Interactive Design of Integrated Systems

José Reinaldo Silva

University of São Paulo

Rua Prof. Mello Moraes, 2231, São Paulo, Brasil, +55-11-818-5545, reinaldo@usp.br

Abstract

The design of autonomous integrated discrete systems requires the anticipation of behaviour and co-operative actions among subsystems. Promising and already successful approaches to this kind of system always rely in sound formal methods, which can be applied once a conceptual design is provided. However, there is a gap between the achievement of a conceptual model and the remaining process, in part because integration and flexibility are currently used as vague concepts, and also because organisation aspects in conceptual design appear only informally.

We present a framework based on object oriented design methods that combine functionality and behaviour as separated and tight views of the artefact. Functional and behavioural models are treated as metaphors ruled by consistency requirements. The resulting schema has being implemented over the deductive database called ROL, from University of Regina.

Keywords

Interactive design, functionality, behavioural analysis, OO-design, integrated systems.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35390-6_58](https://doi.org/10.1007/978-0-387-35390-6_58)

L. M. Camarinha-Matos et al. (eds.), *Intelligent Systems for Manufacturing*
© IFIP International Federation for Information Processing 1998

1 INTRODUCTION

The design and modelling of discrete systems have been formally represented through frameworks based on Petri Nets, vector addition and graph theory. Such formal representations derive from a sound specifications. However, in the field of integrated and flexible discrete systems, the systematic development of sound design specifications that supports the modelling process is still an open problem.

The process of synthesis of a conceptual and functional model for discrete systems also brings a problem of showing the correctness of an abstract model in the first steps of the design process, which is generally addressed by simulation methods. Validation consists in analysing the system's behaviour from a mathematical model produced afterwards, when all the interaction and dependencies among subparts are already defined and can be compared with the initial description of the system. A match is scarcely found on the first tries what shows that frequently, a gap is generated between the functional model ("what the system is for") and the behavioural model ("what the system does"), when these views are developed separately.

The complexity of the problem increases if the target artefact is a system where the human participation is very important, as it is in anthropocentric and balanced systems (Camarinha, 95). In that case, the challenge of representing the occurrence of observed uncontrolled events is increased with the challenge of representing the expected reaction of operators and (human) clients.

We propose an object oriented representation based on different views of the artefact : the abstract view (AV), and the object view (AO). The AV models the behaviour, the interaction among subparts and between the system and its surroundings (in fact, there is not a real isolated system), while the AO is a conceptual and functional model of the same artefact. The modularity and detachment between functional and behavioural models suits the fact that an artefact (or subpart) can have different kind of behaviours and interactions with distinct elements of the environment while different objects can, under some conditions, present a similar behaviour. Thus, the AV/AO proposal adds interactive representation to partial parametric views of the artefact.

This interactive design and modelling is focused in the fusion between the top-down and bottom-up design approaches and in the conceptual and functional models. The AV/AO representation maps behaviours (AV's) to objects (AO's). The proposed framework not only incorporates this features but also allows that a design synthesis be made by refinement, by bottom-up aggregation or by a mix of both. The matching between behavioural and functional models is in fact a metaphor, in the sense that both implies some features of the same artefact or subsystem.

A simple prototype of this framework has been implemented on top of the deductive database ROL from the University of Regina, Canada. The choice of a deductive database is justified by the possibility of testing the framework to the design of large systems using queries and restrictions in a Prolog-like language. This will be particularly useful in the representation of functionality and behaviour as logic models.

2. THE AV/AO REPRESENTATION

The proposed framework is sound and generates a (behavioural) model associated to a flexible representation of organisational aspects and the expected behaviour of human operators in automated systems. Classical object oriented design (Booch, 91) (Gamma, 95) has almost all basic concepts to support the design of flexible and integrated systems with the remark that a formal presentation for object systems is not a consensus among researchers and designers.

The relationship between design specification and design model is represented in first order logic, that is, by using logical theories to establish and validate design specifications in any abstraction level. Functional and behavioural specifications can be represented using the same approach even if the last model address the interaction of the static functional model of the system. Behavioural models are a detached view of the general functionality and therefore a metaphor of a subset of sentences from its logical representation.

A behavioural model are represented by AV's while functional and conceptual models are represented by AO's. Both AV and AO structures are complex objects, that is, an object that besides its parametric attributes and methods has also two more blocks of data represented in first-order logic as shown in fig 1. For the AO, these blocks stand for the functional self-model and the architecture of the AO (called *Internal Architecture*) which stress the compositionality of the model (see fig. 1).

For instance, if we intend to represent the AV/AO model of a comfort package in an integrated and automated building application, the general schema would be as shown in Fig. 1.

