Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2014, *6*, 1771-1778

An Aggregation Modelling Method Based on Generic Component for Reconfigurable Control

Hexuan Hu[1,2,*]

[1]*Agricultural and Animal Husbandry College of Tibet University, Lin-zhi, Tibet, 860000, P.R. China*

[2]*Energy and Electrical Engineering College of Hohai University, Nanjing, 210024, P. R. China*

Abstract: In this paper, we propose an approach based on generic component model to realize the model aggregation for reconfigurable control. This is a component –oriented method. Two elementary notions, the service and the operating mode, are introduced to construct a hierarchical system and to assure the coherences between specification and realization at each level and between levels. The consideration of aggregated, complex components not only leads to extend this description to the possibilities of reconfiguration but also provides a unified framework for facilitating the knowledge acquisition of reconfiguration.

Keywords: Model aggregation, Generic component model, Service, Operating mode, Reconfigurable control.

1. INTRODUCTION

Reconfigurable control considers the problem of automatically changing the control structure and the control law after a fault has occurred in the plant [2]. The Generic Component Model (GCM) describes components from a point view of the users, who receive services and can use them differently in different operating modes [4]. A formal analysis of the reconfiguration has been described by using GCM [5-6]. Component interconnections are taken into account by considering higher level components, which result from the aggregation of lower level ones. Choukair and Bayart present an approach of modeling of a distributed architecture based on the Cartesian products of components [3].

As mentioned in these researches above, the reconfigurable control based GCM relies on pre-designed alternative control structures. The main obstacle to perform these reconfigurable tasks is the combinatorial explosion of the products of numerous components, particularly for pre-designing alternative control structures in a large-scale system. The model aggregation is a useful method by which a large-scale system is built as a hierarchical system composed of interacting subsystems [1]. Each subsystem or module involves only a few components and is therefore easy to design, analyze and maintain, but a global objective of the system must also be maintained so that the low modules can be coordinated to attain certain desired objectives.

In this paper, our goal is to realize model aggregation for reconfigurable control and to present a systematic procedure for the model aggregation of a hierarchical system. This procedure finds its justification from the coherence it achieves in a level and between levels, unifies the two elementary notions service and the operating mode during the model aggregation, and produces the important information about reconfigurable control.

The paper is organized as follows. The example used to illustrate the different notions is first presented (section 2). Next, we define the notions of generic component model and present the necessary notions, the services and the operating modes in section 3. In section 4, we present an aggregation procedure for a hierarchical system. In section 5, we point on the reconfiguration aspect. The main result is illustrated on two tank system in section 6. In section 7, we provide a brief summary of the main results of this paper.

2. EXAMPLE

The chosen example is a part of a level regulation process. It is composed of two identical connected tanks (see Fig. **1**). Each tank is cylindrical of section A. The inflow Q_1 is provided by pump P_1, (controlled by the signal of level) filling tank T_1. The pump is continuous on a specific range. The flows Q_a, resp. Q_b between the two tanks are controlled by valves V_a, resp. V_b, connecting pipes are at level 0 and 30 cm. Valve V_o which is always opened is an outlet valve, located at the bottom of tank T_2. All the valves are on/off valves.

Tank T_1 is equipped with a continuous level sensor and tank T_2 with two discrete level sensors, indicating if the liquid is above or below the sensor level. Liquid is led into tank T_1 by pump P_1 and from tank T_1 to tank T_2 by valve V_a (V_b should always be closed in the nominal behavior). Some normal operating modes are considered: preparation, regulation, and emptying. During the regulation operation mode, the main objectives are to keep the liquid levels to 50 cm in tank T_1 and to 10 cm in tank T_2.

