Towards Multi-Level Structuring of Goal-Oriented Models for Improved Model-Based Systems Engineering

Christophe Ponsard[1] and Robert Darimont[2]
[1]*CETIC Research Centre, Charleroi, Belgium*
[2]*Respect-IT SA, Louvain-la-Neuve, Belgium*

Keywords: Systems Engineering, MBSE, Goal-Oriented Modelling, Requirements, Refinements.

Abstract: Model-driven methods are gaining momentum in the industry to support the development of software intensive systems. While most methods rely on progressive levels of functional decomposition, the modelling of requirements tends to stay quite monolithic in nature. This position paper analyses how to better structure requirements across multiple levels in the scope of goal-oriented requirements engineering to reconcile it with layered system models and to provide better support for decomposition and analysis of inner sub-components or the outer system of systems. The proposed ideas are discussed based on a classical case study of the meeting scheduler with some early prototyping using a meta-case requirements engineering platform.

1 INTRODUCTION

Model-based systems engineering (MBSE) is defined as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE, 2007). Moving to MBSE is an important paradigm shift for the industry because the system model replaces documents for exchanging between engineers (Michot et al., 2018)(Madni and Purohit, 2019).

Many MBSE methodologies have developed over the past decade (OMG, 2018)(Weilkiens and Mao, 2022). Most rely on long-established proven Systems Engineering Methodologies (SEM). In order to cope with complexity, they support progressive system decomposition in different layers of abstraction. This decomposition is applied iteratively until single purpose components (or design entities) can be identified. Such design entities are structurally and functionally distinct from other elements (DOE, 2002). MBSE methodologies define standard layers with specific purposes. For example Arcadia, proposed by Thales and depicted in Figure 1, defines levels which first cover the problem domain with *operational analysis* (user perspective) and *system needs* (both functional and non-functional) and then the solution domain with the *logical architecture*, *physical architecture* and *end-product breakdown structure*.

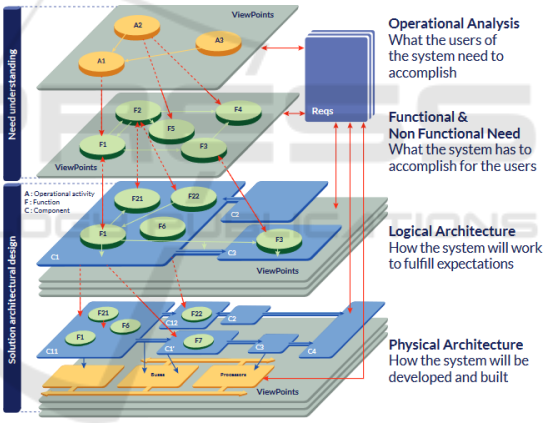

Figure 1: Arcadia MBSE Methodology.

In the same spirit, ASAP (Advanced System Architecture Program by Alstom) is organised in 3 layers: (1) the *operational* level focuses on users to identify their needs and the external interfaces (WHY), (2) the *functional* level defines WHAT the system does to fulfill its mission and identifies functional flows, and (3) the *constructional* level structures into interconnected subsystems and their behaviour (HOW).

The elicitation of requirements is a key activity of any system development. It is well known that many projects are still failing mainly due to poor Requirement Engineering (RE) practices (Hughes et al., 2015). Different methods can ensure key properties of completeness, accuracy, non-ambiguity and testability. They have also evolved from lightweight text-

Ponsard, C. and Darimont, R.
Towards Multi-Level Structuring of Goal-Oriented Models for Improved Model-Based Systems Engineering.
DOI: 10.5220/0011866200003402
In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2023)*, pages 269-274
ISBN: 978-989-758-633-0; ISSN: 2184-4348

based to model-based techniques. In the scope of this paper, we will consider Goal-Oriented Requirements Engineering (GORE) which aims at capturing, at different levels of abstraction, the various properties the system under consideration should achieve. More specifically we will use KAOS (van Lamsweerde, 2009) which was applied on large systems (Ponsard and Darimont, 2017). However other variants can also be considered, e.g. i* (Yu and Mylopoulos, 1997), GRL (ITU, 2012) or GSN (ACWG, 2018).

