Supporting ad hoc Analyses on Enterprise Models

Sabine Buckl[1], Jens Gulden[2], Christian M. Schweda[1]

[1] Chair for Software Engineering for Business Information Systems (sebis)
Institute for Informatics
Technische Universität München
{sabine.buckl,schweda}@in.tum.de

[2] Chair for Information System and Enterprise Modelling
Institute for Computer Science and Business Information Systems (ICB)
University of Duisburg-Essen
jens.gulden@uni-duisburg-essen.de

Abstract: Enterprises are socio technical systems whose management involves multiple stakeholders each demanding for a distinct perspective on the enterprise. A large number of modeling languages for describing different viewpoints on an enterprise have been developed in practice and academia. These viewpoints typically reflect the areas of interest that certain stakeholders pertain in respect to the enterprise, e.g. process modeling, IT resource modeling. While these viewpoints support analysis confined to their area of interest, crosscutting analysis, e.g. of the IT support for processes, demand for ad hoc viewpoints spanning different areas of interest.

In this paper, we present a building-block based approach to enable stakeholders to create ad hoc viewpoints on enterprise models, which are complementingly operationalized via a model-transformation based method for generating visualizations.

1 Motivation

Business enterprises are organizations with a high degree of structural complexity and dynamics in their behavior. They represent socio-technical systems, which are subject to various external influences. A socio-technical system consists of human actors and technical constituents. Thereby, the technical constituents, e.g. information systems, build the infrastructure on which the collaborative actions are performed by humans. The human actors thereby typically do not only pursue the organization's goal but additionally have individual goals and responsibilities, which they try to accomplish. Due to a multitude of dependencies among human actors and information systems, the qualitative complexity of an organization increases exponentially in relation to its quantitative scale. This means, while an organization develops and matures, it continuously becomes more difficult to oversee the relation between its intended goals on the one hand, and the actual implementation of operations that are performed to achieve these goals on the other hand.

To align the structure and behavior of a continuously maturing organization with its strategic goals, cognitive support is required to gain insight into factual as-is constellations as

well as to express possible to-be situations. Due to the high degree of interdependency and meshed complexity, such means cannot be provided by generic instruments of communication, e.g., by using linear natural language. Instead, a method to cope with these tasks is required as an instrument to provide the required semantic expressiveness for knowledge explication. Such support is available through the use of enterprise modeling or enterprise architecture (EA) modeling methods [Fra94b, Sch02, FS06, BEL$^+$07, KW07].

Visual diagram languages for enterprise modeling provide elaborated techniques to capture knowledge about organizations. They support involved stakeholders in creating models of an organization and in explicating this knowledge in a way that can further be communicated and discussed among different groups of stakeholders. Given appropriate abstractions and suitable notation elements for the real-world concepts, enterprise modeling languages serve as a common basis of understanding among the stakeholders, since they avoid ambiguities and semantic overloading of terms, which are typical reasons for misunderstandings and inefficient communication in organizational settings.

To achieve an appropriate level of abstraction and understandability, enterprise modeling languages are designed as domain specific languages which provide designated language concepts that facilitate communication about strategic goals, organizational structure and operational behavior of organizations. Advanced enterprise modeling methods use interrelated multiple perspectives [Fra94a] by incorporating multiple diagram types, which are internally related on the level of language design to allow sharing of identical concepts in multiple perspectives. This is vital to ensure semantic integrity among multiple perspectives, since referenced concepts from other perspectives are ensured to be further explicated in their own designated perspective.

Creating enterprise models with domain specific languages fosters the separation of concerns between on the one hand incorporating general principles of the modeled domain on the language level, and on the other hand creating model instances that accurately describe a subject's perceived reality about real world constellations of concrete enterprises. The tasks of creating and editing model instances can be best performed by the stakeholders who are themselves involved in the organization which is described by enterprise models. The upstream task of language design, however, is a genuine academic challenge to be carried out carefully with support of scientific research. This separation of concerns makes the use of enterprise modeling methods efficient and attractive for practical use. Modelers can rely on previously elaborated domain specific languages, so the responsibility for ensuring semantic integrity and understandability among different groups of stakeholders is shifted to the process of language creation, making the use of individual models more efficient and less prone to errors.When models of enterprises and organizations are to applied with methodical support in a way understandable for all stakeholders, the use of elaborated domain specific enterprise modeling languages thus is a first choice approach.