1874-4443/14

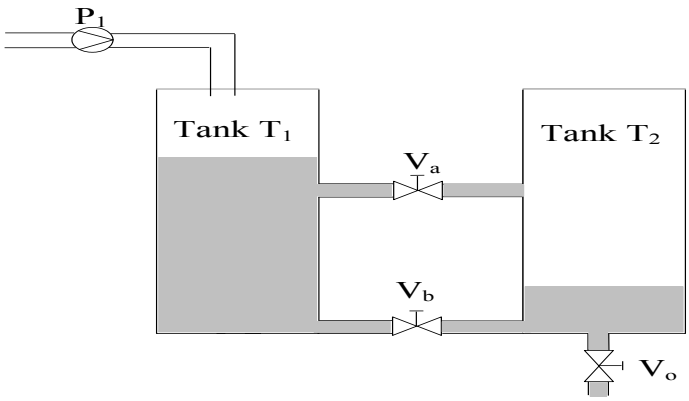


Fig. (1). Hardware process description of two tanks system (TTS).

3. THE GENERIC COMPONENT MODEL

This section introduces the two main notions of the GCM: the services and the operating modes which consider firstly the components of the lowest level called elementary components (components that can not be decomposed into other components).

3.1. Service

From the user viewpoint, a system component provides one or several services. A service is defined as a procedure whose execution results in at least one modification of its output interface.

Inputs, outputs and procedures: A service is first described by the variables it consumes (Cons), the variables it produces (Prod), and a procedure (Proc) which transforms the former into the latter.

Requests: A service is run either on the reception of its specific request (for example, close, open for the valve) or permanently in time without any specific request presented to the component (for example: the storage service which is systematically provided by the tank, at all times and whatever the values of the inputs and outputs). A request (implicit or not) will be noted rqst.

Resources: The realization of a service rests on hardware/software resources (a tank with no leak for the storage service, a non faulty sensor for the measurement service ...). The service cannot be delivered when any of these hardware / software elements is not running properly; this is why they are called resources. Therefore, the model associates with each service the set of resources (res.) that are necessary for its normal running.

Set of services of a component. Formally summarizing the set of services provided by a component is defined as follow.

Definition 1: The set of services associated to a component is:

$S(k)=\{s_i(k),\ i \in I_s(k)\}$

$s_i(k)=<cons_i(k), prod_i(k), proc_i(k), rqst_i(k), res_i(k)>$

Where $S(k)$ is the set of services of component k, I_s the set of indices of the possible services, and the others are straightforward.

For the TTS example, the following services provided by the elementary components will be considered (see Table **1**).

Table 1. Elementary Services of TTS

Elementary Components	Service
Tank 1 and 2	T_i_store
Valves a and b	V_i_open, V_i_close
Pump	$Deliver\_Q_i$, $Stop\_Q_i$

3.2 Operating Modes

The notion of Operating Mode (OM) allows organizing the set of services into coherent subsets taking into account these two following requirements:

1. at a given time only a part of the services provided by a component are required to achieve the objectives linked to this component (for example, some objectives are regulation, initialization …).

2. For safety reasons, incompatible services (for example, initialization and production services) must not be run simultaneously.

So, the notion of operating modes can be used to ensure that the aggregated model is effective in the sense of achieving specifications given either for the higher level model or for the lower level one.

Definition of operating mode: An operating mode is a subset of services of a component. The set of operating mode covers the set of services, i.e. each service belongs at least to one operating mode, and each operating mode contains at least one service.

Definition 2: An Operating Mode (OM) is defined by two elements:

1. One or several objectives to be achieved $O_j = \{o_j\}$.

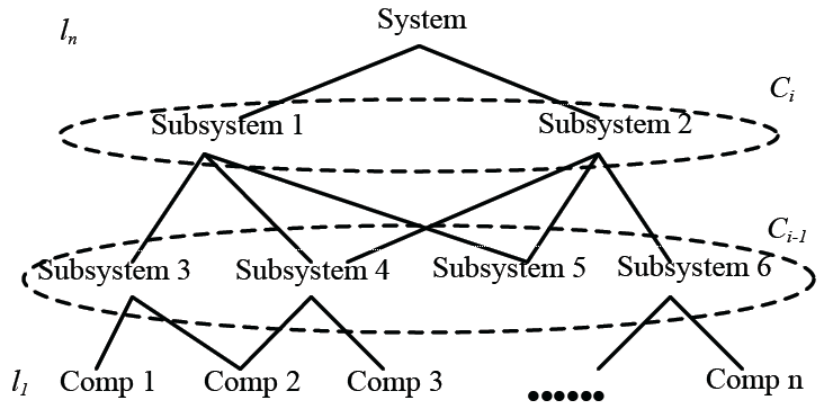


Fig. (2). Pyramidal structure of a system.

