A MODEL FOR THE RECONFIGURATION OF MANUFACTURING SYSTEMS

Florent Frizon de Lamotte, Pascal Berruet and Jean-Luc Philippe

LESTER Laboratory - CNRS FRE2734
University of South Brittany
Lorient, France
lamotte@iuplo.univ-ubs.fr

Abstract: This paper proposes a state of the art about reconfigurable systems and underlines existing links in system reconfiguration in the domains of electronics, mobile robotics and manufacturing systems. Then a model for describing reconfigurable systems is introduced. In this model, described using MOF class diagrams, the architecture of the system and its configuration are separated. This model will then be usable by a reconfiguration manager.

Keywords: Reconfiguration, Manufacturing Systems, Electronic Systems, Recovery, Adaptation, Model

1. INTRODUCTION

Systems are traditionally designed to keep the same structure and behavior all along their lifespan. In case of failures, changes in objective or in environment, they may need external intervention to continue working.

In reconfigurable systems, choices concerning the organization of its elements can be delayed until exploitation and can be modified dynamically or not. This extends the application field of the system. Such systems have been studied in several domains as: electronics (Auguin, *et al.*, 2003), communications (Mitchell, *et al.*, 1998), control (Wills, *et al.*, 2001; Cotting and Burken, 2001), manufacturing (Combacau, *et al.*, 2000) or robotics (Kotay and Rus, 1999; Kamimura, *et al.*, 2001). Even if they target different applications, these systems share common concepts.

Implementation of the controllers in a reconfigurable manufacturing system is done on an electronic system. This electronic infrastructure can, in turn, be reconfigurable. Similarities can also be noted in the design methods used in manufacturing and electronics. One aspect of the design may be the reconfiguration. A common model would be profitable to handle the same concepts concerning reconfiguration in both kind of system, and thus, to take into account the specificities from the electronic system.

The objective of this paper is, through some examples, to find shared features of reconfigurable systems. These features are presented in section 2. Section 3, focuses more particularly on the use of reconfigurable electronics in reconfigurable manufacturing systems for which we introduce a common representation in section 4, this representation is illustrated with an example.

2. RECONFIGURABLE SYSTEMS

Some examples of reconfigurable system are presented in order to underline the common concepts they share and the characteristics of each kind of system (objective of reconfiguration, reconfiguration process).

2.1 Definitions and examples

A configuration corresponds to the way the components of a complex system are parameterized

and connected together. This configuration responds both to external requirements (user request, quality, environment) and internal requirements (productivity). Reconfigurable systems have the ability to go from one configuration to another along their exploitation. This evolution can be triggered by different causes such as a change in the objectives of the system, a lack of performance or a fault on one of its parts. These systems have been used in different application fields such as manufacturing, control, electronics, telecommunications or informatics and often share the same principles. The five following examples represent some aspects of the reconfiguration in these fields. They show how vast is the reconfiguration domain and which similarities there are between them.

The first example of reconfigurable system is an aircraft flight control system (Cotting and Burken, 2001) that has been developed at the NASA for the X-33 craft. This plane' trajectory is given by its height control surfaces which are redundant. A controller has been designed to accommodate the failure of one of them and to redistribute the control effort among the remaining surfaces. To do so, a reconfigurable mixer has been used. It takes the orders from the roll pitch and yaw command and calculates the positions of the surfaces. In case of a failure on one of these, the mixer is reconfigured. Here, the reconfiguration is triggered by a failure on the controlled system. It consists in the modification of the parameters of the controller. The decision is centralized.

The DJINN multimedia-programming framework (Mitchell, *et al.*, 1998) from the Distributed Systems Laboratory at the University of London is an example of reconfigurable system in the domain of multimedia communications. The aim is to guaranty quality of service (QoS) in communications through a distributed network of computers or mobile devices. In this network, nodes may be added or deleted dynamically triggering a reconfiguration. A configuration is expressed with paths between components. Reconfiguration of this system is done in two phases. A "setup" phase where the new components are created and an "integration" phase where they are started and connected to the system. Depending on the resources offered, the transition between the two configurations is more or less smooth (it depends on if the two configurations can "live" together in the system or not). The reconfiguration consists in the adding or removing of components and the modification of the connection between them. The decision of the reconfiguration and its control are centralized.

