveraging model-based approaches to design systems for dynamic futures", Model-Based Systems Engineering, 2009. MBSE'09, pp. 46–51.

Shen, W., Norrie, D. and Barthès, J. (2003), Multi-agent systems for concurrent intelligent design and manufacturing, CRC press.

Studer, R., Benjamins, V. and Fensel, D. (1998), "Knowledge engineering: principles and methods", Data & knowledge engineering, Vol. 25, pp. 161–197.

Suh, N. (1998), "Axiomatic design theory for systems", Research in engineering design, pp. 189–209.

Wang, L., Shen, W., Xie, H., Neelamkavil, J. and Pardasani, A. (2002), "Collaborative conceptual design - state of the art and future trends", Computer-Aided Design, Vol. 34 No. 13, pp. 981 – 996.

De Weck, O.L., Roos, D. and Magee, C.L. (2011), "Life-cycle Properties of Engineering Systems", Engineering systems: meeting human needs in a complex technological world, MIT Press, pp. 65 – 96.

ACKNOWLEDGMENTS

This work has been sponsored by the French government research program "Investissements d'avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2007-2013 (ERDF – Auvergne region) and by the Auvergne region.