2. A subset of the services of the component $S_j \subset S$ allowing the realization of the objectives.

The automaton of operating modes: Note that a component or a system is always in one and only one OM at the execution time. So the OM of a component can be described by a deterministic automaton given by definition 3.

Definition 3: The operating mode automaton associated to a component is defined by *A (M, T, m°)* where:

$M = \{m_i\}$ is the set of the OM, each of them being a vertex of the automaton,

$T = \{t_{ij}\}$ is the set of the transitions, each of them being defined by $t_{ij} = \{m_i, m_j, c_{ij}\}$ where m_i is the origin OM, m_j is the destination OM and c_{ij} is the firing condition defined from the requests associated to the services belonging to the destination OM.

$m^\circ \in M$ is the initial mode, i.e. the mode where the system stays at its initialization.

The conditions of transitions are very important, but are not shown here because they are not the central subject of this paper.

3.3. Services Management

Nominal and degraded services are the key two notions. The execution of a service needs a set of resources, which includes hardware resources (sensors, memory etc.) and information resources (data with appropriate freshness status). These resources may belong to the intelligent vice or may be external. At a given instant, the service will run in a nominal way if the set of the resources it needs are able to perform normally. Unfortunately, this is not always the case, and it may happen that some of the required resources are faulty. The problem of tackling faulty resources first introduces the problem of evaluating the resource quality at each instant of time. Note that the notion of a quality index associated with a given resource has to take into account the fact that each of the services whose execution needs that resource might have a different quality requirement. To introduce such a possibility, we distinguish two levels in a resources management procedure. At the first level, we assume that some index can be computed in order to characterize each resource state. At the second level, each service takes into account the set of the indexes that are associated with the resources it needs, in order to adapt its operation.

As soon as the resource can no longer be used by the service, it is labelled as faulty, and one has to analyze further. Note that a given resource might be non-faulty for some service and yet faulty for another one. In the case of a faulty resource, the service can no longer run in a nominal way, and two situations may occur. The first situation is that in which the intelligent instrument designer has implemented no replacement procedure for the service. In that case, the service becomes unavailable, and it should be taken away from the list of the services of the different OM in which it appears. The question now arises to decide whether such an OM keeps some sense in spite of the absence of this service. The second situation is a fault-tolerant one, in which the designer has implemented at least one replacement procedure to perform the service. When the resources that this replacement procedure needs are non-faulty, it can be run instead of the nominal one at each request. The list of the available services in the different OM remains unchanged, but the intelligent device operation is degraded.

So, introducing the notion of a degraded service increases the robustness and thus the availability of the intelligent instruments, since the faults that appear in some of the resources they need do not necessarily interrupt the services they render.

4. BUILDING SYSTEMS FROM COMPONENTS

System architectures can be described at different hierarchical levels. Sensors, actuators, process components are at the field-level. High level components can be built from the aggregation of lower level ones at any hierarchical level.

The decomposition of a system to several subsystems and so on until the elementary components can be represented by a pyramidal architecture. Since it may be advisable to allow some components to belong to several subsystems, the structure is not purely hierarchical but is a pyramidal one (Fig. **2**).

In a pyramidal architecture, each component of level *l-1* belongs to at least one component of level *l* and any component of level *l* includes at least one component of level *l-1*.

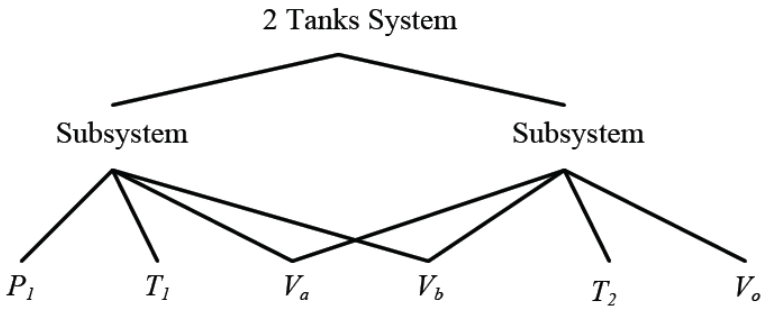


Fig. (3). TTS pyramidal structure.