Another example, from the world of distributed computing is taken from the simulation of virtual worlds as used by the American army. In these systems, reconfiguration may be needed if a computer is disconnected from the network or if it cannot support the charge anymore (think about a computer that controls a bridge where two groups of people are heading). The Bullpen project (Welch and Purtilo, 1997) offers a reconfiguration component for simulators. In this component, three entities have been identified: the Scout detects events of interest in the system and informs the Coach which makes the decisions while the Scoreboard keeps a copy of the application state. The decision is centralized, but the system can handle multiple events at the same time (a thread is launched for each problem).

In manufacturing systems, Belabbas and Berruet (2004) propose an on-line reconfiguration method. This method is applied on a workshop constituted of five working areas. Each area can perform two distinct functions on products; some functions can be performed in multiple areas. In the initial configuration three working areas are operating. When a failure occurs on one of these areas, another configuration has to be found using the remaining ones in order to resume the operations. The objective for the reconfiguration is to finish the logical operating sequence (Toguyeni, *et al.*, 2003) after the occurrence of a fault in the system.

Last example is taken from electronic design. Auguin, *et al.* (2003) present tools for designing reconfigurable computing systems. In these systems, functions may either be in hardware of software. Hardware functions can be loaded in a RPU (Reconfigurable Processor Unit). A genetic algorithm is used off-line to find the best use of hardware and software functions. The goal of reconfiguration is to optimize the use of available resources. It is triggered when a needed hardware function is not loaded in the RPU.

These five examples come from rather different domains and show different aspects of reconfiguration. Reconfiguration is a way to react to event from inside or outside the system. It may also be a mean for optimizing the system. Configurations may be determined on-line or off-line. All the examples feature a transition phase between configurations. Objectives for the reconfiguration are studied in section 2.2. The determination of a new configuration and the transition between the old and the new configuration are part of the reconfiguration process, presented in section 2.3.

2.2 Objectives of reconfiguration

The objective of a reconfigurable system can be a safety one (Berruet, *et al.*, 2003) in which case quick decision has to be taken. This kind of response often takes place as a recovery procedure from a failure on the controlled process. This is the case for example in (Cotting and Burken, 2001) where the mixer has to be reconfigured quickly in response to the failure of a controlled surface.

Another objective supported by reconfiguration is to follow system requirements. These requirements may either be external (user requests, QoS, quality, environmental changes) or internal (productivity, consumption). This was the case in (Belabbas and Berruet, 2004; Mitchell, *et al.*, 1998; Welch and Purtilo, 1997) where reconfiguration enabled the system to continue its function and achieve its objective maybe under degraded performances. Such reconfiguration can be done after a recovery procedure where some components of the old configuration may not work as expected anymore. It can also be the result of a continuous audit on the performances of the system.

Reconfiguration can also be used to optimize the system as in (Auguin, *et al.*, 2003) where functions have to be hardware ones to achieve speed objectives but surface (size) constraints may be very high. As some functions may not be used at the same time, it is possible to load them on-demand through a reconfiguration.

2.3 The reconfiguration process

A characteristic of a reconfigurable system is the way it goes from a configuration to another. This is called the reconfiguration process. This process may be handled by an entity called the reconfiguration controller and can be broken up into two phases: the decision of the new state and its application.

The decision of the reconfiguration is the first step in the process and has been studied by Toguyeni, *et al.* (2003) and Belabbas and Berruet (2004). It has to determine which configurations meet the requirements for the resumption of the system. These configurations can either be determined off-line or on-line. One configuration among them is chosen to replace the current one. Along with the performances of the new configuration, levels of reconfiguration help choosing a new configuration as it characterizes the impact of the reconfiguration phase: the time it spends, the energy it consumes, the loss of partly processed products.