Although supporting abstraction and refinement, such methods do not define specific layers, leaving this task to the analyst which may rely on a SEM for this. A common flaw is the limited ability to reflect layers inside a GORE model. It usually relies on rather syntactic containers such as package and diagrams and lacks semantic structure providing the ability to define a clean interface between layers. Figure 2 shows the inner structure of a KAOS model composed of 225 goals described through 45 diagrams. The structure shows more than 10 refinement levels but such a view does not really make explicit connections with design entities as those are captured by another concept: the agent. Agent allocation in KAOS can only be specified at the leaves and lacks the ability to capture system-level agent. Other languages like i* allow this but without decomposition.

In the scope of this position paper, we are interested to investigate how to better structure GORE models by adopting a decomposition structure which is closer to the a typical SEM structure. This problem has not received much attention in the literature but is useful both in designing new system and for upgrading existing systems. For the former, the process can help identify a good structure and lay the foundation of a good architecture. For the latter, it helps in capturing the requirements at the right layer and identifying traceability impacts both to higher system level or down to specific components. We contribute here an extension to the KAOS meta-model with a more semantic decomposition of a model in interconnected sub-models. Our approach is experimented and discussed on the classical meeting scheduler case study (van Lamsweerde et al., 1995) which is considered both in a wider scope (enterprise business) and more focused scope (specific components).

This paper is structured as follows. First, Section 2 reminds about the existing KAOS meta-model and introduces our meeting scheduler case study. Section 3 explains our extension. Section 4 applies it to revisit our case study. Our current results are then discussed in Section 5. Finally, Section 6 draws some conclusions and identifies our future work.

2 KAOS META-MODEL AND CASE STUDY

Goal is the core concept of a GORE model. Goals prescribe, at different levels of abstraction, key properties the considered system should achieve. KAOS uses several abstraction levels to express goals starting from high-level strategic goals. In the meeting scheduler case study (van Lamsweerde et al., 1995) used as running example, it can be a goal like *Achieve[EfficientSchedulingOfMeetings]* as depicted in Figure 3 as light blue parallelograms. In KAOS, high-level goals are progressively refined into more concrete and operational ones through refinement relationships depicted as yellow circles. A refinement relationship links a parent goal to several subgoals with different fulfilment conditions using either *AND-refinement* (all subgoals need to be satisfied, e.g. the *EfficientSchedulingOfMeetings* requires to achieve four sub-goals:*TimeTableCollected, PotentialRoomsIdentified, MeetinSchedulingBasedOnAvailableData and MeetingInformationCommunicated*), or *OR-refinement / alternatives* (a single subgoal is enough, e.g. *Achieve[TimetablesCollected]* can be either *ManualCollection* or *AutomatedCollection*). The "WHY" and "HOW" questions can be used to conveniently navigate from subgoals to parent goals (why)

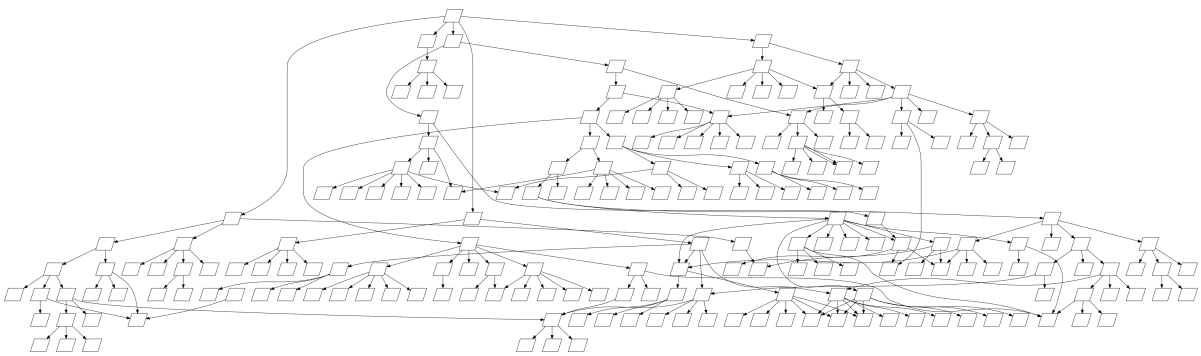

Figure 2: Typical inner structure of a KAOS model.

