Abstractions in Actor and Activity Modeling

Matthias Wester-Ebbinghaus, Daniel Moldt
University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D-22527 Hamburg
{wester,moldt}@informatik.uni-hamburg.de

Abstract: In this paper we argue that actor-centered models are well suited for socio-technical systems of systems (like enterprise and especially cross-enterprise scenarios). Results can especially be drawn from the field of multi-agent system modeling. However, existing approaches reveal a lack of possibilities for switching between different levels of abstraction. This does not only concern more or less abstract models for a given situation (model abstraction), but also to have models with actors of varying granularity, including individual as well as collective actors (actor abstraction). We present a modeling approach that addresses both these aspects. It is based on the core concepts of actors and activities and especially the concept of a collective actor is emphasized. For supporting different levels of model abstraction, we present multiple modeling techniques. The semantic core of all models is based on high-level Petri nets, although this is hidden for the more abstract models.

1 Introduction

Modern perspectives on software systems (e.g. *ultra large scale systems* [Nor06] or *application landscapes* [Mat08]) heavily rely on the concept of *systems of systems*. Core distinguishing features of systems of systems are the operational as well as managerial independence of component systems. In addition, they are software-intense *socio-technical* systems where a suitable joint optimization of social and (software-)technical system parts has to be pursued.

Agent-orientation has emerged as a powerful and general conceptual framework for socio-technical systems of systems. Both the idea of independent, loosely-coupled component systems (the agents) and the establishment of system-level/organizational structures are addressed equally [Jen00, Dig09].[1] As in a system of systems setting, a component system might be a system of systems itself, approaches for collective agency have been brought forth in recent years (cf. [HMMF08, AG09]). These are motivated by the social and especially organization-theoretic concept of collective actors: Collectivities (networks of social relations between social actors) are *Janus-faced*, social contexts as well as collective actors that are embedded in more global contexts.

Nevertheless, on the *modeling* side, there still exists what we call an *actor abstraction problem* as there is a lack of modeling techniques that allow for a seamless integration of

[1] Many of the concepts and ideas of agent-orientation nowadays appear in the "guise" of service-oriented systems (cf. concepts like autonomous, intelligent or collaborating services where the service metaphor is quite stressed and the agent metaphor is better suited). However, we will not deepen on this topic in this paper.

actors of varying granularity/abstraction. In this paper we present a modeling approach with the specific aim of addressing this problem. Our main goal is to support planning and design of actor systems, possibly including social as well as artificial actors. We address abstraction in two dimensions: (1) more or less abstract models for specific scenarios (*model abstraction*), (2) possibilities to integrate actors of different abstraction/granularity in scenarios (*actor abstraction*). For more or less model abstraction, we present three modeling techniques and each of the techniques incorporates the concept of collective actors.

2 Different Levels of Abstraction for Actor/Activity Models

So called *Actor/Activity-Flow Models* provide the semantic core for the more abstract models. They are high-level Petri net models and thus have a precise operational semantic. Figure 1 shows an Actor/Activity-Flow Model of two *Collectivity Units*. With a Collectivity Unit, we denote an entity that is a context for embedded actors and that may actually be an actor itself. A Collectivity Unit is modeled in terms of *actor positions* and *activity patterns*. Actors initiate activities, participate in them and may be added or removed from their positions in the course of activities. In the example, we show two activity patterns transaction and replace-seller for Collectivity Unit A and the associated actor positions seller, buyer, governor and administrator. An activity pattern consists of connected transitions that represent *actions*. Places connect the transitions in a way to make up the *life lines of activity roles* (vertically connected) and *exchanges between activity roles* (horizontally connected). For each activity pattern, we have one role that is the *initiating role*. In the example we have the two roles sender and receiver for the transaction activity pattern (where the sender is the initiating role) and two exchanges between them (for an item sent and the following acknowledge).

The example also includes a second Collectivity Unit B. This Collectivity Unit is actually an actor itself and is embedded in the context of Collectivity Unit A as a buyer.[2] The figure exemplifies that collective agency in our model results from what we call *role implementation via role implementation*. On the level of Collectivity Unit A, there is a buyer actor embedded that implements the role receiver in the context of a transaction activity and thus carries out the actions for receiving an item and for sending an acknowledge. Here however, this actor is itself a Collectivity Unit and in order to implement the role receiver on the next higher system level, there are the two internal roles distributor and inspector. For these internal roles, there are specified several *peripheral actions* that correspond to (/synchronize with) the actions for the role receiver on the next higher system level. To summarize, Collectivity Unit B implements the role receiver in the context of Collectivity Unit A by means of embedding some actors itself that in turn implement the roles distributor and inspector where peripheral actions synchronize with the actions of the receiver role. In general, such action recursion across system levels can be arbitrarily deep and individual actors (not modeled here) represent the recursion termination.

