Journal of Object Technology
Published by ETH Zurich, Chair of Software Engineering, © JOT 2010
Online at http://www.jot.fm.

Using a Situational Method Engineering Approach to Identify Reusable Method Fragments from the Secure TROPOS Methodology

Graham Low[a] Haralambos Mouratidis[b]
Brian Henderson-Sellers[c]

a. University of New South Wales
b. University of East London
c. University of Technology, Sydney

Abstract Situational method engineering (SME) has as a focus a repository of method fragments, gleaned from extant methodologies and best practice. Using one such example, the OPF (OPEN Process Framework) repository, we identify deficiencies in the current SME support for security-related issues in the context of agent-oriented software engineering. Specifically, theoretical proposals for the development of reusable security-related method fragments from the agent-oriented methodology Secure Tropos are discussed. Since the OPF repository has already been enhanced by fragments from Tropos and other non-security-focussed agent-oriented software development methodologies, the only method fragments from Secure Tropos not already contained in this repository are those that are specifically security-related. These are identified, clearly defined and recommended for inclusion in the current OPF repository of method fragments.

Keywords Secure software engineering, Secure Tropos, Situational method engineering, Method fragments.

1 Introduction: Acquisition of New Method Fragments

It is well recognized, within the Agent-Oriented Software Engineering (AOSE) community, that appropriate methodologies are needed for the development of multi-agent systems e.g. [13, 44, 53]. These need to have a quasi-formal underpinning in terms of, say, a metamodel, ontologies and grammars, and be developed with agent-oriented concepts such as goals, plans and capabilities in mind. Towards this direction, a large

Graham Low, Haralambos Mouratidis, Brian Henderson-Sellers. *Using a Situational Method Engineering Approach to Identify Reusable Method Fragments from the Secure TROPOS Methodology*. In Journal of Object Technology, vol. 9, no. 4, 2010, pages 91–125. Available at http://www.jot.fm/contents/issue_2010_07/article5.html

number of metamodels and ontological foundations have been developed to support the description of AOSE methodologies (see for instance various collected chapters in [25]). A software development methodology may be packaged as a single entity, often "branded" (i.e. given a name) for commercial reasons. Alternatively, it could be constructed from pieces: "method fragments" — as proposed in the approach of situational method engineering (SME) — the subset of method engineering that deals specifically with the merging of situational constraints and method construction techniques [28].

Situational method engineering (SME) [9, 35, 47, 49, 51, 56] is a subdiscipline of IT that focusses on the creation of software development methodologies from method fragments — methodologies that are applicable to one highly specific situation (Figure 1). This is in contrast to the more "traditional" (in the literature) approach of attempting to create a one-size-fits all, hard-wired methodological package for industry usage. SME focuses on how to identify and document fragments from existing sources, including best practice in industry; how to store these method fragments and ensure their quality; and how to construct a method(ology)[1] for a specific situation from these pieces (method fragments). Since each situational method uses method fragments from a single repository, a.k.a. methodbase [9], it follows that the method fragments can then be used in more than one methodology construction effort [29] and thus fulfil the criterion of methodological reuse [47].

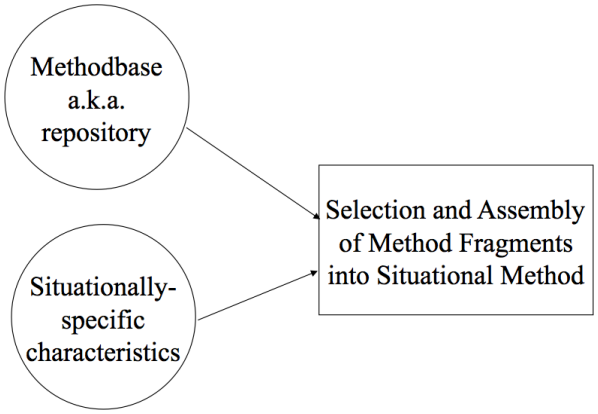


Figure 1 – In SME, fragments are retrieved from the methodbase (or repository) and combined with project characteristics in the creation of a situationally specific method.

