

ULRR

Security design patterns in the MASTER workbench

Item Type	Meetings and Proceedings
Authors	Kearney, Paul J.;Sinclair, David;Wagner, Sebastian
Citation	Proceedings of the CyberPatterns 2012;
Download date	2026-04-12 11:55:43
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/2606

Security Design Patterns in the MASTER Workbench

Paul J. Kearney, David A. Sinclair and Sebastian Wagner

Abstract—This paper describes pattern-related aspects of the prototype Protection and Assessment Workbench developed as part of the MASTER EU 7th Framework collaborative research project. The Workbench supports a model-driven design process within the overall MASTER methodology. It includes a Protection and Regulatory Model (PRM) tool that is a step towards turning the Workbench into an ‘organisational memory’ for design practices that accumulates and improves over time. PRMs are essentially control process design patterns that incorporate proven strategies in a re-usable form, saving time and improving quality and consistency.

Index Terms—business process, design pattern, model-driven design, security

I. INTRODUCTION

THIS paper will describe pattern-related aspects of the prototype Protection and Assessment (P&A) Workbench developed as part of the MASTER EU 7th Framework collaborative research project [1]. MASTER ran from 2008-2011 with the aim of providing a methodology and infrastructure that facilitate monitoring, enforcement, and auditing of security compliance, especially where highly dynamic service oriented architectures are used to support business process enactment in single, multi-domain, and iterated contexts.

The P&A Workbench is a prototype graphical software tool that supports a model-driven design process within the overall MASTER methodology. The term ‘Protection & Assessment’ reflects the focus of MASTER on design and execution of processes that enforce security policies and measure their effectiveness. The P&A Workbench is intended for use by an

analyst/designer working in conjunction with a variety of business stakeholders to develop a Design Model. Typically, a Design Model is constructed, evaluated and elaborated iteratively until stakeholders agree that it describes an effective and affordable P&A solution that complies with control objectives. It is then refined further until the description is sufficiently concrete to provide the basis for implementation of a P&A system based on the MASTER infrastructure. Verification and Simulation tools are used to help confirm that the implemented system will indeed be fully compliant with the control objectives. The main output derived from the Design Model is a Policy Document that specifies rules that are interpreted by the run-time P&A system and determine its behaviour.

The Workbench comprises an extensible set of graphical modelling and transformation tools within a loosely coupled architectural framework, and a Repository in which interim and final models and other documents are stored. Collections of documents related to the same project are checked out of the Repository into a local workspace while being worked on using the Workbench tools. Documents in the Repository, notably Policy Documents, may be made available for deployment to, or consultation by, the MASTER infrastructure components at run-time. Not strictly part of the Workbench, but closely related to it, are the MASTER Verification and Simulation tools. These are invoked from the workbench in order to test certain properties of the Design Model.

The Workbench achieved its objectives as a pre-competitive research prototype. It is a significant step towards the vision of a software tool to support a model-driven approach to the design and implementation of security controls and indicators derived from compliance requirements. The core functionality has been implemented, proving the concept and providing a basis for future development as a production tool. It is also suitable for trials and pilot projects for organisations considering adopting the MASTER approach. A production version would need to form part of an integrated suite or else inter-operate with leading modelling, development and management tools either as part of or mixed and matched on a best-of-breed basis. The prototype also provides an excellent platform for continued experimental implementation of advanced features, e.g. concerning knowledge management and re-use. Furthermore, the experience of developing the workbench has been useful in evaluating and building expertise in model-driven design via domain-specific languages (DSLs), and more specifically in use of the DSL-

Extended abstract submitted May 11, 2012. Full position paper submitted 19th June, 2012. This work was supported by EU FP7 project MASTER (Grant Agreement No. 216917)

P. J. Kearney is Chief Security Researcher in the Security Futures Practice, BT Innovate & Design, ppG13 Orion Building, BT Adastral Park, Martlesham Heath, Ipswich, Suffolk IP5 3RE, UK (phone: +44 1473 606266; e-mail paul.3.kearney@bt.com).

D. A. Sinclair is a Senior Researcher with Lero, the Irish Software Engineering Research Centre (supported, in part, by Science Foundation grant 03/CE2/I303_1) and a Senior Lecturer in the School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland (email: david.sinclair@computing.dcu.ie).

S. Wagner is Research Associate and Ph.D. Student at the Institute of Architecture of Application Systems at the University of Stuttgart, Universitaetsstrasse 38, 70569 Stuttgart, Germany (email: Sebastian.wagner@iaas.uni-stuttgart.de)

building tools from the Eclipse Modelling Project.