Then, the system goes to the new configuration. Reconfiguration can be performed as the system is working, through intermediate configurations, putting the products on their new way. It can also be performed after a shut down the system and an emptying of it. In manufacturing systems, working mode management (Kamach, *et al.*, 2003) is responsible for the coordination of the machines during the reconfiguration. Working mode management insures that the state of the whole system remains safe and coherent. Similar techniques are also used in other type of systems.

3. RECONFIGURABLE SYSTEMS IN MANUFACTURING AND THE ROLE OF ELECTRONICS

This section emphasizes the particularities of manufacturing systems concerning the use of electronics and shows that the same concepts may be found in both systems concerning reconfiguration.

3.1 Granularity in manufacturing systems

Manufacturing systems can generally be described from multiple levels of granularity. Fractal manufacturing systems (Ryu, *et al.*, 2003) takes into account this granularity by using a multi-level representation. Reconfiguration is done in the same way at each hierarchical level through self-organization of agents.

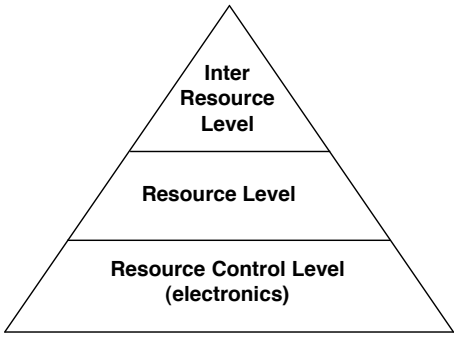

Fig. 1. Granularity levels in manufacturing systems

The decomposition of a manufacturing system presented on fig. 1 leads to the electronic system controlling it. This system may also be a reconfigurable one. Sharing the same concepts and models for reconfiguration between electronics and manufacturing would lead to a better mastery of the reconfiguration of the whole system.

3.2 Shared concepts between electronic and manufacturing reconfigurable systems

Both electronic and manufacturing systems perform operations on *products*. In manufacturing systems, products are the processed part whereas in electronic systems they are data. These products are transferred between the *stationary components* of the system, also called *resources*. Resources correspond to machines in manufacturing systems and to processors in electronics. *Transport components* transfer products between stationary components through the *ports* of these components. A conveyor in manufacturing or a bus in electronics plays a transport component role. A *function* is an action that can be made on a product. Functions are defined at the system level. In an electronic system, they are referred as tasks. An *operation* implements a function on a component.

Major differences come from the scale at witch the operations are made in each kind of system. The rate at which data is processed in an electronic system is

much faster than the one for the parts in manufacturing systems and manufacturing systems are much bigger than electronic systems. This induces that data processing and transfer in electronic systems goes as fast or faster than control while in manufacturing systems, parts generally go much slower than control that may be transferred through electronics.

Having a common model capable of representing both manufacturing and electronic reconfigurable system configuration would enable the use of the same model throughout the whole system. One such model is currently developed accordingly and is presented in the next section.

4. A MODEL FOR REPRESENTING CONFIGURATIONS

To represent and manipulate the concepts from reconfigurable systems, a model has been developed. This model has been inspired by the works made on configuration language in the domain of distributed computing (Kramer, 1990). The concepts used in the model are described by meta-models using Meta Object Facility (MOF) (OMG, 2002) class diagrams, which are close to UML class diagrams. Meta-models show the relations between different aspects of a configuration. A manufacturing system example depicts how the model is represented using a textual syntax.

4.1 Represented aspects

The model should represent the relations between its components and the products processed at one level of granularity. This model does not cover the details on the functions being used.