The range of possible uses of enterprise models is broad, once a coherent set of models from interrelated perspectives is available and maintained using domain specific languages. Besides serving as means for communication, enterprise models can be utilized to develop information systems which supply the described organization's tasks, e. g., by deriving executable workflow descriptions from business process models. They can furthermore be used reflectively as tools to access information about operative systems and

organizational entities represented in the models [FS09]. When applied in such a manner, enterprise models are no longer used for capturing knowledge from different perspectives to make it commonly accessible for diverse stakeholders, but they now serve as a repository of knowledge from which different stakeholders with their individual concerns can extract modeled facts and relate operational information to them.

Extracting knowledge from enterprise models is a task orthogonal to the situation when capturing knowledge by creating and editing them. When extracting knowledge, it is no longer the aim to find a commonly understandable way of representing facts in a semantically integrated way, but to grasp specialized information for specific concerns of stakeholders from existing models. To distinguish this mode of operation from creating and editing enterprise models, we call such operations of utilizing existing models for knowledge extraction *analyzing* of enterprise models.

To support analysis of enterprise models, different methodological means than for creating and modifying enterprise models are required. It is now the aim to allow specific extraction of knowledge, resulting in information that may only be understood and specifically be used by individual groups of stakeholders. Each group of stakeholders has individual concerns and specific tasks to fulfill in the organization. These concerns determine specific reasons and motivations for performing model analyses, which are naturally diverse and heterogeneous among different groups of stakeholders.

A stakeholder who performs analyses to meet specific concerns demands a number of requirements to gain suitable results. These requirements include the ability

- to *access* any of the facts incorporated in various perspectives of an enterprise model

- to *select* desired facts specific to the concern in focus, and *combine* interrelated facts into new relationships which are not determined in advance by a single applied enterprise modeling languages

- to *present* facts in a notation suitable for the specific concern of the stakeholder, but not necessarily understandable for other groups of enterprise model users

- to *perform* these tasks in an "ad hoc"-way, i.e., be quickly and efficiently able to specify the desired steps of accessing, selecting, combining and presenting per analysis scenario in an easy to handle and standardized manner

In response to these requirements we present methodical support for analyzing enterprise models in an ad hoc manner. Preparing the underlying technique, Section 2 lays the terminological foundations reflecting our understanding of *concerns* and *viewpoints*. Section 3 describes the basics of *multi-perspective enterprise modeling* (MEMO) and of *systemcartography* (SyCa), two research fields that have contributed to our methodical approach for performing ad hoc analyses. Section 4 introduces an example from the enterprise modeling domain, which is used to illustrate how the building-blocks of the technique can be used to define ad hoc visualizations. Concluding Section 5 summarizes the paper's findings and gives an outlook on further research topics in the field of using ad hoc viewpoints for performing visual analyses.

2 Terminology

The motivating section already introduced manifold termini, e.g. *concern* or *modeling language*, that shape the field of action in which the method for performing ad hoc analyses acts. Preparing the more formal understanding of these termini throughout the method, we elaborate on the relationships between the corresponding concepts. Thereby, we resort to the definitions provided by the ISO Std. 42010 (cf. [Int07]). This standard reflects that people (*stakeholders*) may have distinct interests in the structure of any system. A *concern* describes the corresponding area-of-interest, i.e. may in a colloquial way be understood as identificator of a part of the overall system. Refraining the more formal understanding as put forward by Buckl et al. in [BKS10], a concern has a twofold nature representing:

- a *conceptualization* of the system, i.e. a mental model that allows to classify real world entities and relationships to classes and associations, as well as

- a *filter* on the system, restricting the entities and relationships to those of interest.

The conceptualization is therein used to provide a metaization for the real world, whereas the filter may be regarded as "instance-level" concept used to restrict the concern to a subset of possible instances. Exemplifying the interplay of both constituents of a concern, we provide an exemplary concern textually described as

sequence of business processes as executed at the enterprise's headquarter.

The conceptualization of this concern brings along two classifying concepts, one for entities that are called "business process" and one for relationships that are understood as "sequence". Complementing, the filter expresses that only those processes executed at the headquarter are of interest. The example shows the twofold nature of the concern and gives an indication on which basis different concerns can be seen as related. Refraining an argumentation put forward by Buckl et al. in [BMS10] two concerns may relate in respect

- to their conceptualization, i.e. may classify real world entities similarly, or

- to their filter, i.e. may restrict to similar or disjunct parts of the enterprise.