To establish the pyramidal structure of a system, the functional viewpoint may be adopted. For example, the TTS will be decomposed into two subsystems, each subsystem grouping the elementary components allowing the level regulation in one of the tank (Fig. **3**).

4.1. Procedure of Aggregation

Let $S_\alpha(a)$, $O_\alpha(a)$ (resp. $S_\beta(b)$, $O_\beta(b)$) be the services and the objectives associated to the component a being in the mode α (resp. the component b being in the mode β). $S_\alpha(a)$ (resp. $S_\beta(b)$) allows the realization of $O_\alpha(a)$ (resp. $O_\beta(b)$).

Let c be a component of level l which aggregate the components a and b of level l-1. The combination of operating modes α and β, if it is consistent from a view point of user, provides a service of the component c. More generally, we will say that the set φ of the potential services of the component c is the Cartesian product of the operating modes offered by a and b.

$$\varphi = \{\Delta = \alpha \times \beta, \text{ such as } \alpha \in M(a), \beta \in M(b)\} \quad (1)$$

Of course, not every combination of lower level OMs is significant or allowed in real practice, for example associate a *Test_OM* with a *Regulation_OM* may be not relevant for the application realization, and such a combination should be removed from φ. Moreover there may exist combinations which have the same functional interpretation, they constitute versions of the same service and provide fault tolerant control perspectives.

Removing irrelevant services from φ allows specifying the set of relevant services φ_r which has to be organized into consistent OMs. Note that it is designers who indeed determine φ_r from φ and structure φ_r into OMs.

Defining φ and φ_r and structuring φ_r into OMs constitute the three steps of the aggregation procedure.

This procedure finds its justification from the coherence it achieves in a level and between levels. Removing irrelevant services from the set of possible combination of lower level components' OMs and organizing the set of relevant services into OMs allows guaranteeing the coherence between the available services and the objectives to achieve. In other words, it is a mean to express the relation of "what the aggregated component could do" and "what it should do".

Defining the services of a higher level component from the OMs of the components it aggregated and then structuring them into OM through the objective notion allows taking into account a specification given in a hierarchical way, assuring a coherence between levels and decreasing the number of combinations at each level of aggregation.

We can say that the operating mode is a bridge connecting the services available and the objective to achieve in the current level and is also a bridge joining the lower levels and the higher ones in a consistent way. Relations between OMs, objectives and services through the different levels are expressed by the Fig. (**4**).

4.2. Aggregation of Services

In Fig. (**4**), we can see that an OM is a set of services allowing the realization of the objectives associated to this OM and a service can be provided by a combination of the OMs associated to the components of lower level. Each of them can transform his role into another one in different levels.

How to succeed this transformation? Firstly, We note the t^{th} service of the component k at level l as $S_t(k)\_l$, which is a combination of the operating modes associated with n components at level l-1. It is represented as a element of $\varphi(k)\_l$, where $\varphi(k)\_l$ is the production, such as $M(1)\_l\text{-}1 \times M(2)\_l\text{-}1 \times \ldots \times M(n)\_l\text{-}1$ and $M(k)\_l\text{-}1$ is a set of $S_i(k)\_l\text{-}1$.

As the definition in section 3.1, $S_t(k)\_l = <cons_t(k)\_l, prod_t(k)\_l, proc_t(k)\_l, rqst_t(k)\_l, res_t(k)\_l>$, if let $M_i(k)\_l\text{-}1$ is the i^{th} operating mode of $M(k)\_l\text{-}1$, $/M(k)/$ is the number of OM in this set and $/M_i(k)/$ is the number of services of i^{th} mode in $M(k)$, then we can obtain a service $S_t(k)\_l$ from the following procedure:

For i_1 = 1 to /M(1)_l-1/

For i_2 = 1 to /M(2)_l-1/

……

For i_n = 1 to /M(n)_l-1/

(There are n components at level l-1 forming one component at level l and several modes in each component.)

