Modelling Distributed Network Security in a Petri Net- and Agent-Based Approach

Simon Adameit[2], Tobias Betz[2], Lawrence Cabac[2], Florian Hars[1], Marcin Hewelt[2], Michael Köhler-Bußmeier[2], Daniel Moldt[2], Dimitri Popov[2], José Quenum[2], Axel Theilmann[1], Thomas Wagner[2], Timo Warns[1], and Lars Wüstenberg[2]

[1] PRESENSE Technologies GmbH, Hamburg
[2] University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences, Department of Informatics
http://www.informatik.uni-hamburg.de/TGI/

Abstract. Distributed network security is an important concern in modern business environments. Access to critical information and areas has to be limited to authorised users. The Herold research project* aims to provide a novel way of managing distributed network security through the means of agent-based software. In this paper we present the first models, both conceptual and technical that have been produced in this project. Furthermore we examine the Paose development approach used within the project and how it contributes to Herold.

Keywords: Distributed network security, Software agents, Petri nets, Agent-oriented methodologies.

1 Introduction

Computer networks have become more and more omnipresent in recent years. Accompanying this trend, the need for adequate, versatile and flexible distributed systems has risen as well. Distributed systems capitalise on the properties of the different network nodes and use them to provide functionality difficult and inefficient to obtain in classical, centralised systems. A paradigm particularly suited to design distributed systems is agent-orientation. The use of autonomous entities, called agents, which use asynchronous messages for communication or can (to a certain degree) intelligently adapt to unforeseen circumstances makes this paradigm perfect for modelling and implementing distributed systems.

Of course distributing functionality over open networks introduces security risks and issues. Questions that arise are how to protect critical data and how to make sure only authorised users can execute sensitive actions. Classically networks are protected using a perimeter model in which network security components (NSCs) protect the virtual *border* of the network. The problem with this

* The Herold research project is supported by the German Federal Ministry of Education and Research (Grant No. 01BS0901). For further information see http://www.herold-security.de

J. Dix and C. Witteveen (Eds.): MATES 2010, LNAI 6251, pp. 209–220, 2010.
© Springer-Verlag Berlin Heidelberg 2010

approach is that once the perimeter of the network is breached, the attacker can access the rest of the network almost unimpeded. The Herold research project follows another approach. Instead of relying on a perimeter to protect a network we divide the network into multiple subnetworks, called cells, each protected by their own NSC. The overlapping cells together form the entire network and the NSCs enforce a network-wide security policy in a cooperative effort. This way, if one NSC is breached, only part of the network is open to attacks. The other cells are unaffected.

Further aspects of interest that will be discussed in this paper include for example the use of abstract, global policies that are automatically translated into efficient, localised configurations for NSCs.

To implement our prototype we use the Petri net-based agent architectures Mulan and Capa. Agents within these architectures possess many properties, like mobility and proactiveness, which make them very versatile and allow and encourage their use within the context of distributed network security. Additionally the Paose (**P**etri net-based and **a**gent-**o**riented **s**oftware **e**ngineering) software development approach is especially well suited for these two architectures, so that our work is supported by an elaborate and functional approach.

In this paper we present both a conceptual and technical model of how to realise part of the overall Herold vision. The technical model we will describe is one of the first prototypes developed within the Herold Paose cycle and will serve to show the results of using this approach both in regard to the target domain as well as to the versatility of the approach. We will also offer an outlook on possible extensions to our current model.

The paper is structured in the following way. Section 2 distinguishes our approach from related work. In Section 3 we will outline the theoretical and technical background of our work. Section 4 describes the conceptual view of the overall Herold vision and the current Herold model, while Section 5 examines parts of the implementation. The paper concludes in Section 6 and gives an outlook on future work.

2 Related Work

Herold lies at the intersection of multi-agent systems, policy-based management, and network security. Different approaches have been proposed in the literature that also address these or related domains. Herold differs from previous work in its unique combination of an agent- and policy-based approach to network security, the automatic transformation of global policies to NSC configurations (i.e., "refinements"), and provisioning network security as a cooperative effort.