Committing further to the terminology of the ISO Std. 42010 (cf. [Int07]), we revisit the notion of the *viewpoint* as the a "set of conventions for the construction and interpretation of a view, i.e. a [representation of a system from the perspective of a set of concerns]". In the context of describing a complex system, different viewpoints are selected and used by the stakeholders that raised the corresponding concerns. A viewpoint presents a twofold nature incorporating a *modeling language* consisting of *syntax, semantic*, *notation*, and a filter (cf. Buckl et al. in [BKS10]. Inline with this understanding we may say that each modeling language as used for enterprise modeling *commits* to at least one conceptualization as incorporated in an according concern. This in turn means that a specific modeling language summarizes the interests in the architecture of the enterprise as raised by different stakeholders. Figure 1 graphically illustrates the concepts as introduced above as well as their interrelations.
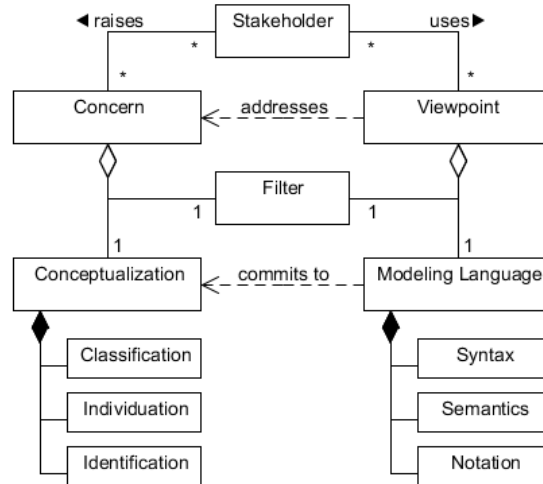
Figure 1: Relationships between stakeholders, concerns and viewpoints

3 Foundations and Prefabrics

As an example procedure for creating and modifying enterprise models, we have chosen the "Multi-perspective Enterprise Modeling" method MEMO [Fra94a], which defines a set of domain specific languages that form multiple interrelated views on an organization and its information systems infrastructure. It satisfies the theoretic demands stated above for providing an elaborated set of languages that are semantically integrated. MEMO also offers views and a domain specific graphical notation suitable to be understood by multiple different groups of stakeholders. Tooling support for the MEMO method is available via the software MEMOCENTERNG [GF10], which provides domain specific diagram languages with corresponding diagram editors.
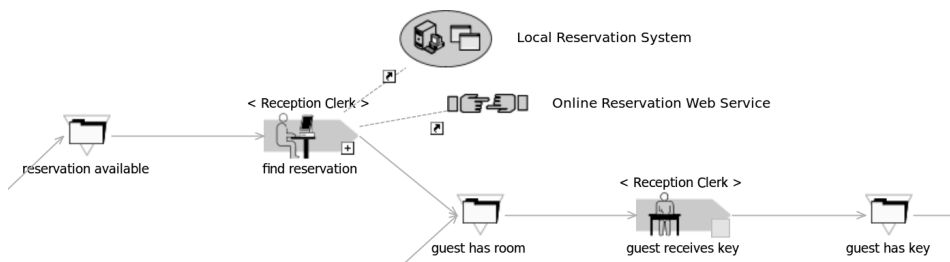

Figure 2: Excerpt from a MEMO process control flow model referencing elements from other perspectives

The MEMO method offers modeling perspectives for strategic goals and high-level actions, as well as for modeling organizational roles, resource entities and processes. These

multiple perspectives are centrally integrated through the use of process control flow models. In process control flow models, process steps get associated with responsible actors whose roles are defined in organization diagrams. Resources are modeled with a resource modeling language and can be allocated to process steps, to express which resources are accessed in that specific process step. The semantic integrity of these multiple perspectives on an organization is internally ensured by the language architecture, which is implemented in the tool support offered by MEMOCENTERNG. Figure 2 shows an excerpt from a MEMO process control flow model edited in MEMOCENTERNG, in which organizational roles and resources from other perspectives are referenced.

For the example implementation of our approach we use model data edited with MEMO-CENTERNG and transform it to an intermediate format suitable for further import into the System Cartography (SYCA) tool [BEL$^+$07]. This procedure is prototypically applied to model data from MEMOCENTERNG, however, the approach we present is independent from the concrete modeling method and is applicable to other enterprise modeling methods, too, provided a transformation to the intermediate import format is available. Once an enterprise modeling language is implemented using the same meta-modeling environment as SyCa, which is the Eclipse Modeling Framework (EMF, [SBPM09]), the effort for integrating it with our approach remains low and boils down to developing a single model-to-model transformation with the enterprise modeling language as input, and SyCa model elements as output.