$\{Cons_{i1i2\ldots in}(k)\_l = 0$

For j=1 to n

$\{m = i\_j$

For s=1 to $/M_m(j)\_l\text{-}1/$

$Cons_{i1i2\ldots in}(k)\_l = cons_{i1i2\ldots in}(k)\_l \cup cons^s{}_m(j)\_l\text{-}1\}\}$

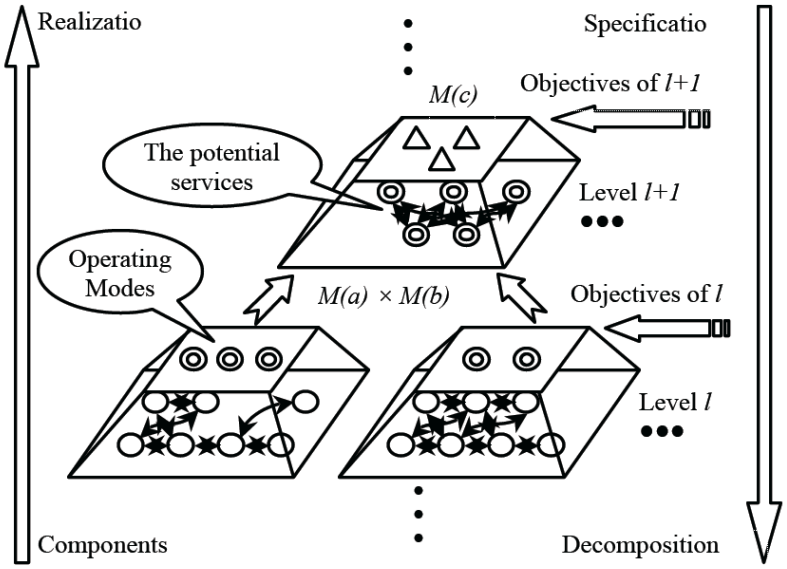


Fig. (4). The aggregation of a hierarchical system.

(Here, $cons^{s}{}_{m}(j)\_l\text{-}1$ is the variables consumed by the s^{th} service in m^{th} OM of j^{th} relevant component at level *l-1*.)

End

As the same procedure, we obtain $prod_t(k)\_l$, $proc_t(k)\_l$, $rqst_t(k)\_l$ and $res_t(k)\_l$. In fact, expect for level 0, the services of level *l* result from the Cartesian production of the operating modes at level *l-1* as mentioned above. So we can infer the signification of this service from variables it consumes and produces.

5. RECONFIGURATION ASPECT

The realization of a higher level service is reliant on the resources that allow the realization of the lower level ones it needs. The set of the hardware resources of a higher level service is the union of the sets of the resources of the lower level ones it needs. At the lowest analysis level, the hardware resource of a service provided by a component is the component itself. For example, the resource of the "open" service of a valve is the valve itself. Consequently there is only one version for a service of the lowest analysis level. Higher level services may have several versions due to the possibility to attribute the same functional interpretation to distinguish combinations of low level OMs. Each version is characterised by a different set of resources. In this case, the system reconfiguration may be possible in case of failure.

When a hardware resource is faulty, one or several services of the lower level become unavailable and it is possible that some others become active permanently in time. This is the case, for example of a blocked and closed valve. The opening service of this valve is unavailable and the closing service is permanent in time. Consequently to the loss of lower level services, some OMs are not reachable. For a given fault, several cases can be distinguished when one or several OMs disappear.

1. The unreachable OMs are not implied in the high level services allowing the realization of the system's current OM objectives. The system behaviour is not directly influenced by the fault.
2. The high level services allowing the realisation of the system's current OM objectives cannot be provided under their nominal version
3. There are other versions allowing the realization of these services. The system reconfiguration is possible.
4. There is none version allowing the service realisation. The only possible reconfiguration is changing the system aims.