Different approaches on agent-based network management have been proposed in the literature [7,1,10,5]. In principle, Herold is closely related to such approaches while, however, focussing on network security and solely relying on stationary agents. In particular, Herold differs from most previous approaches for being policy-based, addressing the localisation of policies, and transforming policies to NSC-specific configurations.

Closely related to Herold, Uszok et al. [13] have presented an approach to the representation, conflict resolution, and enforcement of policies for the agent framework *KAoS (Knowledgeable Agent-oriented System)* [2]. Besides using a different policy language, focussing on network security, and having a different architecture, Herold can be considered a generalisation of this approach as the KAoS approach does not address refinements of policies.

Ponder2 [12] is a prominent example of a policy-based management system that supports refinements of policies. In contrast to Herold, it relies on a Peer-to-Peer architecture instead of an agent-based architecture. Moreover, its concept of refining policies significantly differs from Herold's policy transformations: Having policies defined in terms of hierarchical "domains," Ponder2 allows to refine policies by adding additional objects to these domains or by moving policies down in the domain hierarchy. In contrast, Herold refines policies by mapping them to NSC-specific configurations defined in a different language than the policy itself (using NSC-specific transformations).

3 Background

Our agents follow the Mulan architecture described in [11]. Mulan stands for **Mul**ti-**a**gent **n**ets, a name that perfectly describes the main idea behind the approach. Every aspect of agents in Mulan is modelled with reference nets, a high level Petri net formalism described in [9]. Following the nets-within-nets idea examined, for example, in [14], tokens of reference nets can again be reference nets, thus allowing for a hierarchical nesting of nets, which can interact with one another. With the help of this formalism it is possible to model the different layers of the Mulan reference architecture. The lowest level is represented by agent behaviour. Behaviour is modelled through so-called protocol nets, which are tokens in the next layer, the agent layer. The agent nets are located in their runtime environment, the platform nets, which themselves are located in system nets. This way it is possible to model a complete multi-agent system using four levels of interacting nets-within-nets. Capa (Concurrent Agent Platform Architecture) is an extension to Mulan focussing on interoperability and communication, which was described in [6]. It provides full compliance with the standards of the Foundation for Intelligent Physical Agents (FIPA). It does so by providing a standardised message format for the Mulan agents and by replacing the upper Petri net layers of the Mulan reference architecture. These layers are now implicitly defined by actual communication relations between different platforms. The Petri net editor and simulator Renew serves as the build- and runtime environment for systems using Mulan and Capa. A description of Renew can be found in [9] or on the website `www.renew.de`.

The Paose approach has been described, for example, in [3] and [4]. The accompanying development methodology is especially suited for software development within the Mulan and Capa architectures, but is general enough to be used in many other contexts. The aspect most relevant for this paper is rapid prototyping, which gives the approach some agile properties. In general

Paose exhibits concepts and ideas from many other approaches. First steps within Paose use subsets of UML to define coarse designs, which are directly translated into Petri net models. These models only need to be slightly modified to provide the desired functionality, so that they almost directly correspond to prototypes within the approach. These prototypes are then recycled in further iterations. In this way the Paose methodology contains aspects of UML, model driven approaches and agile software development.

4 Conceptual View

Before going into the details of our conceptual model we must first describe the overall vision and approach behind Herold. Generally speaking the Herold project aims to provide a novel, agent-based approach of managing and controlling NSCs, both active (e.g. firewalls) and passive (e.g. intrusion detection systems). Security attributes and rules are defined in an abstract global policy that covers the entire network. This policy is enforced locally and cooperatively by the NSCs under the control of a Herold system. The global policy is created and maintained by different network administrators cooperatively and is automatically transformed into technical configurations for the NSCs. These technical configurations only contain the information relevant and necessary for the NSC they are deployed on. Information and rules that are irrelevant to a certain NSC, for example information about network events that, due to network topology, cannot possibly occur within the scope of this NSC, are removed from the configuration for this NSC. This so-called localisation of policies ensures efficiency in policy enforcement by keeping the policies at the smallest size needed. This approach entails that the NSCs, which are distributed within the network, are responsible for the security of their own compartment, or cell, of the network. This cell-based approach to network security improves security compared to a more classical perimeter approach, in which NSCs are located at the border of the supervised network and form a single line of defence against attackers. In the perimeter approach an attacker that breaches or circumvents one NSC can, in the worst case, access the entire network. With the cell-based approach an attacker can only access the (small) part of the network that lies within the cell of the NSC breached and is controlled by no other NSC.