We pursue an approach for enterprise modeling based on model transformation to ensure the consistency between information, e.g. data in an enterprise architecture (EA) repository, and visualizations of the EA. Therefore, a strict separation of the content to be visualized – the semantic model – and its representation – the symbolic model – is required. Additionally, a well-defined link between these models – the syca transformation – is needed. Figure 3 shows the basic idea of the model transformation approach.
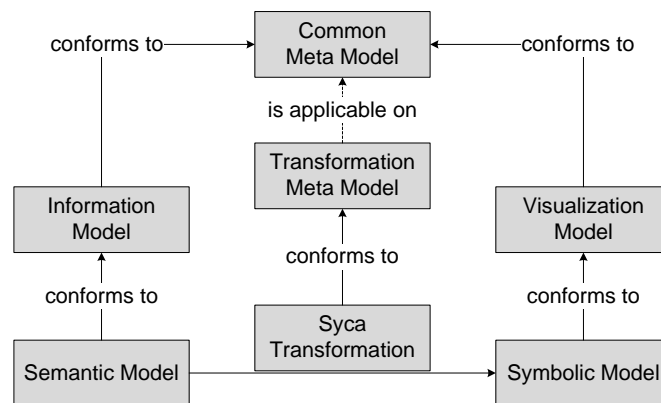

Figure 3: Generating visualizations based on model transformations

The information model sets up the language for describing the modeling subject, i.e. it introduces the core concepts, which are used to create a model of the subject's reality.
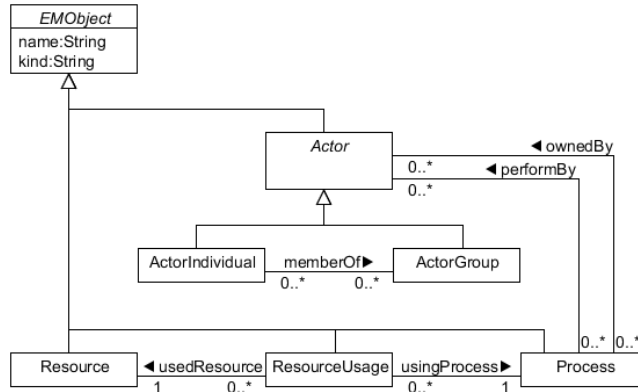
Figure 4: Information model integrating multiple model perspectives

In the context of EA management, the information model (for an example see Figure 4) contains concepts like processes, actors, etc., which are represented irrespective a visualization. Instance data documented in accordance to the information model is part of the semantic model, which contains so called *information objects*. In this sense, the information model acts a meta-model for the semantic model. Via the integrated information model the different facts, i.e. concepts, properties thereof and relationships inbetween, are made accessible for a stakeholder that seeks to define a corresponding viewpoint.

3.1 Symbolic model and visualization model – the *right* side

The visualization model contains elements, which represent graphical concepts, namely *map symbols*, e.g. "rectangle", or *visualization rules*, such as "nesting". These rules do not represent visible concepts, but they exert distinct demands on the positioning, size, or overall appearance of the symbol instances. For example, instances of the "nesting" rule demand that "inner" map symbol instances are grouped into the "outer" map symbol. Figure 5 introduces the map symbol and visualization rule that are needed for the exemplary visualization. Figure 6 displays the symbolic model describing that rectangles representing business applications are nested in the rectangle representing the hosting location.

The syca transformation creates a symbolic model based on the corresponding semantic model, while the map symbol instances in the symbolic model are not yet supplied with absolute positions. These positions are in a second step calculated by a layouter, which is capable to interpret the visualization rule instances and to compute appropriate positioning and sizing. Sketching the mathematical formalism incorporated in the layouter, we give examples of the layouting constraints that apply on the rectangle instance[1].

---

[1] According to the visualization model of Ernst et al. [ELSW06], the symbols' x and y-coordinates are anchored at the symbols' centers.
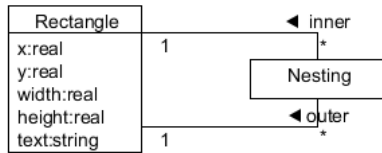
Figure 5: Visualization model

Figure 6: Cutout of the symbolic model

```
check.x − check.width/2 < rClerk.x − rClerk.width/2
check.x + check.width/2 > rClerk.x + rClerk.width/2
check.y − check.height/2 < rClerk.y − rClerk.height/2
check.y + check.height/2 > rClerk.y + rClerk.height/2
```
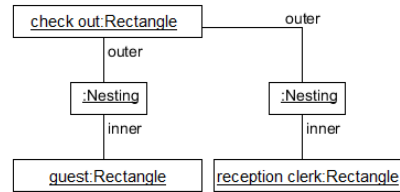
3.2 The syca transformation and its meta model – the *middle*