A common means of obtaining method fragments is to decompose existing methodologies [22, 48]. The unit of decomposition is specified by a metamodel that defines, for example, all the characteristics to formally describe a fragment, such as a task, a technique, a role etc. That is, each fragment is conformant to an element in the methodology metamodel. Consequently, new fragments are compatible, at least in form, to pre-existing fragments in the method repository. Metamodels have strong connections to ontologies e.g. [21] and provide the abstract syntax for a method description language. Since each meta-element provides a clear definition for each kind of method fragment, compatibility is further encouraged in terms of granularity (of the generated method fragment), pre- and post-conditions for each fragment and application guidelines, the last giving information regarding the methodological source of the fragment and hence its underlying rationale or mindset. However, fragment

[1] We treat "method" and "methodology" as synonyms in this paper (see [34]).

contents can never be guaranteed as being compatible — indeed, such compatibility is not necessarily desirable since a method repository contains method fragments relevant to various paradigms and approaches e.g. agent-oriented, object-oriented, ER-focussed. It is therefore at the construction phase (for a new method) that these paradigmatic consistencies must be ensured — probably the most difficult challenge within the SME approach at present. Combined, these attributes of a metamodel-underpinned SME approach, together with practical experience e.g. [3, 30, 57], assist the method engineer in creating a complete and consistent methodology (see also [10]). However, despite its strong theoretical underpinning, SME has had little penetration to date in industry. This is probably because its biggest drawback is perceived to be the difficulty and expense of method construction — especially in comparison to purchasing an off-the-shelf, shrink-wrapped methodology. What is typically forgotten is the potentially large cost of customizing (usually by deletion) such a comprehensive methodology to the more meagre requirements of a particular situational context.

To date, SME experiences have been largely in the object-oriented, business information systems area (see, for example, [29]). We believe it is equally important to apply the SME ideas to agent-oriented development and to identify and document fragments from existing AOSE sources. Initial work in this area (see Chapter 13 in [25]) considered the identification and documentation of fragments from a number of AOSE methodologies. In that work, to guide the decomposition, an existing metamodel-underpinned repository of method fragments (OPEN Process Framework (OPF) [17]) was utilized. Nevertheless, this work did not consider an important aspect of multi-agent systems, that of security. As argued in recent research [41] in the agent-oriented software engineering community *"security plays an important role in the development of multi-agent systems and is considered as one of the main issues to be dealt for agent technology to be widely used outside the research community"*. It is also recognized by recent research [16, 38] that security should be considered from the early stages of the development process and security requirements should be defined in parallel with the system's requirements specification. Considering security and functional requirements together throughout the development stages helps to limit the impact of conflict between them by identifying it early in the development process. Contrariwise, adding security concerns as an afterthought increases the chance of conflict. A solution to this kind of problem requires an in-depth study of the system, its organization and its properties. Thus, a considerable amount of money and valuable time is needed. It is therefore important that a repository of method fragments to support the development of multi-agent systems cannot be considered as complete without the inclusion of security-related method fragments.

In this paper, we synergistically combine the SE sub-disciplines of agents, SME and security and focus on the identification for inclusion in the OPF repository of security-related method fragments from an existing AOSE source, that source being the Secure Tropos agent-oriented methodology [37, 42]. An important feature of the chosen Secure Tropos methodology is that security requirements of the system under development can be traced back to early requirements. Therefore, the developers can understand why specific security-related requirements need to be introduced to the system as well as their impact on other functional requirements of the system. This provides a well structured approach to developing secure multi-agent systems where security is considered from the early stages of the development process.

In Section 2, we give a brief overview of both the OPF and Secure Tropos. In Section 3, we identify method fragments from Secure Tropos that have no counterpart

in the current contents of the OPF repository and are therefore proposed for addition to this repository. Section 4 discusses related work and Section 5 concludes the paper.