6. APPLICATION TO THE TTS

6.1. Model aggregation

The specification of TTS is firstly given in a hierarchical way by Fig. (**5**).

The operating modes of elementary components are given by Table **2**.

Table 2. The Operating Modes of Elementary Components

Component	OM	Services	Objectives
Tank	1	T_i_store	Store the flow
Valve	1	V_i_open	Pass the flow
	2	V_i_close	Cut off the flow
Pump	1	$Deliver\_Q_i$	Offer the flow
	2	$Stop\_Q_i$	Stop the flow

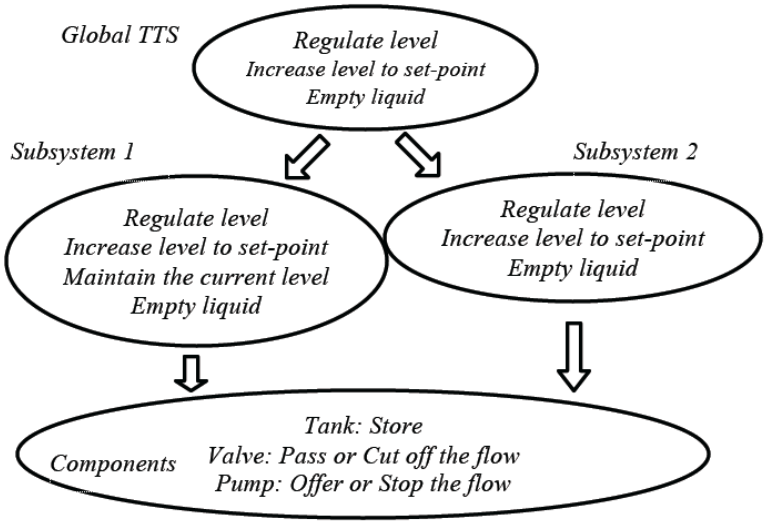


Fig. (5). The hierarchical specification for TTS.

The elementary components are those that can not be decomposed into other components. Each operating mode in these components has only one service and is associated with an objective. For simplicity reasons, the same names for these services and OMs are chosen.

Following the pyramidal decomposition of the TTS given by Fig. (**3**), the Cartesian product of the OMs of the elementary components P_1, T_1, V_a and V_b is calculated to define the potential service set of the subsystem 1 (see Table **3**). The three relevant services $H_1\rightarrow$ (to maintain), $H_1\uparrow$ (to increase) and $H_1\downarrow$ (to decrease) can then be defined for the subsystem1 and $H_1\downarrow$ is defined under three versions (see Table **4**).

Table 3. The Table of Services Combinations of Subsystem 1.

T_1	P_1	V_a	V_b	Service
T_1_store	Stop_Q_1	V_a_close	V_b_close	$H_1\rightarrow$
		V_a_close	V_b_open	$H_1\downarrow$
		V_a_open	V_b_close	$H_1\downarrow$
		V_a_open	V_b_open	$H_1\downarrow$
	Deliver_Q_1	V_a_close	V_b_close	$H_1\uparrow$
		V_a_close	V_b_open	$H_1\downarrow$
		V_a_open	V_b_close	$H_1\downarrow$
		V_a_open	V_b_open	$H_1\downarrow$

The relevant service set is then organized into OMs which are consistent with the objectives to achieve in this subsystem 1 level. These operating modes are shown in Table **5**.

Table 4. The Versions of Services for Subsystem 1.

Versions		Lower OMs	Resources
$H_1\rightarrow$	V_0	T_1_store, Stop_Q_1 V_a_close, V_b_close	T_1 ,P_1 V_a, V_b
$H_1\uparrow$	V_0	T_1_store, Deliver_Q_1 V_a_close, V_b_close	T_1 , P_1 V_a, V_b
$H_1\downarrow$	V_0	T_1_store V_a_open, V_b_close	T_1 V_a, V_b
	V_1	T_1_store V_a_ close, V_b_ open	T_1 V_a, V_b
	V_2	T_1_store V_a_ open, V_b_ open	T_1 V_a, V_b

Table 5. The Operating Modes of Subsystem 1.