The syca transformation establishes the link between the data and its visualization and thereby enable the automatic generation of corresponding architectural views. Different types of model-to-model transformation languages may be used to implement the syca transformation. The transformer component of the tool interprets the transformation rules and generates a symbolic model from the corresponding semantic model. Over the last years, we have successfully applied different model-to-model transformation languages for defining architectural viewpoints in an executable manner. Wiegelmann showed in [Wie08] that the Atlas Transformation Language (ATL) [ATL06] can be used to describe the necessary transformations. With ATL, architectural viewpoints can be defined in a highly declarative manner, although especially the appropriate instantiation of visualization rules becomes fairly complex, e.g. when matrix-like views should be created. An exemplary ATL-like code for generating a visualization is given below.

```
rule Process2Rectangle {
  from infoObject : Semantic.Process
  to symbol : Symbolic.Rectangle (text = infoObject.name)
)

rule Actor2Rectangle {
  from infoObject : Semantic.Actor
  to
    symbol : Symbolic.Rectangle (text = infoObject.name),
    rule : Symbolic.Nesting (
      inner = symbol,
      outer = transforming (infoObject.performedBy)
    )
)
```

4 Method and Technique for Defining ad-hoc Visualizations

Building on the foundations described in the preceeding section, especially on the understanding of visualizations as the results of applying a model transformation to an underlying conceptual model of an enterprise, this section establishes a method and a respective technique for defining arbitrary visualizations on an enterprise model.

Central to the method is the notion of the viewpoint building-block, i.e. of a re-usable component for defining a viewpoint. Each building-block of that kind describes a distinct form of visualization (or part thereof), whose operational semantics can be specified in terms of an according model transformation. In respect to the type of contribution that a building-block makes in specifying a visualization, three different types of building-blocks can be distinguished as follows:

Structural building-blocks which describe the basic structure of a visualization,

Symbol building-blocks which describe (complex) symbols used in a visualization, and

Decorator building-blocks which describe graphical annotations that apply to symbols.

With each of the above building-block types prescribing distinct visualization forms or parts thereof, only a configuration employing building-blocks of different types can define a distinct viewpoint. Furthermore, the building-blocks are formulated in a generic way, i.e. do not assume a specific underlying meta-model. In order to combine viewpoint building-blocks into an executable viewpoint definition, relationships to meta-model concepts have to be added.

Before we complement above abstract considerations with a concrete example of building-blocks, we revisit the model transformation approach once more from an abstract perspective. Put in a mathematical sense a viewpoint described via a model transformation can be understood as a function mapping one[2] enterprise model element (an IOBJECT) to one visualization model element (a VOBJECT). This kind of transformation may either be atomic and self-contained, i.e. be executable without relying on other transformations (*symbol building-block*), or may be parameterized with one or more sub-transformations (*structural building-block* or *decorator building-block*). In terms of our formal understanding, building-blocks of the latter two types may be regarded as *functional operator*, i.e. some sort of function that takes another function as input. While such mathematical considerations may be very beneficially for defining consistency rules on the possible combinations on viewpoint building-blocks, we abstain from detailing such rules here. Instead we retreat to an intuitive understanding of consistency based on the analogy of *pipe-and-filter* [BMR+96]. In this sense every building-block is a filter with a distinct signature, expecting input and output of certain type as well as appropriate parameters. Pipe-and-filter models are used for manifold purposes with "yahoo pipes"[3] being a novel

---

[2]One might argue that more than one enterprise model element may serve as input of such model transformation. Nevertheless, the restriction to a single element is no restriction at all, as – possibly synthetic – *container* elements may be used.

[3]For more information on yahoo pipes, see `http://pipes.yahoo.com/pipes/`.

and succesful application of this paradigm of thinking to the field of web-based mashup design. In a similar sense the paradigm is applied in the following to the field of viewpoint definition using building-blocks. Thereby we rely on a graphical way to describe such building-blocks as well as their composition in an intuive way. Table 1 exemplarily introduces the graphical primitives of this description technique.
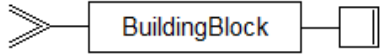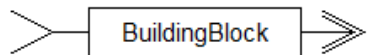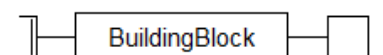
	A building-block taking multiple IObjects as input and producing multiple VObjects
	A building-block taking a single IObject as input and producing multiple IObjects
	A building-block taking multiple VObjects as input and producing a single VObject

Table 1: Examples for the graphical way of describing viewpoint building-blocks

Above notation uses rectangles to depict viewpoint building-blocks and arrows with different types of heads to distinguish between input (inward symbol) and output (outward symbol) and between IObject (triangular symbol) and VObject (rectangular symbol). Further the corresponding input or output indicator is augmented with line doubling, if not a single object of the according type is concerned but multiple ones.
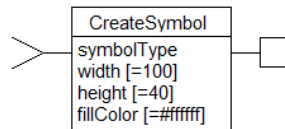