[2] Such hierarchies of Petri nets being embedded in other Petri nets is covered by the high-level Petri net concept of *nets-within-nets* [Val04] and is for example supported by the Renew tool (www.renew.de).

196

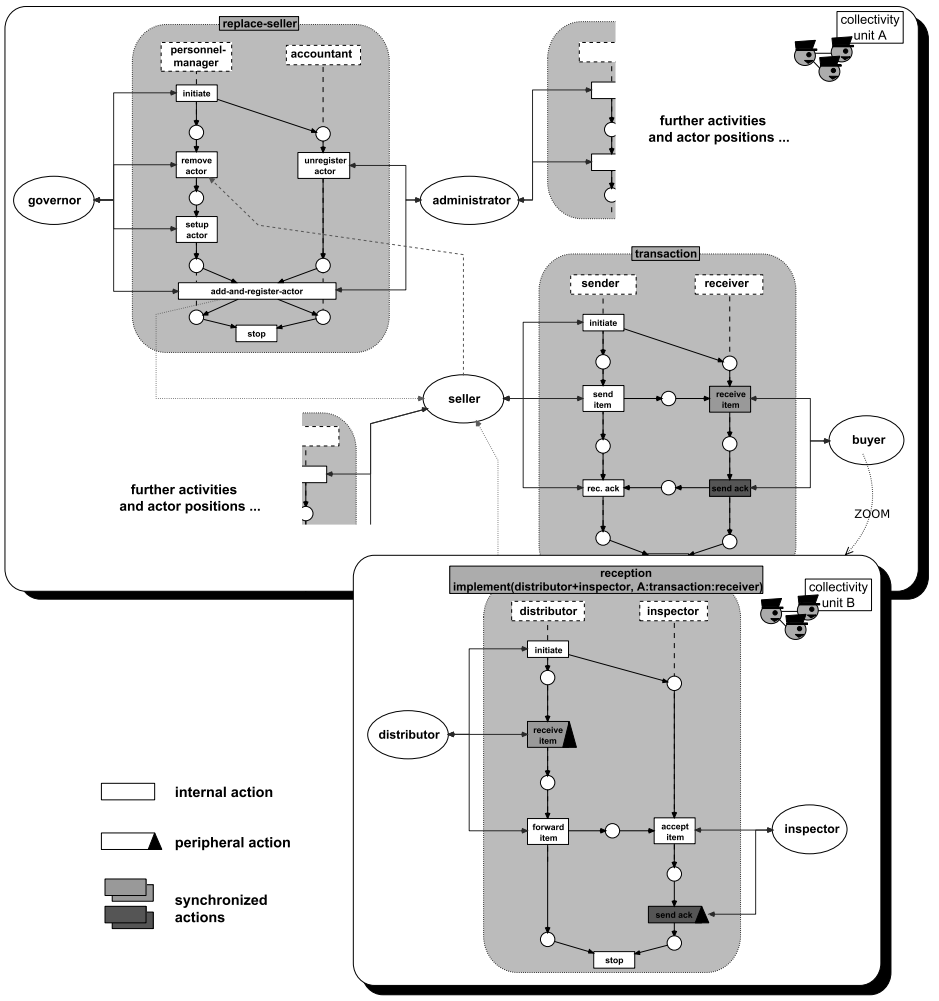

Figure 1: Excerpt from a more complex Collectivity Unit

Actor/Activity-Flow Models are quickly becoming large and unhandy. Thus we will present means for abstracting away from details. In the upcoming models, the underlying Petri net semantic is no longer directly visible but it still provides the basis for understanding them. The first abstraction step is in omitting details of activity flows and instead just referring to the roles of activities. Figure 2 shows an example of such an *Actor/Activity-Role Model*. It is the abstraction of the Actor/Activity-Flow Model from Figure 1. Actor positions are now just *associated* with activities (initiation, participation, addition, removal associations) and internal details of the activities are hidden. The information revealed is similar to common Use Case Diagrams. However, for our purpose to regard Collectivity Units as collective actors, we need an extension in the form of associations across system levels. We use a notation of combining activity roles via logical operators. Above, we have characterized collective agency basically as *role*