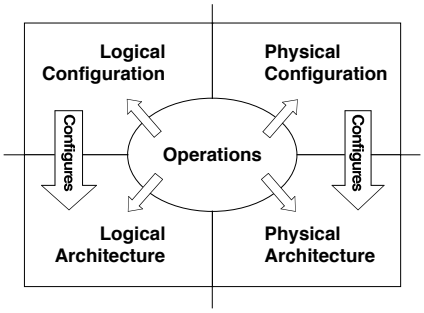

Fig. 2. Organization of the model

As illustrated on Fig. 2, a reconfigurable system may be broken up according to two axes. Horizontal axis concerns the separation between the architecture of the system and its configuration. The architecture consists of all the components (functions, or resources) constituting the system and their potential connections. It is presented in section 4.2. These components are parameterized and inter-connected through the configuration presented in section 4.3.

Vertical axis separates the logical subsystem from the physical one. The logical subsystem consists in the functions of the system and their associations to form logical operating sequences (Toguyeni, *et al.*, 2003). The physical subsystem consists in the resources and the transport between these resources. The physical subsystem provides the structure on which the logical subsystem is executed. Both the logical and the physical subsystems can be configured.

4.2 The logical and physical aspects of an architecture

The architecture, as described on Fig. 3, should represent all the potentialities of the system. It is broken-up into two parts: the logical architecture and the physical architecture.

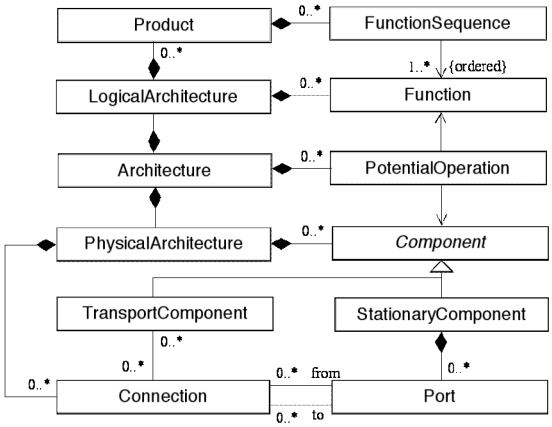

Fig. 3. MOF class diagram for the architecture

The logical architecture is constituted by *functions* that can be performed on the system. These functions are put together to form *function sequences*. These sequences are then applied on *products*.

Physical architecture contains the physical components of the system. These components can either be *stationary components* that perform stationary operations or *transport components* which function is to transfer a product from one stationary component to another through ports. *Ports* are interfaces belonging to stationary components on which products can be put or taken. The possibility of transport between two components is represented with an entity named *connection*. A transport component has the ability to realize connections. To realize a connection, it has to take a product from the source port of the connection and put it on the destination port.

Logical architecture is mapped onto physical architecture through *potential operations*, which link a function with a component.

Fig. 4 introduces a manufacturing example. Its architecture is described on Fig. 5 using a textual representation. This architecture is configured in the

next subsection. The logical architecture consists in the F1, F2 and F3 functions plus a generic transport function: FT. The physical architecture is constituted by four stationary components: IN, OUT, M1 and M2. There are also two transport components: Cv and R and six connections. Potential operations O[1-4] link the logical and physical architectures.

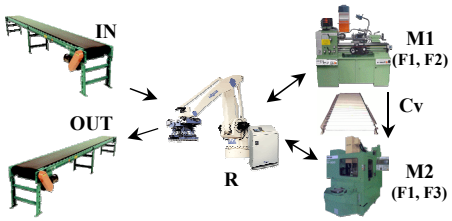

Fig. 4. Informal description of the example

```
logical architecture {
  function F1;
  function F2;
  function F3;
  function FT; -- transport function
}
physical architecture {
  stationary IN { port p : out; }
  stationary OUT { port p : in; }
  stationary M1 { port p : inout; }
  stationary M2 { port p : inout; }

  transport R realizes INM1, INM2, M1M2,
M2M1, M1OUT, M2OUT;
  transport CV realizes R1R2;

  connection INM1 (IN.p, M1.p);
  connection INM2 (IN.p, M2.p);
  connection M1M2 (M1.p, M2.p);
  connection M2M1 (M2.p, M1.p);
  connection M1OUT (M1.p, OUT.p);
  connection M2OUT (M2.p, OUT.p);
}
operation O1 (F1, M1);
operation O2 (F2, M1);
operation O3 (F1, M2);
operation O4 (F3, M2);
```