The control and configuration of the NSCs are handled by autonomous software agents, which interact to cooperatively ensure the enforcement of the global policy. Regarding the NSCs as nodes of a distributed system it is possible to naturally map agents and agent concepts into the Herold system. These agents provide means of defining the global policy, allowing different administrators to edit the global policy concurrently, transforming the abstract global policy into technical configurations, localising the global policy for each NSC under their control and finally deploying the localised and technical configurations onto the NSCs. Basically, the agents are responsible for accepting the global policy as an input from the administrators, keeping it consistent for all users and transforming and deploying it correctly.

The ambitious goal of designing and implementing the Herold approach described above covers several iterative models. We will now describe the first complete Herold model in its conceptual idea. Based on this concept and in accordance to the PAOSE approach the following section will present how part of the functionality of our concept has already been implemented in a prototype. The model supports the key concepts of managing NSCs through agents and localising policies for individual NSCs. The localisation is handled with the concepts introduced in [8] and will not be further discussed in the scope of this paper. There are three general aspects of our model which have to be described: the network model, the policy model and the use cases. We will now examine these three aspects.

Network Model. The model assumes a connected network topology, where each node has a unique address. While the network topology is not described explicitly, the Herold users provide a description of the NSCs that are used to execute the global policy. The description of a NSC includes its network interfaces, the associated addresses, a routing table, and information on how to configure the NSC. This information is used when localising the global policy for the respective NSC. As the Herold system is not aware of the network topology with this model, some types of NSCs may not be covered by this model. Handling such NSCs is deferred to more elaborate models with a network model that includes information on the topology of the network. Besides the descriptions of the NSCs, the network description additionally covers groups to organise the network components and the assignment of network components to groups.

Policy Model. The users of the Herold system share a single, common global policy that is always active. The policy is defined in terms of 5-tuples with unique addresses and (logical) groups as defined by the network description. The tuples consist of two pairs of source and target address and port and the action. Semantically, the policy contains a rule with an *allow* or *deny* action for each possible 5-tuple. The groups, however, allow to syntactically describe the policy more concisely. The Herold system localises the global policy for each managed NSC and transforms the localised policy to a NSC-specific configuration format. The Herold system then deploys the configurations to the respective NSCs.

Use Cases. The model includes use cases on editing the global policy and network descriptions, on policy transformations and on configuration deployment. Herold users edit a single global policy and a single network description. Multiple users may edit the policy and the network description concurrently, which requires an explicit design decision on how to cope with issues arising from concurrent editing (e.g., the lost update problem). The use cases are:

- *Use cases concerning the global policy* (View the current list of rules and the status of the system; add/delete/modify/move a rule)
- *Use cases concerning the network description* (View the current set of NSCs; add/delete/modify a NSC; view the current set of groups; add/delete/rename a group; add/remove network components in a group)

The NSCs, as external systems, are actors that interact with Herold in a single use case, namely the deployment of the NSC-specific configurations.

As mentioned before this model is just one step on the way to the overall Herold vision. However it already incorporates the key aspects of Herold: Cooperative creation of a global policy, localisation of this global policy and automated transformation into configurations for NSCs. The core of the Herold approach is already present within this model and it is possible to extend and enhance this model towards the overall vision.

A quite interesting aspect of the Herold project in general is how the Paose approach plays into the development of the Herold system. The approach allows us to follow a fast iterative process, which produces many prototypes each incrementing and enhancing the last one. In a way the conceptual model described in this section can be seen as only one prototype on the way to the overall vision. But not only the iterative process naturally supports the development of Herold. The self-organisational aspects of the approach and the division of the target context into three dimensions (roles, interactions, ontology) naturally support the development of systems that have a distributed context, like Herold.

5 Implementation

We will now present the implementation of a subset of the functionality provided by our conceptual approach. We have chosen this subset of our model for two main reasons. On the one hand this subset is small and simple enough to be presented in the scope of this paper, on the other hand the prototype implementing this subset highlights the very important prototyping aspect of our Paose approach, discussed above. It is the first prototype created for this model and is referred to as *model zero*.

