How to Detect Risks with a Formal Approach? From Property Specification to Risk Emergence

Vincent Chapurlat, Saber Aloui

LGI2P - Laboratoire de Génie Informatique et d'Ingénierie de Production
Parc Scientifique Georges .Besse, Site EERIE de l'EMA
30035 Nîmes cedex 5, France

Abstract. The research work presented in this paper has two goals and is currently in progress. The first goal is to define a modelling framework allowing representing a system by using multi views and multi languages paradigms in a unified way and including knowledge and model enrichment by defining properties. The second goal consists to define some formal properties verification mechanisms in order to help a modeller to detect and to avoid dangerous situations and inherent risks which can occur to the system. The same mechanisms are then used to improve the quality of the representation which is the classical verification goal. This paper focuses on the set of formal properties modelling concepts and analysis mechanisms mainly based on Conceptual Graphs, which are proposed. The resulting approach is currently dedicated here to the risk assessment in healthcare organisations.

1 Introduction

A socio-technical system such as a hospital unit, a business process or a production line in a factory is a complex system where interact technical and human resources, flows and processes, etc. and which has to be considered under different points of view: the customer as the designer or the manager. Some of these interactions and points of view can be identified and then modelled because of their evidence or their necessity. However, some other interactions remain difficult to understand. They often make emerge some unpredictable and unforeseen events, unexpected behaviours and situations. Those emerging characteristics can have harmful consequences i.e. they induce risks. So, the willing to design, to improve or more simply to control more efficiently the behaviour of a socio-technical system during its life cycle needs to dispose of modelling and analysis techniques. It allows users to describe, to detect or to make emerge these risks, their causes and their possible consequences taking into account the complexity of the pointed out system. Analytical methods particularly for industrial plant [1] are mainly based on probabilities of occurrences and statistics of passed events. That is why these methods are still difficult to use in a predictive way. Simulation, for example, based on multi-agent paradigm, can be used in order to visualize auto-organisation capabilities of the system components and to make emerge some new organisation's behaviour. At last, there are few formal concepts of models analysis for the risk assessment, starting from system representation. The research work presented below intents to complete the existing

Chapurlat V. and Aloui S. (2006).
How to Detect Risks with a Formal Approach? From Property Specification to Risk Emergence.
In *Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 134-146
DOI: 10.5220/0002484201340146

tool box in risk assessment by using formal verification mechanisms for detecting and avoiding dangerous situations and inherent risks. A modelling framework has been developed and is briefly presented and argued in the following. Then verification mechanisms are described and illustrated considering the chosen application domain of healthcare organisations.

2 Requirements

The following summarizes modelling and analysis requirements which are at the origin of the proposed approach.

Considering the engineering system approach [2], a system must be modelled taking into account functional (What?), behavioural (How?) and structural (With what?) views and taking into account detail refinement and decomposition rules. So existing enterprise modelling languages [3], [4], approaches [5], and systemic concepts issued from the SAGACE method [6], [7] have been studied and discussed.

The three aspects described above, corresponding to a given point of view have to be coherent with the other ones i.e. to be interoperable and to share the same conceptual data model without any ambiguity. So it is necessary to define a unique meta model allowing to pass from one modelling language to another without any misinterpretation [8]

Different users are involved during modelling tasks taking into account their own objectives (to control, to reuse existing parts, to optimise a given criteria, etc.). Each user has then a point of view about the system which has to be coherent (multi view modelling) also with the other ones and avoiding different viewpoints. It is necessary to define a common and unique set of concepts and relations between the concepts, i.e. ontology [9]. This one gathers commonly used and shared terms by all users for describing the main characteristics of the pointed out system.

At last, some characteristics of the model must be checked in order to assume the quality i.e. the consistence, the completeness and the coherence of the different obtained models regarding the pointed out system [10].

3 System Modelling

Taking into account those requirements and the application domain of healthcare organisations, a modelling framework has been developed. A partial view of the modelling language meta model supported by the modelling framework is given in annex. This modelling language has been implemented by using a meta modelling environment called GME [11] allowing to dispose of a modelling tool.