The schema on the right in Fig.1 shows the AO (in a very abstract level) Thermal Comfort System with its attribute values and methods and two distinctive field blocks associated to its functional model and the nesting architecture, that is, the self-model (which describes the functionality) and a description of the aggregation of its components, respectively.

Fig. 1: Schematic representation of an AO (Thermal Comfort System) and an AV (Temperature Setting) which connects to it.

The schema on the left in Fig. 1 is the AV TSet (or Temperature Setting) which represents a connection between the AO just mentioned and another (human) AO : an user who intends to change the standard setpoint of temperature for its particular environment. Besides the normal attributes and methods, the AV has also a logic (abstract) description of its behavioural model as well as a list of the AO's it can reach, that is, those containing a functional model that gives support to its behaviour. The simple model above anticipate the existence of an AO connected to the AV for the temperature control, which models the role of the human user in the system, that is, how he/she can use the facilities¹.

¹ In the model of Fig. 1. the human agent is associated to an AO once he/she is supposed to be an information source, in other situations human agents could participate more actively in he process as a link among machine agents. In that case he/she should be modeled as an AV.

Notice that the reachability relation is asymmetric, that is, AO's do not know which AV to connect what is reasonable since AO's are functional objects and not have any interactive initiative (Cowan, 95). AV's, on the other hand, implies some dynamics in the exchange of messages and seeking of data sources.

AV's and AO's can be refined into lower level elements until the degree of abstraction is low enough to give rise to, for instance, a Petri Net model of the discrete control system. Once the behavioural and functional models are encapsulated by distinctive objects, it is possible that during the design process one of them be temporarily more refined while the other remains in a more abstract form. We call this a transient state that should evolve to a more feasible one where the behavioural and functional models can be matched by a metaphor. The validation process should not be performed over transient states.

3. METAPHORS AND COHERENCE RELATIONS

Metaphors are generated when a same semantic content is represented in two different languages. In practical cases each of these languages is used by different classes of speaker.

Indurkha (Indurkha, 87) proposed a formalization of metaphors based in Model Theory and putting a metaphor presentation as a semantic transference among domains. Although it shows some drawbacks (Guerrig, 89) and relay to much on consistence proof, it is an interesting framework to represent similes, that is, a special kind of metaphor, which is more suitable to the problem of mapping behavioural and functional model.

A metaphor is a semantic transference between two domains where:

Def 1.] A domain is a tuple $\langle V, f, Sd, S \rangle$ where

V is a set of symbols

$f: V \rightarrow \Omega$, is a mapping in the set of types Ω

$Sd \subseteq S$ is a set of derivations defining terms

S is a set of well formed formulas over the vocabulary $V = \langle V, f \rangle$

A mapping $V_1 \rightarrow V_2$ between two sets of symbols is said admissible if it preserves arity, that is, only predicates with the same number of (instance) arguments are mapped.

Def.2] Given two domains:

$D_1 = \langle V_1, f_1, Sd_1, S_1 \rangle$ and $D_2 = \langle V_2, f_2, Sd_2, S_2 \rangle$ and an admissible mapping $\phi: V_1 \rightarrow V_2$, a metaphor is a pair $\langle \phi, S \rangle$ where $S \subseteq S_1$ is a set of transformable sentences.

Def. 3] A metaphor $\langle \phi, S \rangle$ is said to be coherent if $S'' = \phi(S) \cup S_2$ is consistent.

A simile is a special kind of metaphor where $V_1 \cap V_2 \neq \emptyset$, that is, there is a common set of terms that are used in both domains. For instance, these domains could stand for a behavioural and a functional model of the same process or artefact.

Besides, some special symbols like “call”, “send-message”, “connect”, etc. (see example in Fig. 1) have a very defined meaning and have a coherence support in the other domain if the functor can be mapped to the empty string while its arguments are matched by an admissible extended mapping. An extended admissible mapping is the one that accepts higher order symbols (which arguments are procedures and functions) and apply the admissible mapping only to the arguments into the extended set $V_2 \cup M$, where V_2 are the symbols of the functional model and M is the methods specifications of the AO.

A metaphor $\mathcal{G} = \langle \phi^*, S \rangle$, where ϕ^* is an extended mapping, is said to be a coherent relation among an AV and an AO if $S'' = \phi^*(S_1) \cup S_2$ is consistent.

Complex objects mapped by metaphors are a base to represent a dual behavioural/functional model for integrated and flexible systems. In the next section we propose a definition for these concepts.

4. INTEGRATED AND FLEXIBLE SYSTEMS

Let $M_1, M_2, \dots, M_n$ be a set of independent functional modules loosely coupled that composes a large system. Each module has a unique functionality S_i , $i=1, \dots, n$. These modules are said to be integrated if and only if there exist at least one composed functionality $\Phi(m_1, \dots, m_k)$, $k \leq n$, over the set of modules.

