A Policy-based Model-Driven Security Framework

Peter F. Linington
University of Kent, UK
pfl@ukc.ac.uk

Abstract

The adoption of a model-driven approach to the construction of applications places the focus on business logic and takes it away from detailed middleware mechanisms. It also opens new opportunities for more detailed and more dynamic control of non-functional properties. This position statement illustrates the possibilities by considering the ways in which maintenance of security infrastructure can exploit the model-driven approach.

1. MDA and policies

The basic idea, on which model-driven architecture is based, of using a powerful tool-chain to create complete applications together with their support environment from descriptions of organisations and their business processes has been a widely acknowledged objective of distributed systems development for many years. It is at the core, for example, of the Open Distributed Processing (ODP) architecture [1], which was finalised in 1995. It is only now, however, with the increased strength of modelling and transformational tools, that the goal is within reach. The concept of a model driven architecture was given a major impetus by its strategic adoption within the OMG in 2000 [2].

In ODP, the system specification is divided into a number of viewpoints, including an enterprise viewpoint [3] expressing organisational or policy constraints, and a computational viewpoint expressing the required business logic. The engineering viewpoint specifies a set of templates that are used by the tool-chain to create the necessary supporting infrastructure for the applications. The functional and non-functional requirements stated in the enterprise and computational viewpoints are used by the tools to select an available engineering template for instantiation.

In addition to the business-specific models, however, there will be a number of models expressing organisation-wide structures and policies; merging these different models into a single set of requirements implies that a model-driven architecture tool-chain will need to incorporate aspect-oriented techniques.

Policies governing non-functional concerns, such as resource usage or security, are likely to be structured hierarchically; there will be some organisation-wide policies that are refined by more specific policies established by individual sub-units.

This position paper makes some general points about the form of policies needed in such a framework (in section 2) and then outlines (in sections 3, 4 and 5) a specific example of model-based policy currently being investigated at Kent. It ends with a discussion of the requirements for the evolution of such systems (in section 6).

2. Policies and policy envelopes

Much of the current work on policy-based management focuses on the definition of powerful and flexible notations for the expression of policy, on the assumption that decisions to be taken within the system will best be expressed in a notation that reflects the design process undertaken by a suitably authorised manager or administrator. There is an assumption that the design created is consistent with the over all aims of the organisation concerned. In practice, however, there are generally strict constraints on what policies are acceptable, given that existing organisation-wide policies are already in place. In current architectures, this is often reflected as a requirement to detect conflict between different

policy definitions, and to establish rules for the resolution of such conflicts. However, this gives only weak guidance to the definer of policies as to what freedom is intended, and gives no guarantee at all of future correctness when the organization-wide policies are updated.

The author believes that categories of policy should be established within an over-arching organizational architecture model, and that this process should include the definition of a *policy envelope*, so that it is clear to the maintainer of a particular policy what flexibility is allowed within the bounds established by the architecture. In this view, establishing a policy involves:

- defining, in the over-arching model, a scope or set of circumstances in which the policy is to apply;
- identifying some non-trivial choice to be made under the control of the policy (a specific set of rules);
- identifying an envelope that constrains the range of behaviours that can be specified for the choice made by the policy;
- identifying what information must be available from the environment for the policy to interpret in order to make the choice;
- defining a decision procedure to be applied in assessing the situation and in actually making the choice;
- defining any invariants that may need to be respected by the system in general for the policy to be effective.

This view of policy fits well with the model-driven architecture approach, since it views a hierarchy of policies as being represented by a hierarchy of models, and the definition of each model is constrained to be within an envelope stated in the more abstract model establishing its role. Thus each model defined as input to the tool-chain is interpretted in a context established by its place in the overall view of the enterprise; the constraints establish the envelope for that particular policy or model. This parallels the structures of delegated authority within the organization.