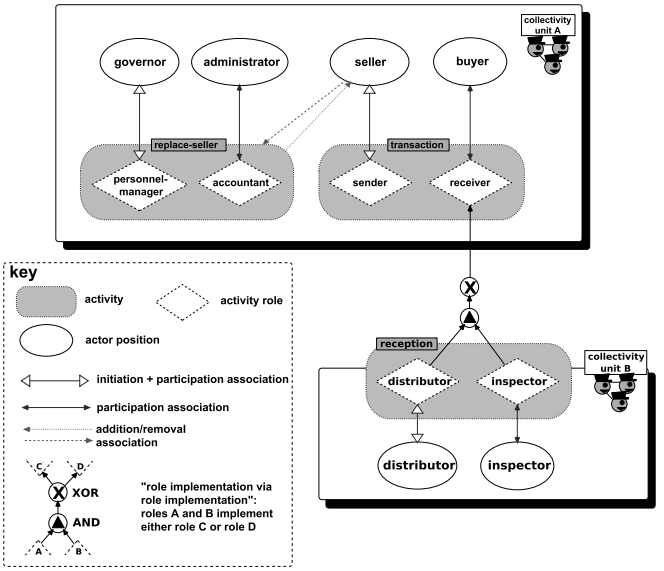

Figure 2: Abstracting away from the activity flows of Figure 1

implementation via role implementation. For Actor/Activity-Role Model models, we introduce AND/XOR configurations. For each AND/XOR configuration, we combine a set of roles of an activity pattern via a logical AND. For each instance (concrete activity) of the activity pattern, these roles together implement exactly one role of a set of roles that are combined with a logical XOR. In the example, it is simply modeled (just as before) that the implementation of the roles distributor and (logical and) inspector of a reception activity of Collectivity Unit B lead to the implementation of the role receiver of a transaction activity of Collectivity Unit A.

One might also want to consider additional logical operators for combining roles. For now, we have intentionally chosen the restricted means of AND/XOR configurations here as in our opinion, they lead to simple yet expressive possibilities for cross-level role associations. Two canonical applications are shown in Figure 3.

We introduce one further step of abstraction. Instead of regarding singular actor positions and activity patterns, we regard arbitrarily comprehensive *actor and activity sets* and consequently *Actor-/Activity-Set Models*. The transition from the former Actor/Activity-Role Models to Actor-/Activity-Set Models is shown in Figure 4. On the top of the figure is (one level of) the previous Actor/Activity-Role Model. A preliminary abstraction step is shown in the middle of the figure where each actor position and each activity pattern is regarded as a singleton actor or activity set respectively. Addition/removal associations are now subsumed under participation associations. Based on this basic Actor-/Activity-Set Model, different more abstract models can be derived. We call these different *perspectives* on the initial model and we show some of them at the bottom of the figure. For each perspective it is noted which set unions were applied. Here, we distinguish between *absolute* associations between an actor set A and an activity set B (each sub-set $A_{sub} \subseteq A$ has this association to one of the sub-sets $B_{sub} \subseteq B$) and *contingent* associa-