II. USAGE SCENARIO

A typical usage scenario for the workbench is as follows. A company already has a service-oriented infrastructure that it uses to partially automate the enactment of certain key business processes. It has identified that some high level security policies are relevant for the enactment of these business processes and the resources used. We are not concerned at this point whether the policies are due to regulatory requirements, industry codes of practice, or are of company-internal origin. The company now wishes to deploy IT controls to ensure as far as is practical that the policies are complied with and that the extent of any non-compliance is measured.

The company has decided to use the MASTER infrastructure to implement the controls. Company IT and security staff are now faced with a range of design decisions regarding how to use the ‘vanilla’ MASTER software to create an appropriate set of controls and indicators that interacts correctly with the existing infrastructure. The main means of customising the MASTER run-time infrastructure to implement the controls is to deploy a set of Signalling, Monitoring, Enforcement and Assessment policies to it.

Major requirements on the workbench are:

- to support the application of MASTER Methodology steps that guide the analyst and business stakeholders through description of the business context, selection and refinement of Control Objectives (COs) and Control Activities (CAs), specification of Key Assurance and Key Security Indicators (KAIs and KSIs), and definition of Control Processes (CPs) that implement the CAs.
- to use the Verification and Simulation (V&S) tools to help confirm the correctness of the CPs, i.e. that composing the CPs with the relevant target business process will result in compliance with the COs.
- to facilitate implementation of these design decisions by automating the creation of MASTER Signalling, Monitoring, Enforcement and Assessment policies.

Furthermore, the company wants compliance to be auditable to provide its management, shareholders, customers, partners, and legal and regulatory authorities with confidence that the policies are indeed being followed. This means it must be easy to check that appropriate controls are deployed and that they have been implemented correctly. Here, ‘appropriate’ means that the controls are suitable to achieve the intention of the policies taking into account the anticipated threat environment and the risk appetite and budget of the organisation. The workbench therefore needs, in addition, to enable an independent auditor to review and verify design and deployment decisions to ensure that appropriate choices have been made to implement the high level policies

The company will need to update the controls as requirements and business, technology and threat environments change. Furthermore, it is unlikely that the

company will implement controls for a single process in isolation (except as a pilot), but rather would roll MASTER out for all relevant business processes as part of a major compliance or Business Process Re-engineering initiative. The Workbench therefore needs to support the maintenance of controls over time and ideally, re-use of successful controls and accumulation of experience.

III. SECURITY PATTERNS

Schumacher et al. [2], inspired by the approach to pattern-oriented software architecture taken in [3] give the following definition: “A security pattern describes a particular recurring security problem that arises in specific contexts, and presents a well-proven generic solution for it. The solution consists of a set of interacting roles that can be arranged into multiple concrete design structures, as well as a process to create one particular structure.” Security problems typically deal with constraining the behaviour of systems to uphold confidentiality and integrity properties while continuing to provide a service to legitimate users, in the face of malicious or misguided agents. A security pattern establishes an approach to solving a class of security problems by generalising over related successful cases.

Schumacher et al. regard humans as being the *only* audience for patterns. They further say that it is not practical to make patterns machine-readable and automatable, as it is all but impossible to formalise them. They contrast patterns with “other design or modelling techniques such as the Unified Modelling Language” that result in artefacts that are intended to be readable by machines and humans. This view is not universal, however; for example, Sowa identifies three kinds of pattern (syntax, semantics and pragmatics) and three kinds of notation (natural language, linear notations for logic and computation, and graphical diagrams and movies) in his work applying conceptual graphs to knowledge design patterns [4].

The approach taken in MASTER attempts to bridge the two worlds of patterns and modelling languages. A MASTER pattern (termed a Protection & Regulatory Model, PRM) includes fields for textual / free form entries that are aimed purely at a human audience and are broadly similar to those found in ‘classical’ patterns. However, it also contains corresponding design fragments expressed in the MASTER modelling language.

The intention is that the Workbench will incorporate a library of PRMs. In the course of developing a design model, the analyst and stakeholders will browse this library looking for PRMs that are relevant to the application context and the aspect of the model being worked on. For example, the analyst may be working on a control objective aimed at minimising opportunities for insider fraud and find a pattern describing how to do this using the principle of separation of duties. This stage is similar to searching a conventional pattern system.