Fig. 5. Example of architecture

Next subsection presents how this architecture can be configured by instantiating the functions, the connections and the potential operations in the configuration.

4.3 Configuration of the architecture

The architecture, as presented at section 4.2, is split into physical and logical parts. As seen on fig. 6, configuration is also broken up according to the same two aspects.

The logical configuration is constituted by *functions instances*. While functions are defined as actions that can be made on products, a function instance realizes this function on the product. It is performed on a component through an *operation*.

Physical configuration is constituted by the components taken from the logical architecture and *connectors*. Connectors may, at first, be seen as instances of connections. They express the relation between the transport components used to transfer the product from its departure to its destination, the stationary components through which the product may have to go (these components act as controllers) and the connections used by the transport components to do the transfer.

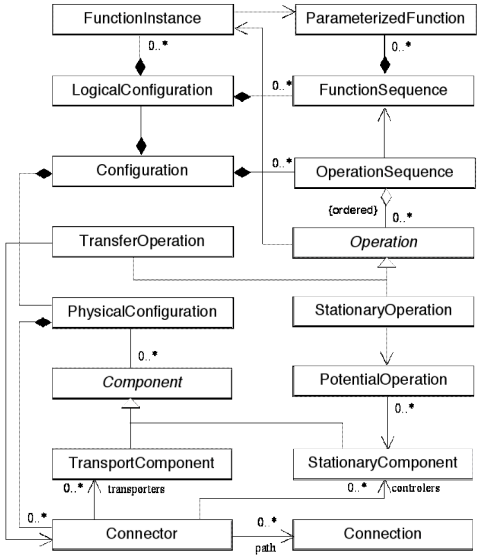

Fig. 6. MOF class diagram for a configuration

The correspondence between physical and logical configuration is done through operations. There are two types of operations. *Transfer operations* link a function instance to a connector. *Stationary operations* are the realization of a potential operation linked with a function instance. Operations are linked together to form operation sequences that realize a function sequence defined in the logical architecture.

```
configuration Config1 {
  logical configuration {
    instance IF1 of F1;
    instance IF2 of F2;
    instance IF3 of F3;
    instance TINM1 of FT;
    instance TM1M2 of FT;
    instance TM2OUT of FT;
  }
  physical configuration{
    connector C1 { transfer INM1 using R; }
    connector C2 { transfer M1M2 using Cv; }
    connector C3 { transfer M2OUT using R; }
  }
  transfer operation TO1 : TINM1 using C1;
  transfer operation TO2 : TM1M2 using C2;
  transfer operation TO3 : TM2M3 using C3;
  stationary operation SO1 : IF1 using O1;
  stationary operation SO2 : IF2 using O2;
  stationary operation SO3 : IF3 using O4;
  operation sequence OS1 realize P1_S1 :
              TO1, SO1, SO2, TO2, SO3, TO3;
}
```

Fig. 7. Example of configuration

Fig. 7 features an example of configuration for the architecture presented on fig. 5. Under this configuration, the system performs the F1, F2 and F3 functions successively. These three functions have been instantiated under the names IF1, IF2, IF3. Three other functions have been instantiated for

transfer operations between components: TINM1, TM1M2, TM2OUT. Functions F1 and F2 are executed on R1 through operations SO1 and SO2. F3 is executed on R2 through SO3. Connectors have been created for using Cv to transport products between IN and M1, M1 and M2, M2 and OUT. Three transfer operations use these connectors.