Model zero introduces the key functionality of the Herold application, in its simplest form. The model covers system administrators (acting as users) who interact with a global policy, described as a list of rules. From these interactions, the resulting policy is localised and deployed to a NSC.

The network submodel of model zero assumes a space of unique addresses with subnets with perfect routing, but without any further network topology and no explicit network description. The network submodel has no concept of network locations or NSCs and also does not support the grouping of components. The policy submodel assumes a single, global policy that is always active and that talks about communication relations that are characterised by protocol, source and target addresses and source and target ports. Every policy is assumed to be total, i. e. it implies an *allow* or a *deny* judgement for each of the possible communication relations per protocol. Since it is impossible to specify all individual rules, the policy may be written as an ordered list of rules where each rule is written as a pattern that can match many possible communication relations. The list is evaluated with a "first match wins" semantics. To make the policy total, there is an implicit last rule that matches everything and specifies

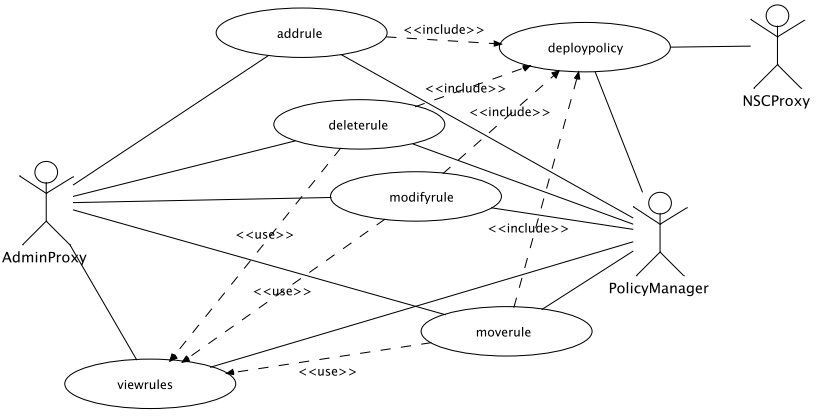

Fig. 1. Model zero - Use case diagram

a default decision for this policy. Since the network model is non-local, there is no localisation of policies. In a way the policy can be seen as the configuration of a single, omnipotent NSC.

For model zero a subset of the overall model containing six use cases has been identified. Figure 1 depicts the use case diagram.

deploy policy. In this use case the current global policy is configured and deployed into a NSC. This is the only use case in model zero that is not directly called by an actor, but is included in all of the other use cases that in some way change the global policy. It is called whenever the global policy has been successfully changed and before the lock on the global policy is lifted.

add rule. This adds a new rule to the policy at a given evaluation order position.

delete rule. The delete rule use case removes a rule from the global policy.

modify rule. This use case modifies one rule of the global policy. The parameters within this interaction are both the old and new rule.

move rule. The move rule use case can be seen as a special case of the modify rule use case, in which the actual rule itself is left unchanged and only the position of the rule within the global policy is modified.

view rules. This use case returns all the rules of the current global policy.

The use cases involve three actors, namely:

Admin Proxy. This represents a system administrator. He has administrative control over the global policy and can thus initiate the different use cases.

Policy Manager. This actor represents an entity, which is in charge of managing the central list of rules.

NSC Proxy. This actor represents the NSC within the use cases. It is only involved in the deploy rule use case, since all other operations only work with the global policy managed by the policy manager.