198

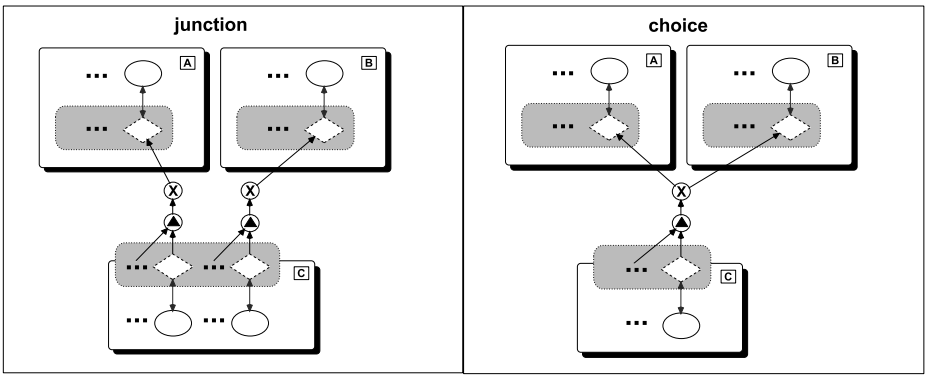

Figure 3: Examples of embedding situations based on AND/XOR configurations

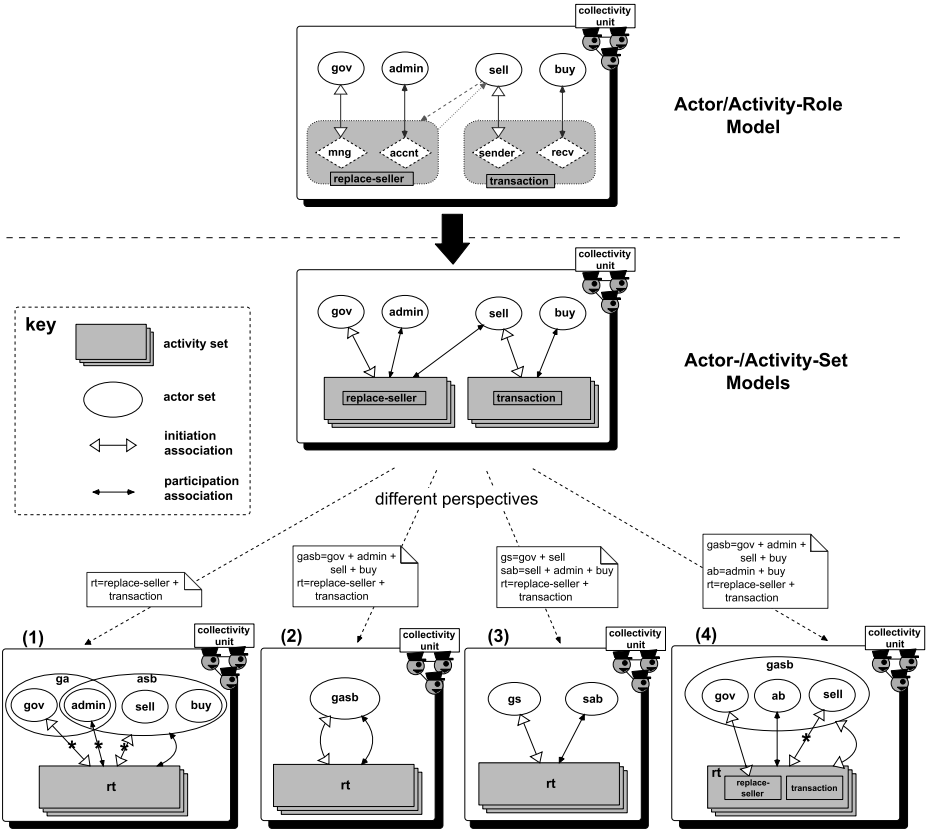

Figure 4: Transition from Actor/Activity-Role Models to Actor-/Activity-Set Models

tions (for at least one but not for all sub-sets $B_{sub} \subseteq B$, there is an association to one of the sub-sets $A_{sub} \subseteq A$).

199

For Actor-/Activity-Set Models it also has to be taken into account that Collectivity Units can be collective actors. For this purpose, it is no longer regarded which activity *roles* are cross-level associated but just which activity *sets* on different levels have which "interleaving" associations (for example, whether an interleaving is specified for each or just for some activities of an activity set). Due to space limitations, we do not cover the corresponding modeling primitives here.

3 Conclusion

In this paper, we present a modeling approach for actor/activity models that explicitly addresses abstraction in two dimensions: (1) more or less abstract models for a given scenario (model abstraction) and (2) more or less actor granularity (actor abstraction). The results we present are part of our ongoing research. We yet have to provide tool support for our modeling approach. Furthermore, up to now the models were mainly used as means to conceptualize quite generic and abstract archetypes of *organizational units* in multi-level organizational scenarios (cf. [WE10], in german). For the future, we need to validate the *practical use* of the modeling approach by applying it to real-world scenarios.

References

[AG09] AOS-Group. Jack Intelligent Agents Team Manual. Available at: http://www.aosgrp.com/documentation/jack/JACK_Teams_Manual_WEB/index.html, 2009.

[Dig09] Virginia Dignum, editor. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Information Science Reference, 2009.

[HMMF08] Christian Hahn, Cristian Madrigal-Mora, and Klaus Fischer. A Platform-Independent Metamodel for Multiagent Systems. *Autonomous Agents and Multi-Agent Systems*, 18(2):239–266, 2008.

[Jen00] Nicholas Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.

[Mat08] Florian Matthes. Softwarekartographie. *Informatik-Spektrum*, 31(6):527–536, 2008.

[Nor06] Linda Northrop. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, Carnegie Mellon, 2006.

[Val04] Rüdiger Valk. Object Petri Nets: Using the Nets-within-Nets Paradigm. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer, 2004.

[WE10] Matthias Wester-Ebbinghaus. Von Multiagentensystemen zu Multiorganisationssystemen – Modellierung auf Basis von Petrinetzen. Dissertation, Universität Hamburg, Fachbereich Informatik. Elektronische Veröffentlichung im Bibliothekssystem der Universität Hamburg: `http://www.sub.uni-hamburg.de/opus/volltexte/2011/4974/`, 2010.