This one unifies the modelling concepts and relations from a coherent and interoperable manner. The term unified means that all the concepts of the language result from the same meta model. So, there is not inherent problem of interoperability classically due to the use of several modelling languages, each one defining its own concepts and semantic. In parallel, a method guiding the modeller (not described in the continuation but more detailed in [12]) has been developed and is now under validation by end users.

The resulting modelling framework highlights five views under which the system has to be modelled:

- Functional view: The user describes the finality (Why does the system exist?), the mission (what does the system make? What are its functions, processes and flows?), and the objectives of the system (what are the efficiency, the level of stability and of integrity which have to be reached in order to fulfil this mission?). The used modelling language is inspired here by the KAOS [13], [14] modelling approach and the IDEF-0 [15] functional modelling language.

- Structural view: In this view, the goal is to define without ambiguity how the mission will be done (what are the processes, activities and flows?), and what is the organisation in charge or fulfil this mission (who/what is getting involved i.e. what are the resources? What are the involved organisation units – i.e. departments or services? What are their capabilities and skills? What are they doing?). The modelling paradigms used here are inspired by Binary Relational Model of NIAM [16], by the class concept from the class diagram of UML [17] for resource and organisation description and by the UEML language [5] in order to support capabilities, processes and activity description.

- Behavioural view: The system, due to the numerous interactions between resources, processes has a wide range of behaviours. The interrogations relate to what are the possible operational scenarios or the ones already encountered (what is the history of the system?). What are the states or situations reached of the system? (a situation is defined as a set of states from several entities). What are the events or conditions and their effects allowing the entities going from one state to another one? Finally, what are the functioning modes (nominal and non nominal) of the system? The used modelling languages are directly an extension of the eFFBD (enhanced Functional Flow Block Diagram) [18] for scenario and configuration description (i.e. behaviours in which several entities must be involved) and classical states diagram (for entities behaviour description).

- Ontological view: The definition of [19] "*ontology is a formal, explicit specification of a shared conceptualization*" indicates that ontology is the result of a consensus between actors concerning terms related to professional, management, decision and information aspects**.** Indeed, the system modelling calls upon several actors from various cultures (modellers, engineers, specialists in the field to study here pharmacist, doctor, nurses, etc.). Each actor or groups of actors has its own mental representation of the system and its components. This diversity of discourses universes may induce misinterpretations of sense and ambiguities on the chosen terms, and eventually a lack of knowledge for some group of actors. So, ontology allows defining a common discourse universe and then a description language of the domain shared by all actors as an Esperanto of the domain. Its design is made from a gradual manner all along the system modelling phase by the various groups of actors and by experts of the domain. It allows groups of actors integrating their own vocabulary and knowledge in the different modelling views and models. The result is of a higher level of confidence around the models and facilitates their analysis during the next phase.

- Property view: On one hand the property view allows users to enrich their knowledge and thus enrich by the same occasion the information already contained in each model coming from any view or any aspect. This is done by specifying properties highlighting for example a given scenario of execution of a process. On the other hand, it allows covering analysis requirements as shown in the following.