Figure 7: Building-block CREATESYMBOL

Using above notation, we exemplify a symbol building-block named CREATESYMBOL (cf. Figure 7). This building-block describes a transformation that creates a single planar symbol (VObject) for an according IObject. Planar symbols in this respect are e.g. rectangles, circles, chevrons. In a more formal sense, any geometric shape with a *Hausdorff dimension* of 2 may be regarded as "planar". For in-depth considerations on this topic as well as for a list of widely used planar symbols refer to Ernst et al. [ELSW06]. CREATESYMBOL has multiple parameters[4] as SYMBOLTYPE, WIDTH, HEIGHT, and FILLCOLOR, of which the latter three are supplied default values. In order to use this building-block every parameter without a default value must be supplied a valid assignment, i.e. the parameter SYMBOLTYPE must be set to a class representing a planar map symbol.

In Figure 8 we exemplify a structural building named CLUSTER. A cluster describes a graphical relationship between a symbol representing a single enterprise model element and nested symbols that represent related enterprise model elements. Thereby, the build-

---

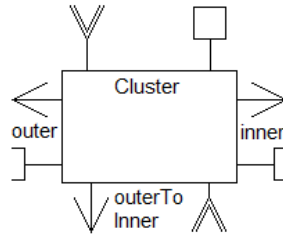[4]Some more parameters exist, but are omitted here for reasons of brevity.

Figure 8: Building-block CLUSTER

ing block does not make prescriptions on the creation of the "outer" or "inner" symbols and does also not impose constraints on the type of relationship. Put in other words, a CLUSTER represents a "functional" building-block not only taking a multiple IOBJECTs as input and creating a single VOBJECT. It is further parameterized with three functions that are to be filled with other building-blocks:

outerSymbol describing the creation of the "outer" symbol (IOBJECT to VOBJECT)

outerToInner describing the relationship between the "outer" model element and its corresponding "inner" elements (IOBJECT to COLLECTION<IOBJECT>)

innerSymbol describing the creation of an "inner" symbol (IOBJECT to VOBJECT)

Especially the parameter OUTERTOINNER deserves special attention as it may not be identified with a building of the types as introduced above but describes a kind of *query* over the enterprise model. A query therein represents a function whose input and output are IOBJECTs, i.e. which remains in the domain of the enterprise model. Examples of possible queries are:

FollowRelationship describing a function that navigates from an enterprise model element to a directly related one. The signature of this query depends on the cardinality of the corresponding relationship.

FilterType describing a function that filters enterprise model elements according to their type. (COLLECTION<IOBJECT> to COLLECTION<IOBJECT>).

Exemplifying how a stakeholder can use these building-blocks to define an ad-hoc viewpoint based on an integrated information model for MEMO (cf. Figure 4), we assume the subsequent concern: "processes and their performing actors". By doing so the stakeholder specifies the selection of desired facts that he wants to be visualized (cf. requirement *selection & combination* described in Section 1). The corresponding concern thereby does not have to cover only a single perspective on the enterprise, i.e. does not necessarily derive its facts only from a single modeling language. The concern, more precisely its constituing concepts, are bound to different viewpoint building-blocks hence determining the *presentation* of the corresponding facts. Figure 9 shows the result of such binding,
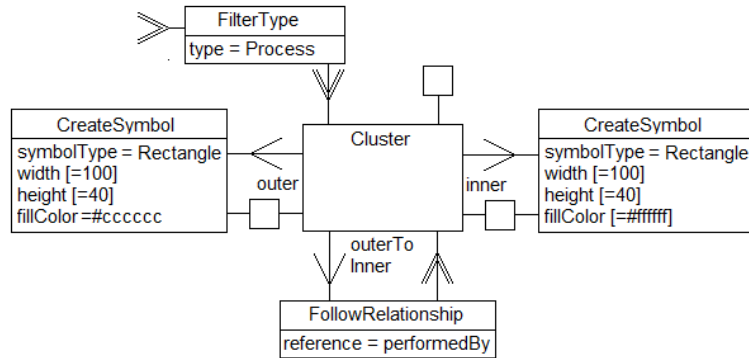
Figure 9: Defining the viewpoint for the simple clustered view