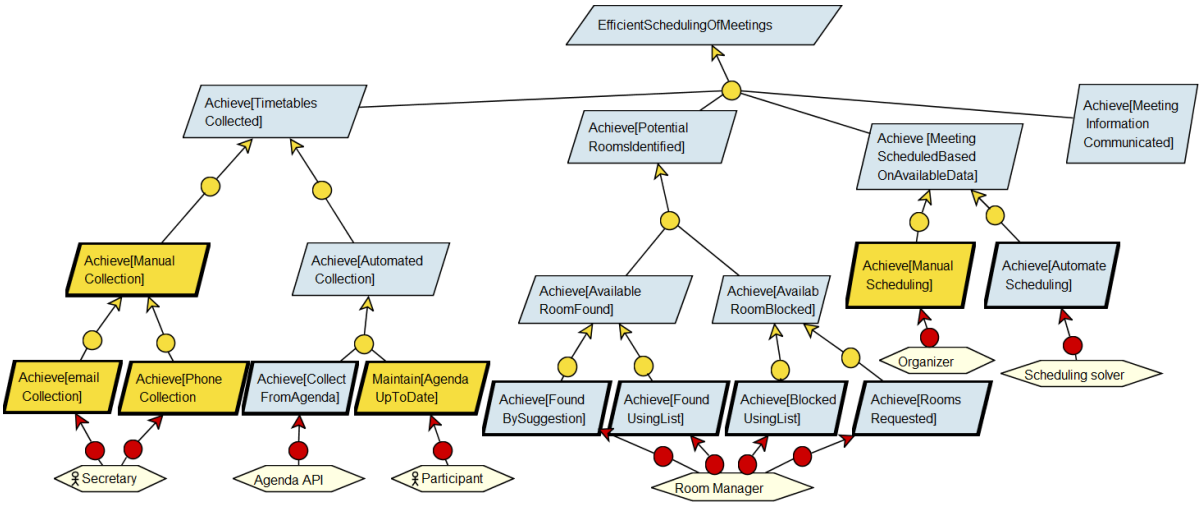

Figure 3: Monolithic goal graph for the meeting scheduler.

and from parent goals to subgoals (how). Goal refinement ends when every subgoal is realizable by some individual agent (depicted as hexagon) which will be assigned responsibility (through relationship with a red circle). Such a leaf goal is called a requirement if the agent is in the system under design. It is graphically differentiated from goals through a thick border. It is an expectation if it is under responsibility of an agent in the environment and then depicted in yellow.

Those concepts are summarised in the top part of Figure 4 depicting goal and agent parts of the KAOS meta-model. In addition, the meta-model also supports the description of the domain through an object model and the functionalities through an operation model. Those will be less central in this paper and will be discusses in Section 5. Note the goal model also supports the reasoning about obstacles to goal satisfaction, like safety or security risks.

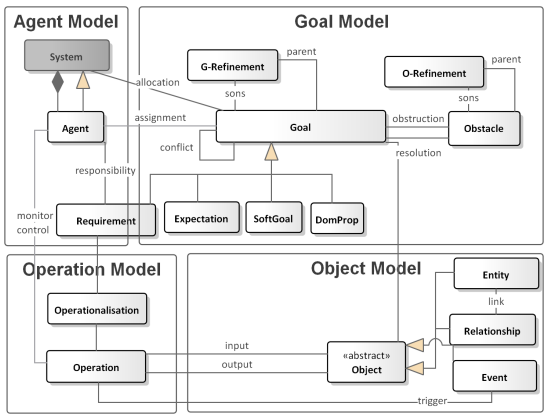

Figure 4: KAOS meta-model with proposed extension.