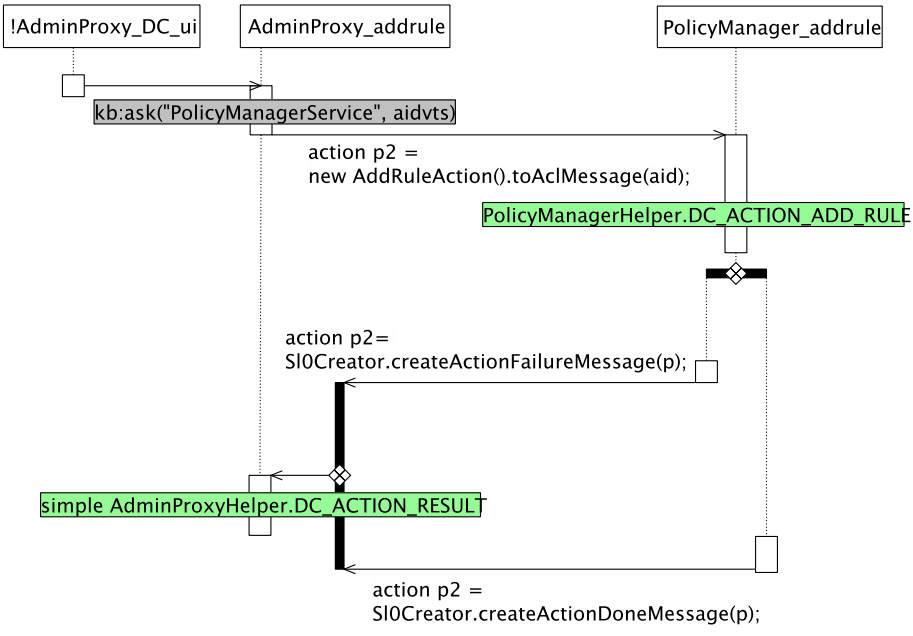

Fig. 2. Interaction protocol for the add rule use case

From these use cases it is possible to automatically generate the artefacts needed for the next modelling steps using the Renew toolset. In the case of Mulan and Capa the use cases are realised through interactions between agents, which themselves correspond to the actors. These interactions are defined through agent interaction protocols, from which the actual agent behaviour protocols can be generated. Behaviour outside of these interactions (e.g. internal decision making) is modelled through decision components which generally serve as the internal behaviour of agents. An agent's knowledge is defined in it's knowledge base that is defined during modelling but can change during execution.

We will now exemplary discuss one of these interactions in detail. We have chosen to illustrate the add rule use case/interaction for this example, since it shares much of its structure with other use cases, as is discussed below. Figure 2 shows the agent interaction protocol for this use case.

The interaction is started by the *AdminProxy_DC_ui*, which represents the user interface for an administrator. The other two actors in this diagram represent the agent protocols, which model the behaviour of the agents which correspond to the admin proxy and policy manager actors of the use case. The first action the *AdminProxy_addrule* protocol executes is to look up and find the address of the policy manager from the knowledge base. Once this is done a message is sent to the policy manager containing the request to add a rule, as well as information about the new rule. This is then passed internally over to the decision component of the policy manager agent. Within this decision