$\Phi(m_1, \dots, m_k)$ is said to be a weak functional integration if it is only a simple composition of module functionality', that is, if

$$\Phi(m_1, \dots, m_k) = \prod_{i=1}^k S_i$$

Similarly, $\Phi(m_1, \dots, m_k)$ is a strong functional integration if it is the result of composition and also the sharing of data and information.

If each module M_i has a set of distinct functionality $\{S_i\}$ where at least one can be more salient in a specific situation the module is said flexible². A flexible system is the aggregation of flexible, distributed, independent and co-operative modules, that is, modules with a strong functional integration.

A flexible system is closed if it is defined to be composed by a choice of functionality's over a fixed base of modules. Conversely, the flexible system is said to be open if it is defined as a composition of choices of functionality over an open base of modules.

A module M_i that has internal control and supervisory capabilities is called an integron. An integron is a two level system architecture (Wonham, 88): a plant, the lower level, and a supervisory that contains the control system responsible for the integration and flexibility (Ramos, 96).

The definitions above characterise the kind of automated system we intend to model starting with a conceptual representation based in AV/AO. As an example let us look to the case of a very simple automated building system.

In Building Automation we find all the automation problems with flexible and co-operative systems without the implementation problems of manufacturing systems. General approaches to the design of such intelligent buildings require a mixed of top-down and bottom-up design, in the sense that some of the object parts are already known and reused from other projects, while other parts - principally the co-operative functions - must be drawn from scratch.

5. A SIMPLE EXAMPLE

As an example of the expressive power of a representation combining behaviour and functional/conceptual models we choose a small automated building facility. All the problems treating integrated and flexible systems can be found in domotic with the advantage that in this domain the components are easily identified.

For instance, the sharing of data among different sub-systems can be clearly shown by the Cupertino between a security system, that register the presence of individuals in the building, and the illumination system. Although these sub-systems have different functionality they use identical data. In a really integrated

² CNC machines for instance, can be used to different milling steps.

system such data should be shared instead of being redundantly collected by both systems.

The experimental site proposed is the set of two rooms as shown in fig. 2. Room 1 is supposed to be of public access, thus, there is no access control devices, only smoke detectors, presence sensors, temperature sensors, air conditioning and an alarm loudspeaker. The other room is private and thus it has an access control device, besides of other sensors and actuators. Most important, in this room, a privileged user should interact with the automated system granting access permissions, independently of the existence of other operators and a higher level supervisor.

Because of the difference in the functionality and in its interaction with human agents each room has its own supervisory system, which together with the Programmable Controllers (PC's), sensors and devices (what we call plant) compose an integron. Thus, the whole system is an integron composed of two subparts that are also integrons.

However the AV/AO representation of the system do not necessarily follows the physical layout showed in fig. 2. The functional architecture is depicted in fig. 3 and is concerned with the purpose of each sub-system, independently of the physical layout of the building.

Fig. 3 shows the first steps of the general architecture. The main objects (AO's) depicted stand for the main functions of the subsystems that would be implemented by several devices distributed all over the physical facility. Each of this AO's can have its functionality synthesised by a set of sentences as shown in Fig. 1. Subsystem's behaviours are represented by AV's according the following rule: *there is no interaction between AO's (machine, operators, or whatever) that is not modelled by an AV.*

Fig.2: Example to system integration technique in building automation.

Fig. 3 : General Architecture of the System2

After all the functionality is schematised in a top-down way - including the identification of shared functions or co-operative behaviours - the model is instantiated in a set of potentially reusable modules - the integrons of the system. The final system is then very close to a Petri Net model of the general control algorithm that will be the base to synthesise the supervisors.

The AV/AO representation has been implemented in a deductive database system called ROL (from Rule-based Object Language). We choose a database to implement the AV/AO representation, first because the present work is addressed to the design of large systems, and second because it is also our intention to show that the AV/AO models can be implemented in a mixed framework including objects and first order logic.

ROL language is based on objects and their attributes, and support also rule programs in a Herbrand model semantics (Liu, 97)(Liu, 97a)(Liu, 98). At the same time ROL is a typed language and supports the classical notion of schema. A powerful implementation for sets is implemented that can be handled decoratively besides being also object-oriented. Relationships are implemented as objects and can be nested in other relationships.

In the early versions (and also in the version 2.2 we are using) there is not an explicit representation for methods, and the encapsulation of complex objects are not fully implemented. That fact creates a problem for AV/AO representation since it makes a simple declaration of an AO, like the one in fig. 1, very large and cumbersome. For this reason we avoided including parts of the ROL implementation in this article.