3 PROPOSED EXTENSION

The goal model presented in Figure 3 shows a complete refinement of the top-level goal into different requirements and expectations under the responsibility of 3 agents to be designed and 3 agents of the environment. Even small, this case study highlights limitations of the approach. A first observation is that part of the goal tree is under the control of a single agent (e.g. *Achieve[PotentialRoomIdentified]*) and one might wish to assign the responsibility higher in the tree which is not allowed. Another observation is that other parts require collaboration between agents but often with a single agent under design, while the others are environment agents, e.g. *Achieve[AutomatedCollection]*. Again here one might consider this agent as central while the others can be identified as interacting with it from the outer context. In both cases, the result is that a higher level goals might be considered as a requirement in a higher-level description of the system which still needs to be refined to be fully realizable. In the process, finer grained agents purely internal to an agent might also be identified, e.g. for the *RoomManager*.

In order to enable such system decomposition, we make explicit the notion of *System* in the meta-model. It is depicted in darker in Figure 4. It is introduced as a generalisation of *Agent* which will allow all agents to be considered as system, thus having a system dimension while being less constrained, e.g. only an agent can carry a *responsibility*. For a system, a more abstract relationship named *allocation* is introduced. A system is also an aggregation of finer-grained agents which is captured by the aggregation

relationship. This enables to capture hierarchical system breakdown structure and to ensure the alignment with the breakdown of a system model. Note also that the goal refinement must be consistent with this structure through some meta-constraint.

Our extension is supported by the KAOS Objectiver toolset (Respect-IT, 2005), a meta-case tool with multiple levels from a generic meta-meta level (M0) down to the instance level (M3). The meta-model (M1) is updated through a plugin inserting the extra meta-concepts and meta-relationships.

4 EXPERIMENT ON THE CASE STUDY

Applying our extension on the meeting scheduler yields the model depicted in Figure 5. It is much simpler to understand than the complete model of Figure 3. Of course such structuring becomes fully meaningful for large system as depicted in Figure 2.

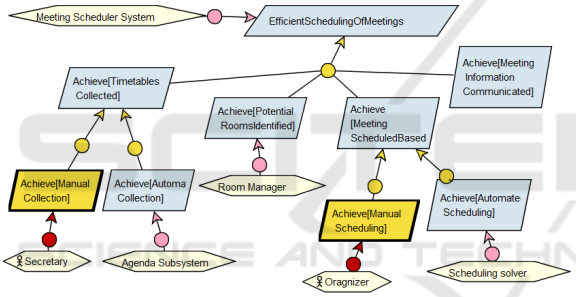

Figure 5: Meeting Scheduler with system refinement.

Of course, it does not include all the information as some agents now have to be refined through subsystems. For example, the subsystem to manage the available rooms is depicted in Figure 6. Note also the top-level goal has been tagged with an explicit agent representing the corresponding system level.

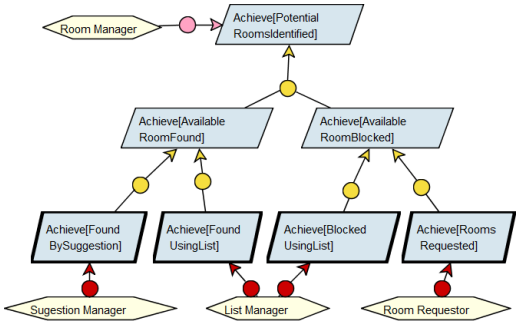

Figure 6: Room management subsystem.

Based on this, it is now easy to extract the agent breakdown structure of Figure 7.

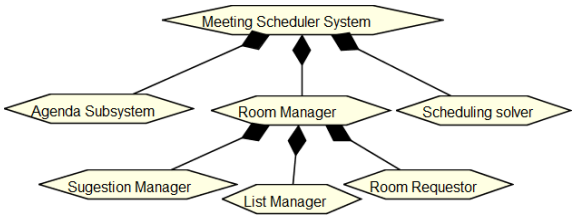

Figure 7: System breakdown structure.

Considering our meeting scheduler system, the enforced goal contributes to the fulfilment of goals of a larger enterprise system, e.g. for project management or internal collaboration, as depicted in Figure 8.

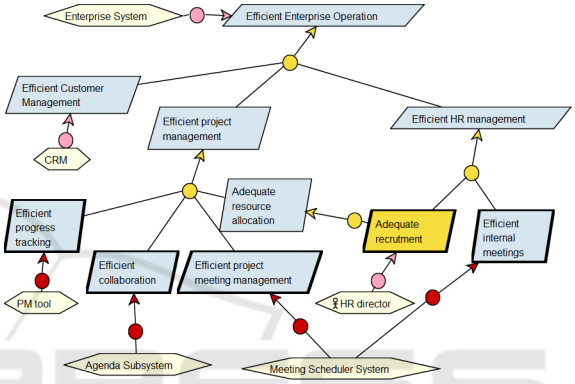

Figure 8: Organisation level system.