5. CONCLUSION

In this paper, the concept of reconfigurable systems, which had been studied in different fields of research, has been generalized. A model has been introduced to represent concepts shared both by electronic and manufacturing reconfigurable systems. Using this model, the same high-level representation may be used to represent reconfigurable electronic and manufacturing systems. As manufacturing systems generally embeds electronic systems for communication and control, same model may be used for representing different levels of a reconfigurable plant. This model may also help in sharing mechanisms, strategies and tools between electronics and manufacturing.

Concerning the model, it will evolve to support hierarchic configurations where a configuration may be decomposed in sub-configurations. This will enable local reconfiguration that will not have consequences on the remainder of the system. Other representations for the model, using graphical diagrams showing different views on the system are also being studied.

REFERENCES

Auguin, M., K. Ben Chehida, J.P. Diguet, X. Fornani, A.M. Fouilliart, C. Gamrat, P. Kajfasz and Y. Le Moullec (2003). Partitioning and CoDesign tools & methodology for Reconfigurable Computing: The EPICURE philosophy. In: *The Third International Workshop on Systems, Architectures, Modeling Simulation SAMOS03*.

Belabbas, A. and P. Berruet (2004). FMS Reconfiguration Based on Petri nets Models. In: *IEEE Int. Conference on System Man and Cybernetics SMC2004*.

Berruet, P., T. Coudert and J.L. Philippe (2003). Integration of Dependability Aspects in Transitic Systems. In: *IMACS-IEEE Int. Conference on Computational Engineering In Systems Applications CESA2003*.

Combacau, M., P. Berruet, E. Zamaï, P. Charbonnaud, A. Khatab (2000). Monitoring and Supervision of Manufacturing Systems. In: *IFAC MCPL2000*, pp. 348-353.

Cotting, C. and J.J. Burken (2001). Reconfigurable Control Design for the Full X-33 Flight Envelope. In: *AIAA Guidance, Navigation and Control Conference*.

Kamach, O., L. Pietrac, E. Niel (2003). Multi-model approach for discrete event systems : application to operating mode management. In: *IMACS-IEEE Int. Conference on Computational Engineering In Systems Applications CESA2003*.

Kamimura, A., S. Murata, E. Yoshida, H. Kurokawa, K. Tomita and S. Kokaji (2001). Self-Reconfigurable Modular Robot. In: *IEEE/RSJ Int. Conference on Intelligent Robots and Systems IROS2001*, pp. 606-612.

Kotay, K. and D. Rus (1999). Locomotion Versatility through Self-reconfiguration. In: *Robotics and Autonomous Systems*, Vol. 26-2,3, pp. 217-232. Elsevier.

Kramer, J. (1990). Configuration Programming – a framework for the development of distributable systems. In: *Proceedings of the IEEE Int. Conference on Computer Systems and Software Engineering CompEuro90*.

Mitchell, S., H. Naguib, G. Coulouris and T. Kindberg (1998). Dynamically Reconfiguring Multimedia Components: A Model-based Approach. In: *8th ACM SIGOPS European Workshop*.

Object Management Group (2002). OMG Meta Object Facility (MOF) Specification. *http://www.omg.org*

Toguyeni, A.K.A., P. Berruet, E. Craye (2003). Models and Algorithms for Failure Diagnosis and Recovery in FMSs. In: *Int. J. of Flexible Manufacturing Systems*, Vol. 15-1, pp 57-85. Kluwer.

Welch, D.J. and J.M. Purtilo (1997). Domain-Driven Reconfiguration in Collaborative Virtual Environments. In: *Univ. of Maryland CS technical reports*. CS-TR-3772.

Wills L., S. Kannan, S. Sander, Mu. Guler, B. Heck, J.V.R. Prassad, D. Schrage and G. Vachtsevanos (2001). An Open Platform For Reconfigurable Control. In: *IEEE Control Systems Magazine*. Vol. 21, pp. 49-64.