where a CLUSTER is linked to two corresponding CREATESYMBOL instances. These instances generate rectangular symbols for PROCESS instances (outer) and ACTOR instances (inner). In addition the relationship from outer to inner is bound to the PERFORMS relationship from the corresponding information model. Figure 10 shows an exemplary view created by applying the viewpoint from Figure 9 to a fictional semantic model from the hotel management domain.
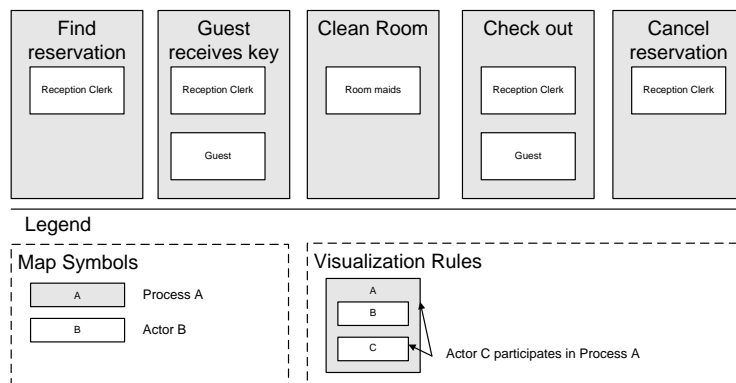

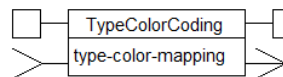Figure 10: Exemplary view clustering actors into processes

Figure 11: Building-block TYPECOLORCODING

Concluding the exemplary exposition of building-blocks, we introduce the decorator building-block TYPECOLORCODING (cf. Figure 11). In the sense of the *decorator pattern* described by Gamma et al. in [GHJV94] this building-block is used to augment a symbol

building-block with additional functionality, while not changing the overall signature. Put in other words, TYPECOLORCODING forwards its input IOBJECT to a symbol building-block, whose output VOBJECT is conversely forwarded to the output of TYPECOLOR-CODING. During this second forward, the type of the the IOBJECT is used to determine the fill color of the corresponding VOBJECT. For doing so, the buiilding-block is further parameterized with a set of TYPE-COLOR-mappings.
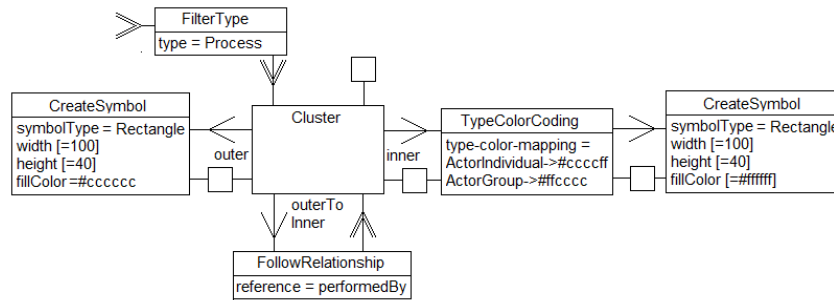

Figure 12: Defining the viewpoint for the decorated clustered view

Continuing the example from above and performing an ad-hoc adaptation of the viewpoint, we assume that the stakeholder decides to change the presentation of the information. More precisely, he wants to use color-coding to distinguish between instances of ACTORGROUP and ACTORINDIVIDUAL. For doing so, he adds the decorator viewpoint building-block introduced above to decorate the CREATESYMBOL defining the symbol representing ACTORS of arbitrary type. The result of the viewpoint definition after the performed adaptation is shown in Figure 12, whereas an example of a corresponding view is given in Figure 13.
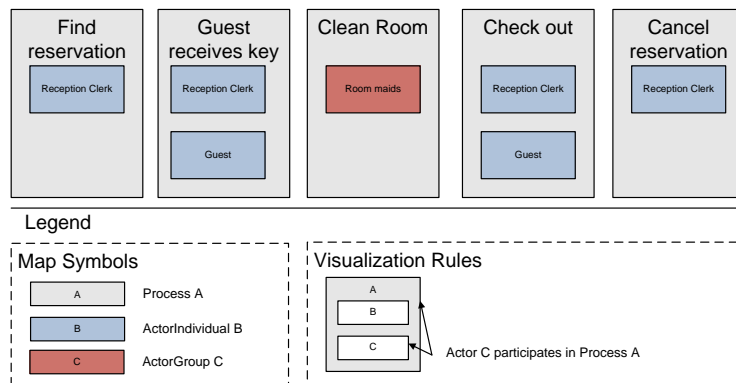

Figure 13: Exemplary view clustering actors of different types into processes

5 Outlook

Our results can further be elaborated by providing a closer integration between enterprise modeling languages and ad hoc analyses, e.g., by introducing constructs for interlinking model elements with concern-specific analysis views. This way, enterprise models would additionally serve as managing environments for navigating and accessing ad hoc analysis views. Stakeholders with different concerns would be able to go through the commonly understood enterprise model and invoke their specific analysis views when required. This idea would utilize existing enterprise models as models-at-runtime environments not only to provide previously captured knowledge for making it accessible for analyses, but to dynamically access appropriate analysis views from within enterprise models.