Analysing this system reveals an interesting point: the agenda agent which was identified as subsystem of the meeting scheduler seems more general and also used to support collaboration and not only for scheduling meetings which looks more a secondary usage. Analysing this reveals the agenda is not a subcomponent but rather an external component which is collaborating with the meeting scheduler through a specific interface in order to collect availabilities. Extracting the breakdown structure again gives a more global picture of the system as shown in Figure 9.

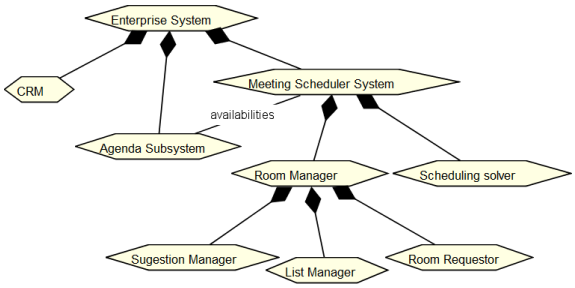

Figure 9: Enterprise level breakdown structure.

5 DISCUSSION ABOUT RELATED WORK

As stated earlier, many GORE frameworks have limited support for system decomposition. However, researchers have investigated this problem. First Jackson problem frames (Jackson, 2001) is based on a process of parallel - not hierarchical - decomposition of user requirements and result in the analysis of many subproblems that are then combined to form a solution design. To better support this essentially semi-formal approach, a meta-model was proposed to capture the notion of sub-problems but also sub-domains and to represent problems in intermediate situations during the (de)composition phases (Lavazza et al., 2010). Although it eases the analysis, the process does not result in a breakdown structure that can be applied to MBSE as in our proposal.

Another approach is DeSyRe which systematically decomposes system requirements into subsystem requirements according to a given system architecture and with focus on informal or semi-formal requirements (Penzenstadler, 2011). It provides guidance to derive subsystem requirements from system requirements by use of assumption/guarantee reasoning and decomposition patterns. This can complement standard goal-decomposition patterns (Darimont and van Lamsweerde, 1996) in order to define subcomponent requirements.

More recent work has focused on more formal approaches for industrial system (Teufl et al., 2014). It has shown the benefits of using a two-step process for refining system requirements to subsystem requirements: interface refinement on the system boundaries, and a decomposition of system requirements to subsystem requirements. This can be complemented with formal analysis and verification techniques on the refinement of requirements. So far, our work has considered more the decomposition step but the need to reason about interfaces was identified in our structuring approach. For this purpose, the KAOS agent model can capture monitor and control flows across agents and can enable this kind of reasoning.

Globally our approach is still semi-formal and relies on a rich meta-model with a focus on properties through goals and agents which can capture components under design but also existing parts or human agents interacting with the system. The object and operation model are less relevant as they are better managed by MBSE using notations like SysML (OMG, 2005). For example, a class diagrams can describe domain models while an internal block diagrams can capture system functions. Such models can then be synchronised with the GORE model for enriching it if necessary. In order to be consistent, the meta-model also need to enforce a few meta-constraints related to our extension. So far, we could identify the following ones:

- Root goals in an agent subsystem decomposition must match all requirements allocated to corresponding agent in parent system.
- An agent part of a subsystem must be responsible of requirements in that subystem.
- A system is only composed of agents allocated to requirements resulting from alternatives specific to that system and not refining external functional goals.

Our meeting scheduler case study contains OR-refinement but we did not provided any specific treatment for them. They can result in alternative agents and produce a richer system breakdown structure that can evolve towards configuration diagrams. The coupling between goal refinements and agent structure can provide a good bridge between those two different ways of capturing variability. However this requires more work. Along the same line, we have also limited to functional requirements and not considered possible conflicts among them.