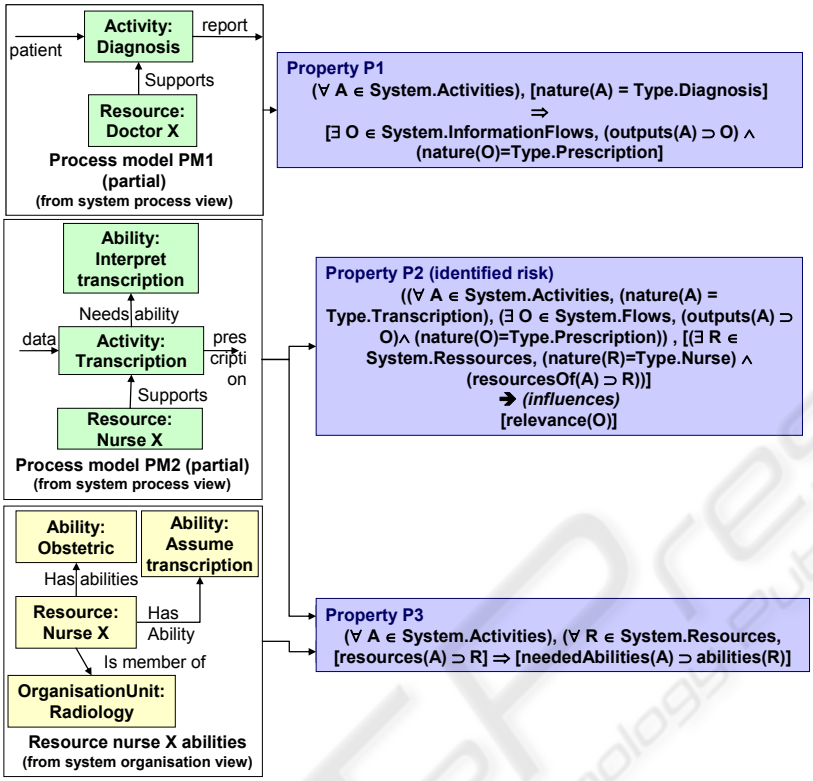

Fig. 1. Example of modelled Properties for each of the three kinds of requirements.

4 Property Modelling

As proposed before, the approach analysis is based on verification mechanisms. Taking into account the definition given for example in [20], verification must *provide rigorous arguments* (for example properties proofs*) in order to convince users of the correct functioning and reliability of a model and of model-based systems*.

In this case, it is not a question of complying with syntactic rules (already commonly verified by any modelling tool and without interest here) but of making sure that the semantics of the concepts and the relations contained into the meta model are strictly respected. The aim is to assume a high level of confidence in the models by specifying and verifying properties.

So, the proposed meta model includes a property description language called LUSP (French acronym of Unified Specification Property Language [21]). This modelling language follows the model proposed in [22] very close to those proposed by [23] or [24]. A property is here considered as a causal, constrained and typed relation linking a set called causes and a set called effect. Each set is composed of events and data extracted from the model to be analysed. Each pre condition, post condition and constraint describing the relation between causes and effects is then

described by using predicates and functions respecting the meta model structure (i.e. each role of each relation between two concepts of the meta model is translated into a predicate or a function). An example of property modelling is shown in Fig. 1 in which for example the predicate named “neededAbilities” translates the relation role called “needed abilities” between an instance of concept named “Activity” and an instance of concept named “ResourceRole” as shown in Annex.

At last, the Properties Reference Repository (PRR) [22] gathers a set of generic properties described by using LUSP. It allows helping the modeller to select and to specify properties (that is to say to interpret generic properties) he wants to prove all along the model life cycle.

The properties are modelled taking into account the three following requirements:

- Users want to trust and to share the models contents.
- Users want to assume absence of identified but potential risks.
- Users want to make emerge new potential risks

4.1 Trusting The Model

In this first step of verification, it is a question of modelling properties making it possible to ensure itself of the coherence of each system component (scenarios, activities, processes, etc.):

- The models representing different viewpoints of the same system must share and must agree with the definition of the entities. A consensus is therefore needed between the different actors involved in a process. The properties allow to fix the ideas and to assume each viewpoint which describes the same behaviour of this process. If a consensus stays impossible to establish, then there are some potential dissonances between actors such as detailed in section 4.3.
- Models representing a higher detail level of another one. It is necessary to guarantee that refinement mechanisms respect some rules.
- Facing to the moving environment, to ensure that structural, functional and behavioural aspects of the system stay coherent considering all possible described scenarios.

4.2 Tracking Identified Risks

The concept of property is used during this step in order to model a risk clearly identified by using the model of risk suggested by the MADS-MOSAR approach [25].

Indeed this approach describes a risk as a causal relation between a potential cause called source state and its effect on the system called sink state. The source’s state describes what the states of each entity, data and events are like at a given time. The sink state describes what can be the resulting state of each entity in the system. These last entities may of course be different from those which are involved into the causes. Going from the source state to the sink state needs to describe occurring events (from different types such reinforcing event, initial event and so on). So the proposed property concept allows describing causes, effects and set of events which allow the system to go from a source state to a sink state. When a modeller encounters some risk which has occurred in the past he can use a property in order to describe it and to test if this risk can occur again.