The analyst then turns to the model fragments described in the PRM. These are templates with place-holders that must eventually be unified with entities in the model. The template

is instantiated and added to model, and the analyst stitches it into place by associating existing entities with placeholders. Often, the existing model will need to be re-worked to enable a good fit, for example, a role in a business process might have to be divided into two simpler roles to allow separation of duties. Sometimes, the changes required to achieve a fit may be judged excessive, and the pattern will be discarded and a new one sought. Not all the place-holders need to match existing model elements; indeed some may give rise to further searches of the pattern library.

Use of patterns in this way results in improved quality (as a result of adopting proven solutions and best-practice) and also in increased productivity as models are constructed partly from pre-fabricated components rather than always working at the level of the primitives of the modelling language.

Once a project is completed, the model is reviewed with the aim of identifying successful strategies with potential for re-use. These are recast in more abstract form, documented as PRMs and deposited in the library. PRMs that have been used should also be critiqued as part of this review and comments added to the record in the library. If appropriate, a PRM may be modified or elaborated in the light of experience, or even deleted if it has proved unsuccessful on several occasions. In this way, the Workbench will become an ‘organisational memory’ for design practices that accumulates and improves over time.

PRMs are now covered in more detail, before moving on to describe an early prototype PRM tool that was implemented as part of Workbench during the MASTER project.

IV. PROTECTION AND REGULATORY MODELS

Each PRM describes a design pattern that captures “best practice” in designing control processes for a specific control objective in a specific context. The essence of a PRM is a set of parameterised fragments of control processes, and each PRM is linked to a generalised control objective. The control process fragments in each PRM are composed and sequenced in a specific manner. Each control process fragment is defined by a set of allowed event traces. An event trace is a sequence of observable events. These event traces may consume events from or add events to its enclosing environment. Some of the events in the traces may be bound to terms in the PRM’s control objective or the events that are consumed or added to its enclosing environment. These events are defined as *exposed PRM events* and are included in the parameters of the PRM. Non-exposed events, *internal PRM events*, are not visible to the actors who interact with the PRM. Internal PRM events can be renamed without any impact on actors and processes that interact with the PRM. Exposed PRM events are typically the input and output events of the PRM, but may also include intermediate events.

Each PRM contains the following elements.

- 1) A textual description that describes the purposes of the PRM, its associated control objective(s) and the type of environments in which the PRM is envisaged to be deployed. This description should be brief enough to allow a user to browse this description within a collection

of PRMs, yet it should contain sufficient information to allow a user to make a preliminary decision on whether a PRM is appropriate for a particular design.

- 2) A description of the parameters of the PRM.
- 3) A description of the context in which the problem that is being solved arises. In addition to describing the attacks that the PRM is addressing, the section also describes the organisational context in which the PRM can be used. More specifically the PRM describes the requirements and restrictions on the organisational structure in which the PRM can be used in terms of:
 - a) The *entities* participating in the solution: Actor roles, Goals, Activities, and Resources.
 - b) The *structural dependencies* among entities. These structural dependencies can arise due to the context of the problem the PRM is addressing or due to the structure of an organisation. An organisation can impose dependencies on entities and actors. For example, the organisation may require certain entities collaborate together. It may define which actors are subordinate to other actors, and which actors have incompatible roles and hence require a separation of their duties.
- 4) A description of any infrastructural services required by the PRM. For example, an authentication PRM may require a service that associates a set of roles with each user.
- 5) A description of the control objective implemented by the PRM. This is given by a textual description and a formal statement of the control objective using the MASTER Property Specification Language.
- 6) A formal description of the control process fragments that implement the PRM. This description is specified in the M-calculus, which has both a graphical and textual representation. The M-calculus is a process calculus that extends Milner’s π -calculus [5] to include timed events and probabilistic events. The motivation for describing the control process fragments formally is that this enables the PRM designer to prove that the PRM implements its control objective in the specified contexts.
- 7) A description of the possible impact of the PRM on the overall system. This section should address, but not be limited to, issues such as availability, confidentiality, integrity, performance and usability.

When composing multiple PRMs, care needs to be taken that the names of the exposed PRM events do not conflict. Internal PRM events can be renamed to avoid any conflict with events in the target processes, control processes or other PRMs. Since the exposed PRM events are all parameters of the PRM, each PRM is instantiated with events from the target and control processes. If two PRMs reference the same event from the target and/or control processes, this event is called a *conflicted event*. Conflicted events may require special handling, such as duplication to ensure that each PRM can consume the event, or amalgamation to ensure that conflicted events produced by both PRMs do not invalidate the system’s control objectives when consumed by the environment.