Furthermore, a set of template analysis views may be anticipated by language designers and be incorporated together with the set of enterprise modeling languages at the stage of language design. This would allow to prepare suitable forms of analyses and visual representations in the responsibility of scientifically grounded work of language designers. End-user stakeholders would be able to adopt the suggested analysis views to their specific concerns, and thus gain an additional increase in efficiency and reliability for performing ad hoc analyses, while preserving the full degree of flexibility provided by the described approach.

While the prefabrics of the presented approach have already been evaluated in practice, the method for enabling ad hoc visualizations has until now only been evaluated in terms of technical feasibility. This theoretical discussion will in a next step be evaluated in practice. Therefore, either case studies or an empirical survey, e.g. questionnaire, with enterprises willing to perform ad-hoc analysis can be conducted. We are planning to utilize the case study approach to support a more in-depth analysis of the tool support and to ensure inclusion of stakeholders with both business and IT background, which can provide feedback on the suitability of the chosen method and technique.

References

[ATL06] ATLAS group at LINA & INRIA. ATL: Atlas Transformation Language, 2006.

[BEL⁺07] Sabine Buckl, Alexander M. Ernst, J. Lankes, Florian Matthes, Christian Schweda, and André Wittenburg. Generating Visualizations of Enterprise Architectures using Model Transformation (extended version). *Enterprise Modelling and Information Systems Architectures – An International Journal*, 2(2):3–13, 2007.

[BKS10] Sabine Buckl, Sascha Krell, and Christian M. Schweda. A formal approach to Architectural Descriptions – Refining the ISO Standard 42010. In *6ᵗʰ International Workshop on Cooperation & Interoperability – Architecture & Ontology (CIAO2010)*, 2010.

[BMR⁺96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[BMS10] Sabine Buckl, Florian Matthes, and Christian M. Schweda. Interrelating Concerns in EA Documentation – Towards a Conceptual Framework of Relationships. In *2nd Eu-*

ropean Workshop on Patterns for Enterprise Architecture Management (PEAM2010), Paderborn, Germany, 2010.

[ELSW06] Alexander M. Ernst, Josef Lankes, Christian M. Schweda, and André Wittenburg. Using Model Transformation for Generating Visualizations from Repository Contents – An Application to Software Cartography. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), Munich, Germany, 2006.

[Fra94a] Ulrich Frank. MEMO: A Tool Supported Methodology for Analyzing and (Re-)Designing Business Information Systems. In R. Ege, M. Singh, and B. Meyer, editors, *Technology of Object-Oriented Languages and Systems*, pages 367–380, 1994.

[Fra94b] Ulrich Frank. *Multiperspektivische Unternehmensmodellierung – Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung*. Oldenbourg, München, Germany, 1994.

[FS06] Otto K. Ferstl and Elmar Sinz. *Grundlagen der Wirtschaftsinformatik*. Oldenbourg, München, Germany, 5th edition, 2006.

[FS09] Ulrich Frank and Stephan Strecker. Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems. Technical report, Institute for Computer Science and Business (ICB), University Duisburg-Essen, 2009.

[GF10] Jens Gulden and Ulrich Frank. MEMOCenterNG – A full-featured modeling environment for organization modeling and model-driven software development. In *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE '10)*, Hammamet, Tunisia, 2010.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional, Munich, Germany, 1994.

[Int07] International Organization for Standardization. ISO/IEC 42010:2007 Systems and software engineering – Recommended practice for architectural description of software-intensive systems, 2007.

[KW07] S. Kurpjuweit and R. Winter. Viewpoint-based Meta Model Engineering. In *Enterprise Modelling and Information Systems Architectures – Concepts and Applications, Proceedings of the 2nd Int'l Workshop EMISA 2007*, pages 143–161, St. Goar/Rhine, Germany, 2007.

[SBPM09] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF – Eclipse Modeling Framework (2nd edition)*. Addison-Wesley Longman, Amsterdam, 2009.

[Sch02] August Wilhelm Scheer. *ARIS – Vom Geschäftsprozess zum Anwendungssystem*. Springer, Berlin, Germany, 4th edition, 2002.

[Wie08] Jan Wiegelmann. *Analysis and Application of Model Transformation Languages for Generating Software Maps*. Bachelor thesis, Fakultät für Informatik, Technische Universität München, 2008.