The PRR is then used in order to gather some classical properties modelling identified risks in the medical domain. For example, delivering some kind of medicine such as drug to a patient needs to control the patient state every time. Does one scenario on which the case is described exist? Is there a procedure corresponding to this constraint?

A medical organisation has to face numerous identified risks and they have to respect some standards such as [26] but also internal procedures and so on. An analysis of these documents allows modeller and experts to add some properties to the PRR.

4.3 Making Emerge Possible Risks

A property is then used to model situations considered generally as potential generators of risks and dysfunctions in all kinds of systems. These situations and these risks are described by Cindynic [27]. However, these proposals remain more easily usable for analyzing known and passed cases and to formalize a kind of experience feedback. They are unfortunately regarded as not very usable a priori helping to detect emergent risks.

The goal is to model some of these propositions as property and to use formal verification tool presented below in order to make emerge some new but non identified risk.

Cindynic is based on a theoretical representation called "*hyperspace of the danger"* as shown in Fig. 2. This representation abstracts 5 information domains allowing to describe a system or a situation observed through the point of view of a group of actors. The members of this group share a common set of knowledge, of know how, position or job, role and responsibilities about the system. They have then a point of view about the system that is to say they own a particular and dedicated hyperspace of danger which may be very different from another group of actors.

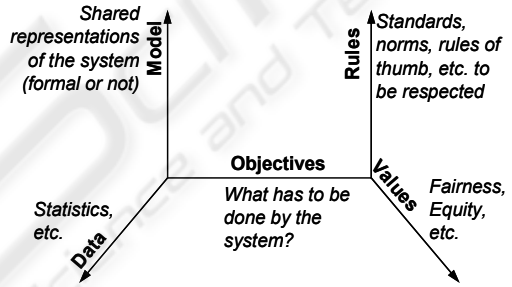

Fig. 2. Hyperspace of the danger.

Cindynic defines then two interesting concepts. First, the concept of dissonance is defined as the variation between each axis of two hyperspaces describing each one the same system seen by two groups of actors. It may be then possible to make emerge some possible conflict between actors (and then some risks which are not taken into consideration for the moment). It may also be possible to *detect* positive and negative *interactions* between some actors or a group of actors.

Second, the concept of deficiency (called Cindynogenic Structural Deficiencies DSC) is defined as a possible lack of knowledge on one or several of the 5 axes of

each hyperspace of danger. It can be for example a behaviour which is not completely defined, a scenario which do not take into account a possible incident, etc. This lack can be established by comparing the models issued from the system representation of each group of actors. Deficiencies were experimentally classified by [27], into three main categories (cultural deficiency, organizational deficiency and management deficiency).

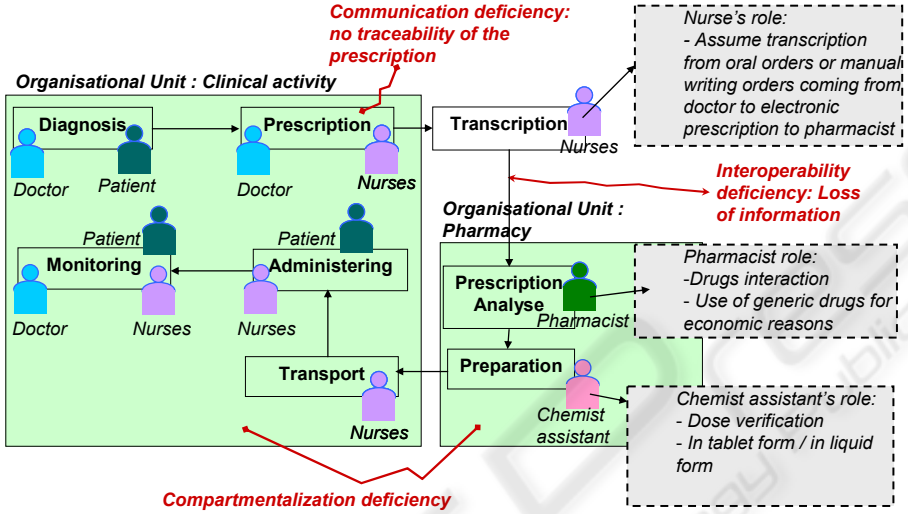

Fig. 3. Modelling example (simplified view).