The resulting constraints form a contract between the definers of the different models, so that there should be no conflicts as long as the policy definition remains within its envelope, and so long as the definers of policies of broader scope respect the envelopes for more detailed policies they have defined.

This is, of course, an ideal to aim for, and the decidability of consistency with the envelope will often not be straightforward; both this, and the precision with which the envelope can be expressed, will depend on the specifics of the notations that are being used.

3. A JISC Initiative on Authentication, Authorisation and Accounting

The work at UKC most relevant to the views in this position statement has been carried out under the funding of the UK Joint Information Services Committee (JISC), which funds academic networking and content services for Further and Higher Education in the UK. This work examines the possibilities of integration between business models and security models, and so explores a facet of model-driven architecture perhaps unfamiliar to some middleware experts. As part of a larger initiative to encourage uniform authorisation mechanisms for a wide range of data services [4], the JISC has established a number of pilot projects on the use of strong security mechanisms, particularly certificate based schemes, on a national scale. As part of this activity they have funded some longer-term activities, including a study of the application of policies and model-driven techniques to the problem of integrating local and national level security infrastructures.

This project is aiming to demonstrate a proof of concept system capable of generating configuration information and dynamic updates for local security components on the basis of local and national policies and of user registration information.

One of the concerns in providing a flexible security architecture in response to these needs is the balance to be struck between the different kinds of security mechanism. On the one hand, the authentication of comparatively open access to national services from a mobile and volatile population of staff and students suggests a primarily application-lead set of mechanisms with the minimum of real-time coordination between components, and thinking is therefore focused on certificate-based mechanisms. On the other hand, concerns about campus security and protection against widespread denial of service attacks suggest a continuing need for fine-grain network-level measures, with immediate local control.

It would be highly desirable to be able to combine these two aspects while retaining flexibility and

dynamic configuration. The network level mechanisms, such as firewalls, should be managed dynamically, based on the short-term knowledge of interest in particular applications and involvement of individuals in particular activities; for example, student access to an external resource might be enabled during specific class times, or access to an information server from a remote location might be established as a side-effect of the authorisation decisions in the single sign-on and token granting mechanisms for the site.

4. The main models involved

In the current project we are not concerned with the details of the business logic. Whilst, in an ideal world, we would expect there to be a set of application specific models, the current system assumes a generic access process in which named clients interact with multi-tier business logic.

Details of the authentication structure are provided by a local organization model and user directory. A service directory indicates both local and national resources; permissions are represented by mappings from service specific roles by linking them to identities in the organizational model. This includes generic identity templates for presenters of certificates in known category under recognized authorities.

The most concrete kind of models is concerned with representation of the supporting network configuration, particularly the routes through controlling components, such as firewalls and access controlled routers. This model will be updated by feedback from operational information so as to stay in step with the real world situation (except when there is a need to analyse the consequences of proposed changes – see section 6).

Finally, there is a set of models representing security policies. These models represent the rules for deriving route-specific permissions from user-level authentication events. For example, the authentication of an individual and the request by that individual to exercise some specific access right for a resource should result in a suitable access route to the resource being enabled, in much the same way as initialising a service opens access paths through current object-aware firewall systems in a CORBA environment.

These models are maintained partly in open LDAP directories and partly in specific local repositories, depending on the frequency and range of access requirements. The choice of a suitable form of repository will, in general, depend on the constraints derived from existing management tools and interfaces, such as the need to incorporate information from existing network management or infrastructure components. A practical infrastructure should therefore incorporate adaptation mechanisms to unify a number of different repository mechanisms.

5. Tools and transformations

The main tools required for such a system are, leaving aside administrator interfaces for the more static data models, those concerned with the processing of the set of authenticated entities to create a set of required access paths, and those needed to apply these requirements to the known configuration, and so to derive suitable network access control information and target-service access tokens from the set of access paths. This is a notional separation of functions, rather than a practical guide to modularity, because of the potentially intimidating scale of the intermediate set of access paths.