2 Brief Overviews of OPF and Secure TROPOS

2.1 OPF

We propose using an existing SME-based framework for both the metamodel and storage capabilities. OPEN (Object-oriented Process, Environment and Notation) [17] is just such an established approach for developing software within the method engineering paradigm. Within the OPEN approach, the most relevant element is the OPF (Figure 2), which comprises a metamodel, recently updated to be conformant with ISO/IEC 24744 [32], that defines all the methodology elements at a high level of abstraction plus a repository that contains instances of those metalevel concepts (i.e. method fragments) supplemented by a set of construction guidelines.

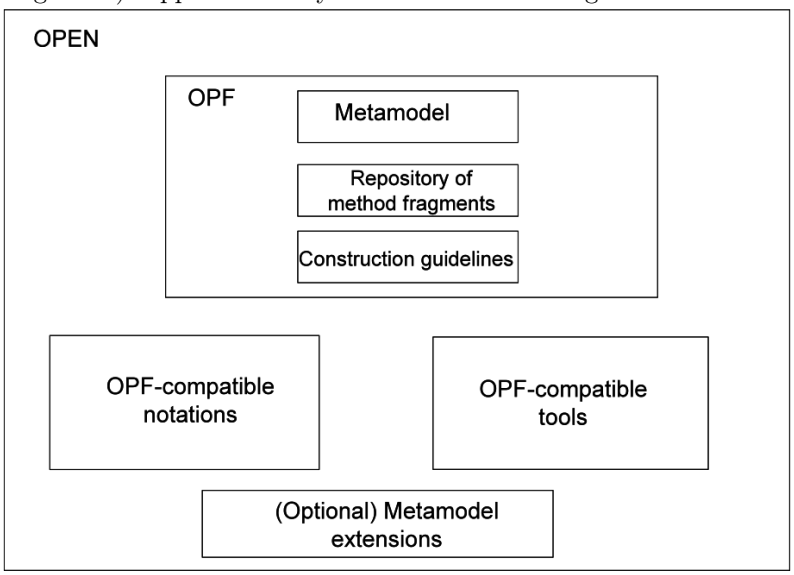


Figure 2 – OPEN consists not only of the OPEN Process Framework but also OPF-compatibles notations and tools as well as potential extensions to the metamodel.

Each element in the OPF repository is a method fragment conformant to its corresponding definitional element in the metamodel. The major elements in the OPF metamodel are Work Units (with subtypes of Process, Task and Technique), Work Product and Producer — see Figure 3. A Task is defined as being the smallest unit of work that can be managed in a project (this definition additionally being in agreement with the terminology of the Project Managers' Body of Knowledge). Often it is useful to group together several Tasks into a Process[2] (previously called an Activity in the OPF), which provides a large scale definition of what must be done. Processes are not used for project management or enactment because they are at too high an abstraction level but are useful for giving a broad view of the overall methodology. A technique is a procedure used to accomplish a specific task.

[2]Here using ISO/IEC 24744 terminology.

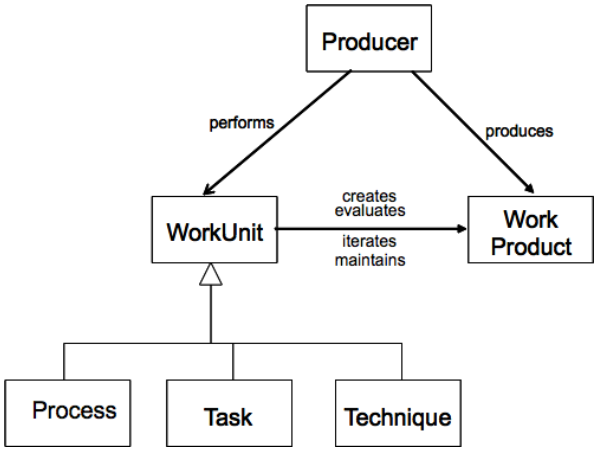


Figure 3 – The three major metatypes in the OPF metamodel together with the three major subtypes of WorkUnit

As seen in Figure 3, these components interact; for example, producers perform work units, work units maintain work products and producers produce work products. Tasks and work products are linked via actions which have constraints associated with them. The Constraint metaclass has subtypes of PreCondition and PostCondition (Figure 4). These two constraint types are thus available for enforcing consistency and completeness in the constructed methodology. (Other pairs of instances of element types can be similarly linked providing integration to the otherwise independent method fragments in the repository.)