Since PRMs represent “best practice” they have an added significance in environment with multiple trust domains. While organisation X may be reluctant to disclose the details of the control processes it is providing to organisation Y , if organisation X agrees to provide a specific authentication PRM then organisation Y can be sure of that the corresponding control objective is achieved (subject to the PRM’s assumptions on its environment) without organisation Y revealing, or allowing access, to the control infrastructure within organisation Y .

The use of PRMs can be illustrated by looking at how one could be used to represent a control taken from ISO 17779-2005. The ISO standard describes Control 11.1.1 “Access Control Policy” as “*An access control policy should be established, documented and reviewed based on the business and security requirement for access*”. The text does not describe how to implement Control 11.1.1, but it does describe some properties that any implementation of the control 11.1.1 must have. The corresponding PRM would capture a “best practice” implementation of this control in specific contexts and outline how the properties specified for this control can be proved in these contexts. The main objective of PRM is to capture precisely and formalise the knowledge required to implement a control process aiming to be compliant with control objective derived from a business objective or regulation. The control objectives describe “patterns” that compliant control processes must match and PRMs formalise these patterns.

V. PRM TOOL

The prototype PRM tool is an Eclipse plugin that is a step towards turning the Workbench into an ‘organisational memory’ for design practices that accumulates and improves over time. As mentioned in the previous section, PRMs are essentially control process design patterns that incorporate proven strategies in a re-usable form, saving time and improving quality and consistency. They map security compliance objectives to control processes and architectures that achieve them in defined business and technical contexts. The tool allows new PRMs to be created and stored in the Repository, and retrieved and instantiated as part of a new model when required.

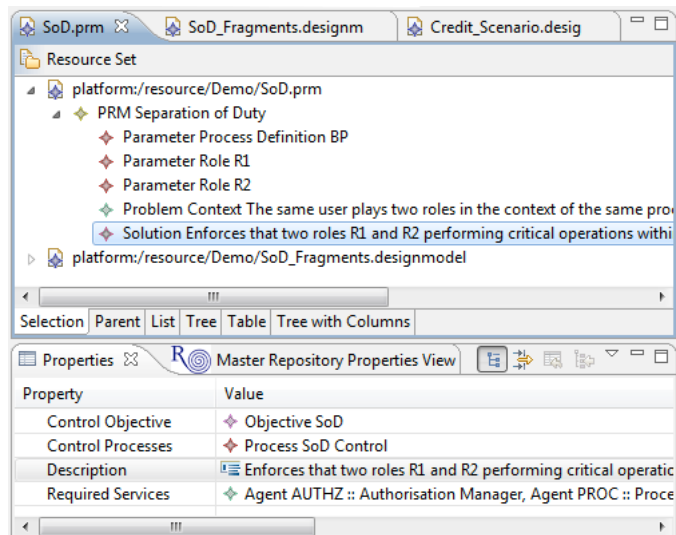


Figure 1 PRM Tool with SoD PRM

During its initial creation the PRM elements and properties described in Section IV (name, description, problem context, parameters etc.) can be defined with the PRM tool. The solution of a PRM including its control objectives and the implementation artefacts, i.e. control processes and the corresponding infrastructure services, can be directly linked to elements modelled within the design workbench. Hence, if they are modified with their corresponding workbench tools (e.g. a process model editor) these modifications are immediately visible within the PRM. Moreover, this has the advantage that these elements can be also used in other PRMs. The PRM in Figure 1 describes a Separation of Duty (SoD) control pattern that ensures that two roles $R1$ and $R2$ performing critical operations within a business process BP are played by two separate agents. The PRM is implemented by a corresponding control process that enforces that the control objective is met. The control process depends on the infrastructural service $AUTHZ$ that acts as authorization manager, and $PROC$ that holds the descriptions of the process instances of BP the control process is applied to. The services and the control process for this PRM are defined in the design model $SoD_Fragments$ and can also be used by other PRMs. The business process and the roles the control process is applied to are use-case dependent. Hence, these elements are parameterized. To achieve that, parameters can be defined on a PRM with the PRM tool. In our scenario there are three parameters: BP , $R1$ and $R2$. Each parameter represents an abstract element to be replaced by a concrete element that provides domain-specific information during the instantiation of a PRM in order to make the implementation artefacts executable. For instance the parameter BP in Figure 1 refers to an abstract *Process* element in the design model $SoD_Fragments$. During instantiation of the SoD PRM the abstract *Process* element and the *Role* elements have to be replaced by a concrete implementation these elements.