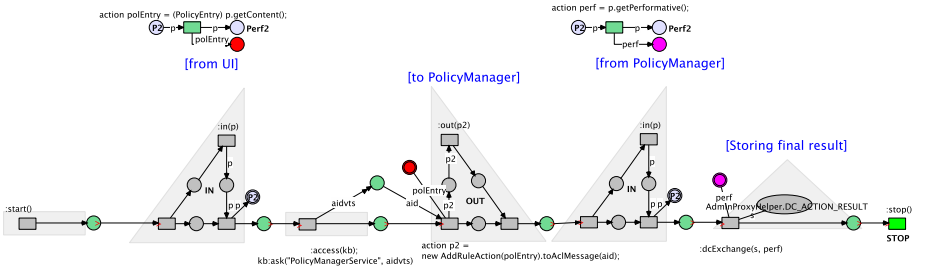

Fig. 3. Agent protocol for the admin proxy actor in the add rule use case

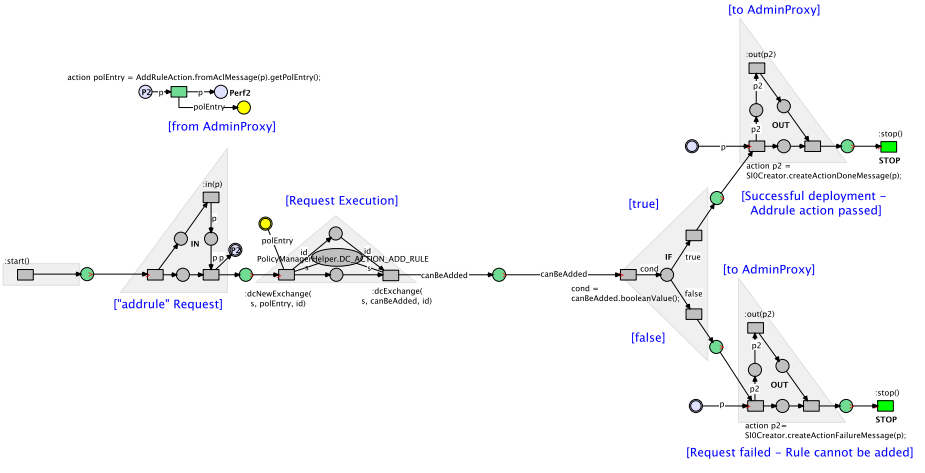

Fig. 4. Agent protocol for the policy manager actor in the add rule use case

component the new rule is evaluated and added to the global policy. If this has succeeded, the deploy rule interaction is called and the new policy deployed in all known NSCs. The decision component replies to the protocol with either a failure or done, depending on whether problems occurred at any step. This result is then sent as either an ActionDone or an ActionFailure message to the admin proxy protocol and passed to a decision component, which stores the results of interactions for later inspection. This concludes the interaction/use case. From this agent interaction protocol the stubs for the agent protocols can be automatically generated using the Renew toolset. These stubs only need a minimal amount of change in their inscriptions until they are complete and model the functionality described above when executed. The resulting final two protocols can be seen in Figures 3 and 4.

5.1 Assumptions and Simplifications

Mainly, our assumptions and simplifications are related to both the *interaction protocols* and the *concurrent* access to the policy pool.

Even though we did not elaborate on all the use cases, one can straightforwardly see that they all rely on the same pattern: identify a single entity and make it execute a functionality on your behalf. This pattern is well captured in the standard FIPA request protocol. A functionality requester, the *initiator*, contacts a functionality provider, the *participant*, to request the provision of a functionality. After examining the request, the participant first notifies the initiator of whether it agrees to providing the functionality or not. Also, in case an error occurs while examining the request, the initiator is notified as well. In case the participant agrees to executing the functionality, it does so and sends the initiator the result. The final result can then be communicated either as a simple notification, a reference to a generated object or a notification of failure.

Given the simplicity of our model, we assumed that an agent enacting the participant role will always accept a functionality whenever contacted. As well, we minimised any message examination error. Thus, we shortened the request protocol and removed the acceptance phase.

Another careful consideration during the modelling exercise is with respect to *concurrency* control, i.e., how to provide concurrent access to the global policy. Indeed, despite the simplicity of the current model, concurrency remains one of its pillars. Although we pondered over several possible solutions, we finally chose a *pessimistic lock* approach for the sake of simplicity. Only *write* operations (add rule, move, modify and delete rules) are considered for lock. Thus, at any time one can view the current set of rules in the policy. The scope of the lock is the entire policy. In the future, we envision to narrow the scope down to a subset of rules and if necessary to a single rule. In doing so we will easily cope with, for example, partial policies, while improving the performance of our models.

5.2 Step-by-Step Modeling

In this section we briefly take the reader through the different steps that led to the Petri net models, which control the behaviour of the agents. As mentioned earlier these steps fall into the Paose guidelines.

1. *Use cases:* Identify the actors and draw the use case diagrams.
2. *Ontology objects:* Ontology objects help define the concepts referred to in the content of agent messages.
3. *Interaction diagrams:* Elaborate on the interactions defined in the use case diagrams.
4. *Interaction models:* Using a Renew plugin, we automatically generated the Petri net models corresponding to the interaction models (*agent protocols*).
5. *Decision components:* We draw the nets that support internal decision makings for each agent.
6. *Knowledge Bases:* We configure the knowledge base for each agent.

Going through this process as a cycle yields multiple prototypes, each improving upon the functionality and stability of the previous version. In the model zero prototype, we packaged the necessary aspects to model the key functionality

of the Herold application. Note that this model bears many simplifications and can only serve as a proof of concept. Further prototypes will extend model zero in various regards, like network modelling, localisation and policy management, until a satisfactory version of our overall model is achieved.