OM	Services	Objectives
Empty	$H_1\downarrow$, $H_1\rightarrow$	Empty liquid
Preparation	$H_1\uparrow$, $H_1\rightarrow$	Increase level to Set-point
Regulation	$H_1\uparrow$, $H_1\rightarrow$, $H_1\downarrow$	Regulate level
End	$H_1\rightarrow$	Maintain the level current

As the same procedure, we can also obtain the operating modes of subsystem 2 (see following Tables **6**, **7** and **8**).

Table 6. The Table of Services Combinations of Subsystem 2.

T_2	V_o	V_a	V_b	Service
T_2_store	V_o_open	V_a_close	V_b_close	H_{2_below}
		V_a_close	V_b_open	H_{2_above}
		V_a_open	V_b_close	H_{2_above}
		V_a_open	V_b_open	H_{2_above}

Table 7. The Versions of Services for Subsystem 2.

Versions		Lower OMs	Resources
H_{2_below}	V_0	T_2_store,V_o_open V_a_close, V_b_close	T_2 , V_o V_a, V_b
H_{2_above}	V_0	T_2_store,V_o_open V_a_close, V_b_open	T_2 , V_o V_a, V_b
	V_1	T_2_store,V_o_open V_a_ open, V_b_close	T_2 , V_o V_a, V_b
	V_2	T_2_store,V_o_open V_a_open, V_b_open	T_2 , V_o V_a, V_b

Table 8. The Operating Modes of Subsystem 2.

OM	Services	Objectives
Empty	H_{2_below}	Empty liquid
Preparation	H_{2_above}	Increase level to Set-point
Regulation	H_{2_above}, H_{2_below}	Regulate level

The aggregation of the subsystems 1 and 2 allows defining the TTS OMs (see Tables **9**, **10** and **11**). Note that combinations which are not significant in real practice are rejected.

6.2. Fault Scenario

To illustrate the reconfiguration on the TTS, we consider a fault scenario. Let suppose, the current OM be the Regulation one and the necessary services and OMs of nominal version given in a hierarchical way as Fig. (**6**).

Scenario: V_a blocked in the closed position.

Service V_a_*close* gets permanent in time and service V_a_*open* becomes unavailable. Therefore, the nominal version of $H_1 \downarrow$ and H_{2_above} becomes unavailable, but the degraded version, $\{H_1\downarrow$: T_1_*store, V_a_close, V_b_open*$\}$ and $\{H_{2\_above}$: T_2_*store, V_a_close, V_b_open*$\}$, remains available (ref. Table **4** and **7**). The OMs, Regulation_1 and 2, are not affected and the mission of regulation TTS can still be achieved using these degraded versions of services.

Table 9. The Table of Services Combinations of TTS

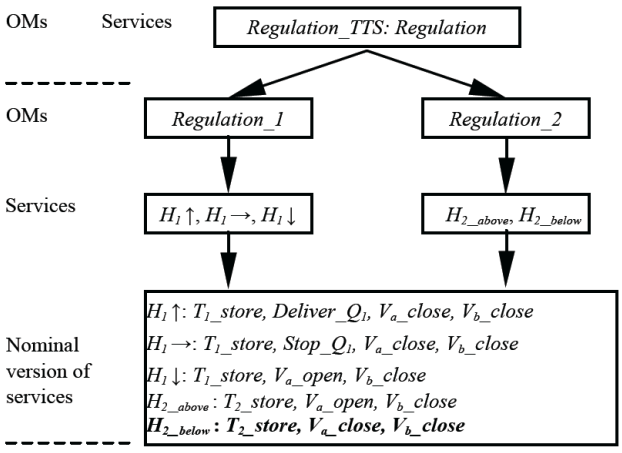
Subsystem 1	Subsystem 2	Service
Regulation	Regulation	Regulation
	Empty	Rejected
	Preparation	Rejected
Preparation	Regulation	Rejected
	Empty	Rejected
	Preparation	Preparation
Empty	Regulation	Rejected
	Empty	Empty
	Preparation	Rejected
End	Regulation	Rejected
	Empty	Empty
	Preparation	Rejected

Table 10. The Versions of Services for TTS.