The step from authentication to path authorisation will involve the interpretation of the service or organization-specific security policies. In practice, there are likely to be resource limitations on the complexity of the access controls that can be expressed in supporting components, so the mapping from the path set to local control expressions will not be exact. For example, a national dataset service which has typically a hundred simultaneous research users could be protected by a low level network filter dealing with individual source network addresses, but this level of granularity could stress router resources. It is certainly likely to do so if there are a thousand, or ten thousand, simultaneous users! It may therefore be expedient to construct a much smaller number of control terms permitting access to, for example, subnets containing one or more users; such a strategy is likely to result in a comparatively small number of control list entries.

Applying this kind of approximation involves an element of risk. It is, for sure, less of a risk than current open access strategies involve, but it is not an exact match to the known authorised user requirements. The transformational tool might also be guided by supporting dynamic trust or threat information to establish how to strike the best balance between security and resource usage.

This approach to coordinating fine-scale application and network security models would be quite

infeasible without a high degree of automation; human security administrators could never come near to the required rate of change of configuration. However, once the process is dynamic, and is based on the processing of explicit models of organizational structure and security policy, it becomes possible for the system to track changes in these models directly. It also becomes possible for integration with other models to be increased, with the models that are representing the business logic being made more security-aware.

6. The evolution of policies

Polices are not static; they are in a continual process of evolution, and the many separate teams may be responsible for the development of different localised or pervasive policies. These teams will all have their own timescales and deadlines. A major campaign to introduce new security features may well proceed in parallel with the introduction of new business processes; any of the activities in the different teams may need to generate testing configurations or to release or rollback new versions without disruption of the other development activities.

This implies that the generating tools need to be able to pick up and act upon contextual information from the developer or administrator invoking the tools so as to determine what combination of versions and what execution environment is needed on a particular occasion. There should also be support for the detection and management of unavoidable dependencies between threads of development, such as the need to make new classes of security information visible within security-aware business logic. For example, different qualifying data may become necessary when a change in tax law introduces new categories of obligation or exemption; the change may come into effect on a particular day, but preparatory steps will have been taken and additional information and validity checking introduced over a period of months beforehand. It may be necessary for people or processes in selected roles to retain access to old policies or simulate application of proposed new policies over a period of months or more. These considerations will make the question of version management in a model-driven architecture extremely important [5].

One may conclude from these requirements that the repository on which a model-driven architecture is based will need to have strong and flexible versioning and scenario planning facilities to support parallel development activities. It will be highly desirable for the toolset to be able to construct test systems based on the selections of versions of the different models that are derived from a user-specific context, with appropriate consistency checking and integration support, as found in a sophisticated version management system.

7. Conclusions

The definition of the model representing security policies is feasible and the work on the demonstrator described above is well advanced; it is expected to be substantially complete before the MAMAD workshop. It has already shown that the model-driven architecture approach is applicable to the management of pervasive infrastructure properties, such as security, as well as to more specific models concerned with business processes.

However, this activity has already helped to identify a number of requirements for a model-driven framework that need to be investigated further, particularly in the support of independent phased development and in the support for evolution within model driven structures, before we really have a mature model-driven software engineering process.

Acknowledgement

The author would like to acknowledge the support of the UK Joint Information Systems Committee for the project “Deriving Authority from Security Policy” described in this paper.

References

[1] ISO/IEC IS 10746-3, Open Distributed Processing Reference Model – Part 3: Architecture, January 1995

[2] Richard Soley, “Model Driven Architecture” Object Management Group White Paper November, 2000

[3] ISO/IEC DIS 15414, Open Distributed Processing – Enterprise Language, 2002.

[4] The JISC Authentication, Authorisation and Accounting (AAA) programme, http://www.jisc.ac.uk/index.cfm?name=programme_aaa

[5] P.F. Linington, “An ODP approach to the development of large middleware systems”, in Proc. DAIS99, June 1999.