Fig. 3 shows an example of system model in which three kinds of deficiencies are highlighted. The deficiency called 'Loss of information' brings a lack of interoperability between two organizational units. This lack exposes the organization and especially the patient to risk (risk of error in medicine, in dose, etc.). This identification of the deficiency can allow the modeller to decrease the system vulnerability by reformulating and improving the activity named 'Transcription' for example by the use of ICT tools.

5 Verification Mechanisms

On one hand, when modelled, a property P has to be proven (checked or verified) when a model M of a system S must satisfy P whatever may be the resources configuration, the available scenarios in which S may be considered and so on. On the other hand, knowing in which condition or for which scenario the property P cannot be verified allows detecting a modelling error or a mistake during modelling phase or even a real dysfunction and then a risk opportunity.

Even partial, the checking up of the properties resulting from each of the 3 requirements described above makes it possible to handle a knowledge each time more relevant, and especially more consensual between the actors. Models can be modified as correct in each iteration possible errors or gaps. They are enriched by a whole of properties which cannot be objected thereafter. However, by assumption,

any new modification of whole or part of the models requires to check again all the properties. That requires having tools making it possible to manage a great quantity of information and allows applying, if it is possible in an autonomous and not guided way. Theorem provers or model checkers are generally used at this stage [28]. However a formal operational semantic is then needed. E.g. the operational semantic of a state diagram can be formally described and then used by a model checker in order to prove behavioural properties. At the opposite, the proposed approach is based on several interacting models and paradigms which have to be merged with a common set of verification mechanisms. So, property proof is done by adapting and using a conceptual graphs [29] analysis approach as proposed by [30].

A conceptual graph is a formal knowledge representation. It is a finite, connected, directed and bipartite graph composed of an alternation of nodes called *concepts* and nodes called *relations.*

A **concept** is a double

[<type>: <marker>]

Where:

- *type* represents the occurrence of the object's class. They are grouped in a hierarchical structure called *concepts lattice*. The concepts lattice is obtained by translating each object class and each attribute described in the meta model by using translating rules summarised in Fig. 4.

- *marker* specifies the meaning of a concept by specifying an occurrence (i.e. an instance) of the type of concept. For example, the concept [Scenario: 'to deliver medicine'] describes an object of type scenario identified by 'to deliver medicine'. These markers correspond to the instances of each object (the marker description is provided by the name of the instance) contained into the system models. A **relation** binds two concepts according to the following diagram:

[Concept1]←(relation)←[Concept2]

For example, the following relation means that the object of type Configuration called 'C1' authorizes the object of type Scenario called 'S1':

[Scenario: 'S1']]←(Authorize)←[Configuration: 'C1']

As for the concepts, all the possible relations between concepts are gathered into a *relations lattice*. This relations lattice is obtained by translating each relation role between object of the meta model in a relation between concepts described in the concept lattice.

The verification approach (see Fig. 5) can be summarised as follows:

1 - Taking into account the concepts and relations lattices translated from the meta model, each model of the system is translated into conceptual graphs in which all the knowledge is then available and formalised by using an unique modelling language: the conceptual graphs. A set of formal translation rules (not presented here) has been developed aiming to transform the different system models on a (set of) conceptual graphs upon which formal reasoning mechanisms can be applied. Conceptual graphs may then be used to prove the previous presented properties.

2 – Analysis mechanisms allowed by conceptual graphs can be used. These analysis mechanisms are:

- **Projection**: This involves comparing the obtained conceptual graph coming from the translation of the model with another one translating the property. If the projection fails, then the modelled property cannot be verified and the causes are highlighted.