However, the forthcoming 3.0 version has an explicit implementation for methods and also a full implementation for any kind of objects including complex objects. Thus, the model presented here can be easily implemented. In the new version we expect that queries will be simple programs that can match the interactions encapsulated in AV's giving feedback to the designer.

6. FURTHER WORK

As a further work we plan to improve the formal representation of simile and use a stronger framework in order to match behaviour and functional models in a representation more suitable to reusability. The present simile implementation is computable and reasonably efficient to be used in engineering design but relies too much in the mapping of symbols and in the fact that, both behavioural and functional models would use common symbols. This intersection of vocabularies is a key to make the search for a coherent mapping computationally affordable for practical applications.

Therefore the method is centred in the syntax of the representation more than in its semantic. On the other hand, reusability for the early design phase is likely to rely on semantic instead of syntax.

We are looking for an alternative framework to represent self models for AO's and the behavioural models of AV's that makes frozen definition of symbols (as "send-message", etc.) unnecessary, and that do not rely on the use of common symbols, besides offering a good basis for a semantic search for reusable components.

Another important point is that general specifications should be transformed in design implementations in a more detailed level. Thus, the general representation based in AV/AO must give place to a Petri Net of the control algorithm for the supervisors. Once the AV/AO model is based on objects and thus in static functional components it is a good question how efficiently it could be mapped in a Petri Net for large systems.

An object-oriented net is now being developed which can make the mapping from another OO-structure more likely to work efficiently for large and complex applications involving integrated and flexible systems.

7. REFERENCES

- Booch, G., (1991) Object Oriented Design with Applications, Benjamin Cummings.
- Camarinha-Matos, L. and Afsarmanesh, H. (eds.) (1995) Balanced Automation Systems II: Implementation Challenges for Anthropocentric Manufacturing, Chapman & Hall.
- Cowan, D.D. and Lucena, C.J.P. (1995) An Specification Concept to Enhance Design Reusability, IEEE Transaction on Software Engineering.
- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.
- Gerrig, R. (1989) Empirical Constraints on Computational Theories of Metaphor: Comments on Indurkha, Cognitive Science, **13**.
- Indurkha, B., (1987) Approximated Semantic Transference: A Computational Theory of Metaphors and Analogies, Cognitive Science, **11**.
- Liu, M., ROL: (1997) A Deductive Object Base Language, University of Regina, <http://www.cs.uregina.ca>.
- Liu, M., Guo, M., Yu, W. (1997a) The ROL System: User Manual, Release 2.2, University of Regina.
- Liu, M., (1998) An Overview of Rule-based Object Language, Journal of Intelligent Information Systems, **10**, 1-25.
- Pahl, G. and Beitz, W. (1988) Engineering Design : A Systematic Approach, Springer-Verlag, Berlin.

- Ramos, R.B.C., Silva, J.R., Miyagi, P.E. (1996) Supervisory Control of Integrated Building System: A Balanced Approach, BASYS'96, Proc., Chapman & Hall.
- Silva, J.R., (1992) Uma Abordagem Formal para Reutilização de Design Baseada em Metáforas : Sua Aplicação ao Design de Sistemas Discretos de Produção, doctor thesis, Universidade de São Paulo (in Portuguese).
- Silva, J.R., Afsarmanesh, H., Cowan, D.D., Lucena, C.J.P. (1995) An Object-Oriented Approach to the Design of Production Systems, in *Balanced Automated Systems: Architecture and Design Methods*, Camarinha-Matos, L. and Afsarmanesh, H. (eds.), Chapman & Hall, London.
- Silva, J.R., Miyagi, P.E., Sebastião, J.R., Rodrigues, M. (1996) A High Level Integrated Petri Net Environment to the Design of FMS, Proc. Of IASTED Int. Conf. on Modeling, Simulation and Optimization, May.
- Silva, J.R., Miyagi, P.E. (1996a) Towards a Formal Approach to Functionality in the Design of Discrete Flexible Systems, ASIC'96, ICIMS-NOE, Toulouse, June.
- Umeda, Y. et al. (1996) Supporting Conceptual Design Based on the Function-Behaviour-State Modeler, AIEDAM, Vol. 10, 4, September.
- Umeda, Y., Tomiyama, T. (1997) Functional Reasoning in Design, IEEE Expert, AI in Design, March-April.
- Wonham, W.M., Ramadge, P.J. (1988) Modular Supervisory Control of Discrete-event Systems, *Mathematics of Control Signals and Systems*, 1, 13-30.

8. BIOGRAPHY

José Reinaldo Silva has a PhD degree by the University of São Paulo in Computer Engineering in 1992 and is an assistant professor at the same university. His principal interest is in manufacturing and building automation.