The PRM tool provides a wizard that guides the user through the instantiation of a PRM. As shown in Figure 2, the wizard lists all parameters of a PRM and the user can provide

concrete values for the listed parameters. More precisely, this means that she can supply concrete design model elements from arbitrary design models from the workbench or repository, respectively. In order to do that for each parameter she provides the path to the design model that contains the element that has to serve as parameter value and selects the design model element from the combo box. Our SoD PRM has to be applied to a loan approval process. Hence, as shown in Figure 2 the *BP* parameter is instantiated with a *Credit Process*. The combo boxes list only elements of a design model that match the type of the parameter defined by the abstract element. Therefore, to instantiate the first role *RI*, only those elements of the selected design model that have the type *Role* are listed in the combo box. Figure 2 lists, in this case *Auditor*, *Clerk* and *Manager*.

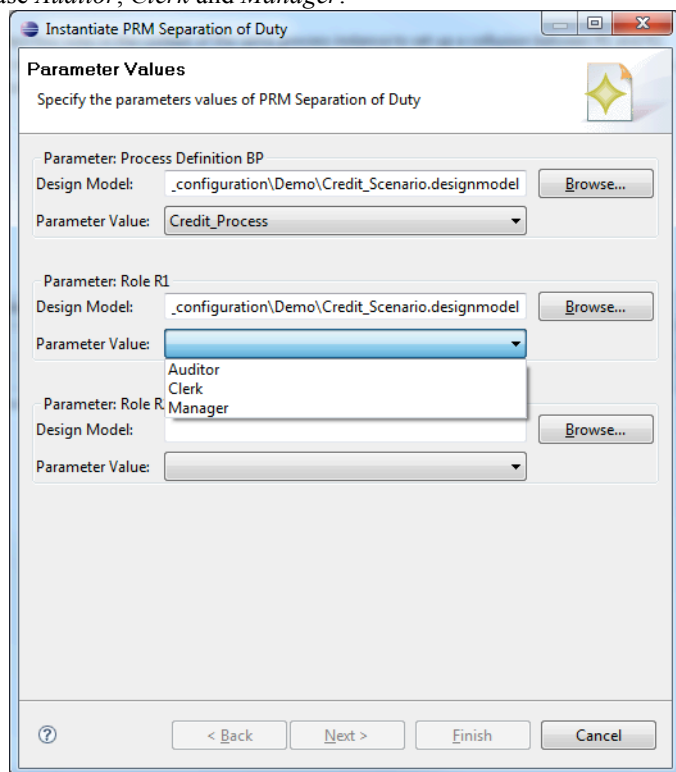


Figure 2 PRM Instantiation Wizard

After the user has provided the values for each parameter she finishes the wizard and the PRM is instantiated by creating a new design model out of it that is stored in the workbench. In our example a design model is created that contains the Credit Process and the SoD control process that can be performed for the specified roles along with the infrastructure services.

VI. FUTURE DIRECTIONS

The current prototype provides an excellent platform for continued experimental implementation of aspects of the Workbench vision that we did not have time to investigate or implement fully during the MASTER project. Re-use of parameterised models in design patterns is already present in the workbench in the form of the PRM tool. While this is effective in proving and demonstrating a viable approach there

are many details to be fleshed out. Once a framework for re-use has been established, PRM libraries need to be developed providing abstract solutions to compliance requirement commonly appearing in regulations and corporate policies.

PRMs are only one form of knowledge management that would be useful in a Workbench. The design process makes extensive use of knowledge derived from experience. Traditionally, this has been held in the heads of experts, and in text books, codes of practice and standards. A future research project could incorporate an expert system into the Workbench to provide advice on design decisions. The knowledge base of the expert system could be seeded with generic guidance, and expanded over time with enterprise-specific and sector-specific knowledge to become a living means of continuous improvement. An important aspect of the project would be investigation of means of capturing new knowledge and otherwise maintaining the knowledge base.

REFERENCES

- [1] MASTER project public web site: <http://www.master-fp7.eu/>
- [2] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*, John Wiley & Sons, 2005
- [3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture – A System of Patterns*, John Wiley & Sons, 1996
- [4] J. F. Sowa, *Knowledge Design Patterns – Combining logic, ontology and computation*, <http://www.jfsowa.com/talks/kdptut.pdf>, 2012
- [5] Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge Univ. Press, 1999.