- **Constraint**: a property describes what the links and/or constraints are between facts. In this case, the property is translated on a positive or negative conceptual graph constraint. A positive constraint between two facts A and B must be interpreted as: “If A is true, then B must also be true”. Conversely, a negative constraint must be interpreted as: “If A is true then B must be false” (if B is true, A must be true or false).

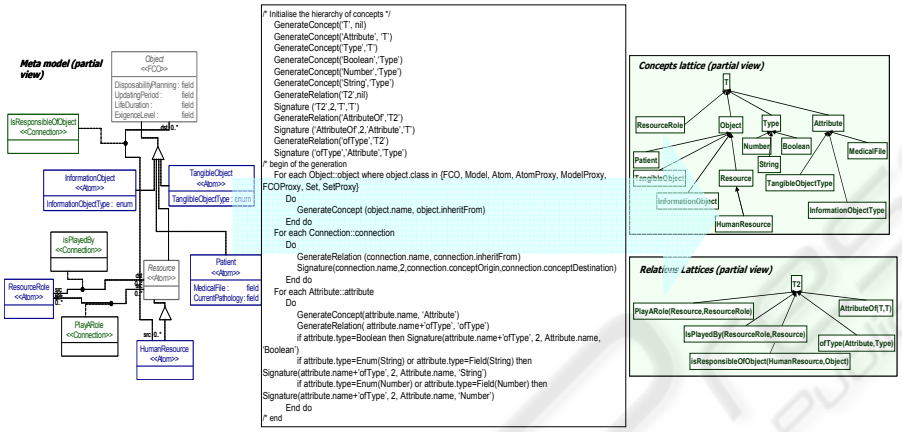

Fig. 4. From meta model to lattices (translating rules).

- **Dynamic and static rules**: A property is directly modelled as a rule composed of a cause and an effect as shown in Fig. 6. If the graph corresponding to the causes matches with a part of the conceptual graph translating the system models, then the effect must be checked in the same way.

For some specific modelling languages used in the modelling framework particularly the state diagrams used for describing state of each entities all along its life cycle, another verification approach can consist to translate the state diagram into input language of existing model checkers [28], [31]. This can allow to prove some behavioural properties such as state attainability or absence of deadlock and to highlight some model improvement or limitation.

6 Conclusion

The proposed methodology for verification is now under development. The modelling framework is still under validation by partners from Hospital of Nice in France. This approach can be a significant benefit for one in charge of risk management in a hospital. The modelling phase is currently used by some medical specialists. The analysis phase has now to define translation rules for obtaining dynamic rules and to define how projection must be automated by using Cogitant tool [32].

A possible development consists on using this approach in order to help the modeller not only to detect risk, but also to test different alternatives of organisation allowing to reduce the vulnerability of a healthcare organisation [33] and to improve the performance of this organisation.