6 Conclusion

In this paper we presented our work within the Herold research project. We examined the overall Herold vision and presented our conceptual approach to realising this vision. We detailed the important aspects of the conceptual model and then proceeded to describe one of the prototypes implementing a first, important subset of the overall, conceptual functionality. The description detailed the submodels and the general use cases of the prototype. For one of these use cases the actual implementation was exemplarily presented. Afterwards the prototype and some design decisions were discussed.

Further prototypes are already being developed. They incorporate more concepts described in the conceptual model and in the overall vision. Aspects of the overall vision being addressed are:

- **Policies:** One aspect lies in handling the global policy. For example the support of partial policies is planned, which do not cover the entire space of possible events. These would have to be completed internally, but would make handling of policies easier for the user. Another point here would be the provision of a policy pool and policy templates, which would make the generation of policies easier. The support of obligation policies in addition to the current authorisation policies and the relationship of the policy model to the network model are also being examined.
- **Network model and localisation:** Another aspect regards the network model and the localisation. The overall vision includes an explicit, expressive network model in order to localise the global policy in the most efficient way. This network model needs to be expressive enough to describe realistic settings, but still easy to use. The network model and the related complex localisation algorithms are important aspects of the Herold project and will be addressed in the near future.
- **Transformation into configurations:** Another aspect that has not been discussed in detail in this paper is the transformation from the (abstract) policy to the actual technical configuration files for the NSCs. This affects the policy model, since the transformation into different specific configuration languages has to be supported.

By following the Paose development approach the Herold project is supported by an approach that naturally maps concepts of the target domain into the development cycle. The fast prototyping encouraged by the approach allows us to iteratively enhance the produced systems. In conclusion the overall Herold project aims at providing a surplus within the domain of distributed network security. The first steps within this endeavour have been presented in this paper.

References

1. Bieszczad, A., Pagurek, B., White, T.: Mobile Agents for Network Management. IEEE Communications Surveys 1(1), 2–9 (1998)
2. Bradshaw, J.M., Dutfield, S., Benoit, P., Woolley, J.D.: KAoS: toward an industrial-strength open agent architecture. In: Software Agents, pp. 375–418. MIT Press, Cambridge (1997)
3. Cabac, L., Dörges, T., Duvigneau, M., Reese, C., Wester-Ebbinghaus, M.: Application Development with Mulan. In: Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07), pp. 145–159 (2007)
4. Cabac, L.: Multi-Agent System: A Guiding Metaphor for the Organization of Software Development Projects. In: Petta, P., Müller, J.P., Klusch, M., Georgeff, M. (eds.) MATES 2007. LNCS (LNAI), vol. 4687, pp. 1–12. Springer, Heidelberg (2007)
5. Du, T.C., Li, E.Y., Chang, A.-P.: Mobile Agents in Distributed Network Management. Communications of the ACM 46(7), 127–132 (2003)
6. Duvigneau, M.: Bereitstellung einer Agentenplattform für petrinetzbasierte Agenten. Diploma thesis, University of Hamburg (2002)
7. Goldszmidt, G., Yemini, Y.: Delegated Agents for Network Management. IEEE Communications Magazine 36(3), 66–71 (1998)
8. Großklaus, A.: Policybasierte Konfiguration von verteilten Netzwerksicherheitskomponenten. Diploma thesis, University of Hamburg (2007)
9. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
10. Puliafito, A., Tomarchio, O.: Using Mobile Agents to Implement Flexible Network Management Strategies. Computer Communications 23(8), 708–719 (2000)
11. Rölke, H.: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen. Logos Verlag, Berlin (2004)
12. Twidle, K., Lupu, E., Dulay, N., Sloman, M.: Ponder2 – A Policy Environment for Autonomous Pervasive Systems. In: Proceedings of the 2008 IEEE Workshop on Policies for Distributed Systems and Networks (POLICY '08), pp. 245–246. IEEE Computer Society Press, Los Alamitos (2008)
13. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. In: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 93–98. IEEE, Los Alamitos (2003)
14. Valk, R.: On Processes of Object Petri Nets. Technical Report FBI-HH-B-185/96, University of Hamburg, Department of Computer Science (1996)