6 CONCLUSION AND PERSPECTIVES

This paper explored how to provide better support for decomposing GORE models across multiple levels of abstraction as an MBSE context with a focus on goals and agents structuring. We proposed a meta-model extension which enables to really capture systems and subsystems as part of a model and beyond the mere use of purely syntactic container such as packages and diagrams. We also identified some meta-constraints that ensure model consistency. We illustrated on a meeting scheduler case study ranging on 3 levels of system refinement. Finally, we discussed our current result in the light of some related work to highlight the challenges ahead.

Our future work is to progress with the validation on more complex case studies with industrial partners, in the context of the modernisation of a complex railway system. At the tool level, we plan to improve the management of layers through the ability to navigate across layer and also focus on a level by temporary hiding the other ones until the analyst wishes to switch again to a more global view. We also plan to enrich the analysis capabilities for dealing with refinement and to deploy a model synchronisation bridge.

This will enable the refinement of subsystem requirements both from the GORE and MBSE environments and, from there, to assess what are the best scenarios and practices.

ACKNOWLEDGEMENTS

This work was partly supported by the MORSE project (nr 8389) under FEDER funding for Wallonia. Thanks to our industrial partners for their inputs in GORE and MBSE for exploring system decomposition in both fields.

REFERENCES

ACWG (2018). Goal Structuring Notation Community Standard, Version 2. The Assurance Case Working Group https://scsc.uk/r141B:1?t=1.

Darimont, R. and van Lamsweerde, A. (1996). Formal refinement patterns for goal-driven requirements elaboration. In *Proc. of the Fourth ACM SIGSOFT Symposium on Foundations of Software Engineering, SIGSOFT1996, San Francisco, California, USA, October 16-18*. ACM.

DOE (2002). Systems Engineering Methodology, Version 3. US Department of Energy DOE G 200.1-1A.

Hughes, D. et al. (2015). *Success and Failure of IS/IT Projects: A State of the Art Analysis and Future Directions*. SpringerBriefs in Information Systems. Springer International Publishing.

INCOSE (2007). SYSTEMS ENGINEERING VISION 2020. http://www.ccose.org/media/upload/SEVision2020_20071003_v2_03.pdf.

ITU (2012). Recommendation Z.151 (10/12), User Requirements Notation - Language Def. https://www.itu.int/rec/T-REC-Z.151.

Jackson, M. (2001). *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc.

Lavazza, L., Coen-Porisini, A., Colombo, P., and del Bianco, V. (2010). A meta-model supporting the decomposition of problem descriptions. pages 50–57.

Madni, A. M. and Purohit, S. (2019). Economic analysis of model-based systems engineering. *Systems*, 7(1).

Michot, A., Ponsard, C., and Boucher, Q. (2018). Towards better document to model synchronisation: Experimentations with a proposed architecture. In *Proc. of the 6th Int. Conf. on Model-Driven Engineering and Software Development, MODELSWARD 2018, Funchal, Madeira - Portugal, January 22-24*.

OMG (2005). System modeling language. http://www.omg.org/spec/SysML.

OMG (2018). MBSE Methodologies and Metrics.

Penzenstadler, B. (2011). Exactly the information your subcontractor needs: Desyre - decomposing system requirements.

Ponsard, C. and Darimont, R. (2017). Improving requirements engineering through goal-oriented models and tools: Feedback from a large industrial deployment. In *Proc. of the 12th Int. Conf. on Software Technologies, ICSOFT 2017, Madrid, Spain, July 24-26*.

Respect-IT (2005). The Objectiver Requirements Engineering Tool. http://www.respect-it.com.

Teufl, S., Böhm, W., and Pinger, R. (2014). Understanding and closing the gap between requirements on system and subsystem level. In *IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE)*.

van Lamsweerde, A. (2009). *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley.

van Lamsweerde, A., Darimont, R., and Massonet, P. (1995). Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. In *Proc. of IEEE Int. Symposium on Req. Eng. (RE'95)*.

Weilkiens, T. and Mao, M. D. (2022). MBSE Methodologies. https://mbse-methodologies.org.

Yu, E. and Mylopoulos, J. (1997). Enterprise modelling for business redesign: The i* framework. *SIGGROUP Bull.*, 18(1):59–63.