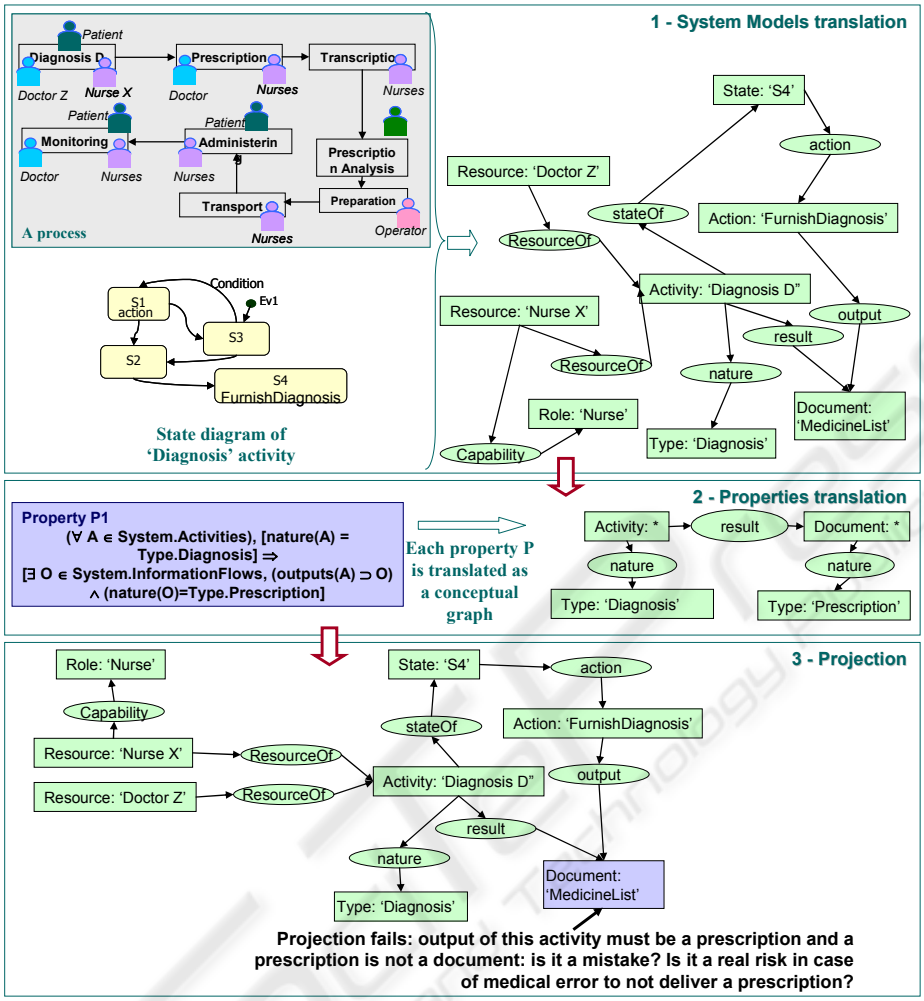

Fig. 5. Overview of the verification approach and mechanisms.

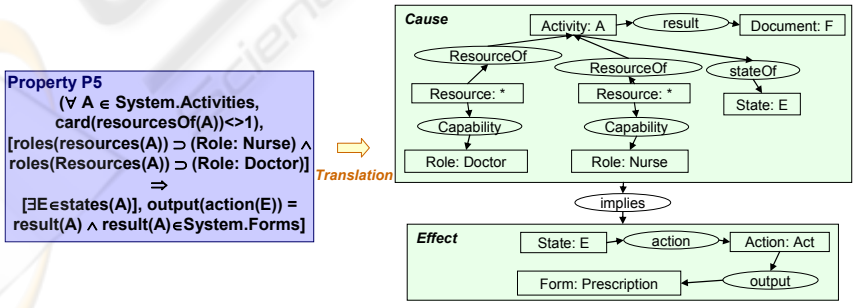

Fig. 6. Example of translation of a property specified by using LUSP in a dynamic rule.

References

1 Tixier J., Dusserre G. (2002) Review of 62 risk analysis methodologies of industrial plants, journal of Loss Prevention in the Process Industries

2 INCOSE (2004) System Engineering (SE) Handbook Working Group, System Engineering Handbook, A « How To » Guide For All Engineers

3 Vernadat F. B., (1996), Enterprise Modelling and Integration: Principles and Applications, Chapmann & Hall

4 Petit M., Doumeingts G. (2002) Enterprise Modelling State of the Art, Deliverable D1.1 of the UEML Project, Unified Enterprise Modelling Language UEML Thematic Network, IST-2001-34229 (www.ueml.org)

5 UEML (2003) Deliverable D3.1: Requirements analysis: initial core constructs and architecture, Unified Enterprise Modeling Language UEML Thematic Network - IST-2001-34229 (www.ueml.org)

6 Chapurlat V., Montmain J., Gharbit D. (2005a) A proposition for risks analysis in manufacturing and enterprise modelling, Knowledge sharing in the integrated enterprise: Interoperability strategies for the enterprise architect, Springer IFIP (P.Bernus & M.Fox ed., p 193-202

7 Penalva, J-M (1997). La modélisation par les sytèmes en situations complexes, PhD Thesis, Université de Paris Sud. (in french)

8 EICTA (2004) Interoperability white paper, European Industry Association for Information Systems, Communication Technologies, and Consumer Electronics

9 Uschold M., Gruninger M. (1996) Ontologies: Principles, Methods and Applications, Knowledge Engineering Review, vol.11:2, pp. 93-136