Versions		Lower OMs	Resources
Regulation	V_0	Regulation_1 Regulation_2	Subsystem 1 Subsystem 2
Preparation	V_0	Preparation_1 Preparation_2	Subsystem 1 Subsystem 2
Empty	V_0	Empty_1 Empty_2	Subsystem 1 Subsystem 2
	V_1	End_1 Empty_2	Subsystem 1 Subsystem 2

Table 11. The Operating Modes of TTS.

OM	Services	Objectives
Empty	Empty	Empty liquid
Preparation	Preparation	Increase level to Set-point
Regulation	Regulation	Regulate level



Note: Level 0 is omitted for simplicity reason.

Fig. (6). The hierarchical description for regulation mission.

CONCLUSION

In this paper, a model aggregation procedure based on generic component model to reconfigurable control has been proposed. Two elementary notions, the service and the operating mode have been introduced to construct a hierarchical system and to assure the coherences between realization possibilities and given specifications not only at each level of the system decomposition but also between the levels of the decomposition. It is a mean to expose the relation of "what the aggregated system could do" and "what it should do". The important benefit of this procedure is that we can describe a system at any hierarchical level in a systematic way. Moreover it provides a unified framework for facilitating the knowledge acquisition of reconfiguration. To our future studies, there is an interesting problem that is introducing suitable mechanism for the diagnosis of large scale systems into this framework.

Finally, as the reconfiguration for autonomous systems fuses together researches from such diverse areas of AI as model-based reasoning, qualitative reasoning, planning and scheduling, execution, propositional satisfactory, concurrent reactive languages, Markov processes, model-based learning, and adaptive systems, the representation of the system model, a transition system, can be very easily extended by adding several variables to integrate these diverse researches above, as like [7-10] did.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

This work is supported by "The Nature Science Foundation of Tibet", "A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (Coastal Development Conservancy)", "Technology Foundation for Selected Overseas Chinese Scholar, Ministry of Personnel of China", "the Fundamental Research Funds for the Central Universities", and "he Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry".

REFERENCES

[1] K. Aström, P. Albertos, M. Blanke, A. Isodori, W. Schaufelberger, and R. Sandz, "Control of Complex Systems. Springer Publications, Germany, 2001.

[2] M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki, "Diagnosis and Fault-Tolerant Control," Springer Publications, Germany, 2003.

[3] C. Choukair and M. Bayart, "Application of External Model of intelligent Equipment to Distributed Architectures," in ISAS'99, Orlando (U.S.A.), pp. 329-335, Juillet 1999.

[4] M. Staroswiecki and M. Bayart, "Models and Languages for the Interoperability of Smart Instruments", *Automatica*, vol. 32, no. 6, pp. 859-873, 1996.

[5] M. Staroswiecki, and A. L. Gehin, "Analysis of System Reconfiguration using Generic Component Models," in Control'98, Swansea (UK), 1998.

[6] H. X. Hu, A. L. Gehin, and M. Bayart, "Model Aggregation for Reconfigurable Control Based on Generic Component Model," in ICSSSM'06, Troyes, France, 2006.

[7] H. X. Hu, A. L. Gehin, and M. Bayart, "A Formal Framework of Reconfigurable Control Based on Model Checking," in American Control Conference, Seattle, Washington, USA, 2008.

[8] H. X. Hu, A. L. Gehin, and M. Bayart, "An extended qualitative multi-faults diagnosis from first principles I: theory and modelling", In: 48th IEEE Conference on Decision and Control, China, 2009.

[9] H. X. Hu, A. L. Gehin, and M. Bayart, "An extended qualitative multi-faults diagnosis from first principles II: algorithm and case study," In: 48th IEEE Conference on Decision and Control, China, 2009.

[10] A. L. Gehin, H. X. Hu, and M. Bayart, "A self-updating model for analysing system reconfiguration", Engineering Applications of Artificial Intelligence, vol.25, Issue 1, pp.20-30, 2012.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014