10 Popkin (2003) Enterprise modelling: Aligning Business and Information Technology, White paper, Popkin Software (see http://www.telelogic.com/campaigns/popkin/index.cfm)

11 GME (2004) Generic Modelling Environment (GME) User's Manual, Release 4-11, Institute for Software Integrated Systems (ISIS) Vanderbilt University

12 Aloui S., Chapurlat V., Penalva J.-M. (2006) Linking interoperability and Risk assessment: A methodological approach for Socio-technical systems, to be appear in Proceedings of INCOM'2006, 12th IFAC Symposium on Information Control Problems in Manufacturing, Saint-Etienne, May 17 to 19, France

13 Bertrand P., Darimont R., Delor E., Massonet P.,Van Lamsweerde (1998) A. GRAIL/KAOS: an environment for goal drivent requirements engineering Proceedings ICSE'98 - 20th International Conference on Software Engineering, IEEE-ACM, Kyoto, april

14 Van Lamsweerde A. (2003) From System Goals to Software Architecture. In Formal Methods for Software Architectures, M. Bernardo & P. Inverardi (eds), LNCS 2804, Springer-Verlag, 25-43

15 Menzel C.P., Mayer R.J. (1998) The IDEF Family of Languages in Handbook on architectures of information systems, Bernus P., Mertins K. et Schmidt G. ed., Berlin, Springer

16 Habrias H. (1988) Le modèle relationnel binaire. Méthode Niam, Paris, Eyrolles [in French]

17 Booch G., Rumbaugh J., Jacobson I. (1999) The Unified Modelling Language User Guide. Addison-Wesley

18 Oliver D.W., Kelliher T.P., Keegan J.G. Jr (2004) Engineering complex systems with Models and Objects, McGraw-Hill

19 Studer, R., Benjamins, V. Fensel, D. (1998). Knowledge Engineering: Principles and Methods, Data and Knowledge Engineering n°25, 161-197.

20 ISO 8402 (1994): Quality management and quality assurance – Vocabulary, Second edition 1994-04-01, International Standard Organization.

21 Lamine E. (2001) Définition d'un modèle de propriété et proposition d'un langage de spécification associé : LUSP, Ph.D. Thesis, Montpellier II University [in French]
22 Chapurlat V., Kamsu-Foguem B., Prunet F (2005b), A Formal Verification Framework and Associated Tools for Enterprise Modelling: Application to UEML, Computers in Industry, Elsevier
23 Chatel V., Feliot C. (2004) Principe de conception système certifiée par la preuve, Journées Francophones des Langages Applicatifs, JFLA 2004 (in French)
24 Accelera (2004), PSL Property Specification Language Reference Manual, Accelera Formal Verification Technical Committee (FVTC), Version 1.1 (http://www.eda.org/vfv/)
25 Perilhon, P. (2003). MOSAR: présentation de la méthode, Techniques de l'Ingénieur, traité Sécurité et gestion des risques (in french)
26 HAS (2005) Normative reference available on the HAS web site (Haute Autorité de Santé), see www.anaes.fr
27 Kervern G.Y. (1994) *Latest Advances in Cindynics*. Economica Paris.
28 Yahoda (2003) web site presenting an overview of formal verification tools (see http://anna.fi.muni.cz/yahoda/)
29 Sowa J.F (1984) Conceptual structures: information processing in mind and machine, New York (U.S.A.): Addison-Wesley
30 Kamsu-Foguem B. (2005) Modélisation et Vérification des propriétés de systèmes complexes: Application aux processus d'entreprise, July 2004, PhD Thesis University Montpellier II [in French]
31 Bérard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen Ph. McKenzie P. (2001) Systems and Software verification: model checking techniques and tools, Springer
32 Cogitant (2005) CoGITaNT Version-5.1 – Reference Manual (see http://cogitant.sourceforge.net)
33 ISDRD (2005), International Strategy for Disaster Reduction, http://www.unisdr.org/isdrindex.htm)

ANNEX: Partial meta model of the main modelling concepts and relations for multi view system modelling (GME 2004)