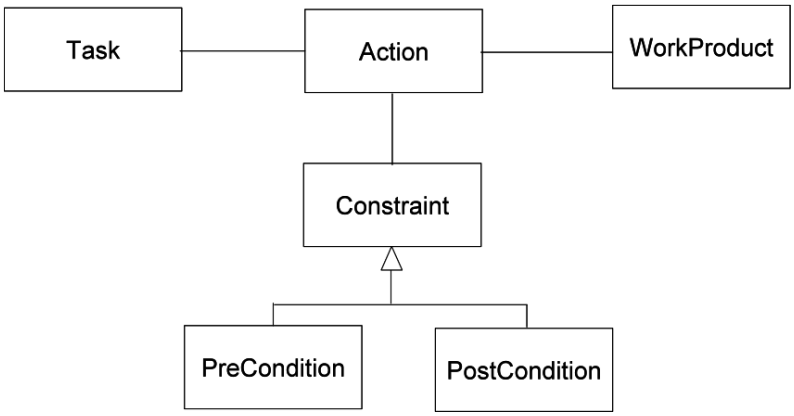


Figure 4 – Actions link Tasks and Work products. Actions have associated Constraints, of two types.

Each of the elements in the metamodel has attributes that determine the fields that need to be completed for each method fragment (see Figure 5 for one such example). As noted earlier, the information in each fragment may be obtained from decomposition of an existing methodology — it could also be created "bottom up" by making a new fragment conformant to the metamodel element, as in Figure 5, but where the information comes not from a single pre-existing methodology but from a

composite of best practice ideas.

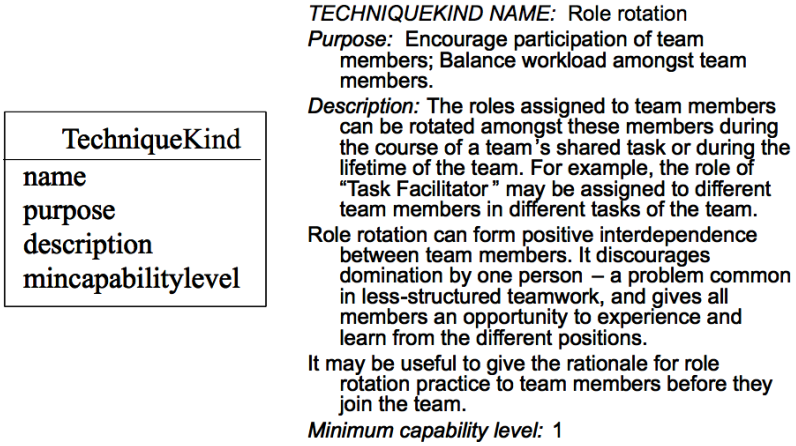


Figure 5 – (a) metamodel for Task and (b) exemplar task fragment conforming to metamodel.

Situational method engineering is then applied in the sense of identification of appropriate method fragments from the repository, within a given situational context (Figure 1), followed by their assembly into a full methodology. Using guidelines, such as ensuring that all output work products (except the deliverable code) are used elsewhere in the constructed method as inputs to some other task, and taking into account all local variables such as skills level, organizational capability level and security levels, a usable and quality methodology can be constructed for application on a specific project or organization i.e. situational method engineering [11].

2.2 Tropos

In this section, we first describe the base methodology of Tropos e.g. [5] and then discuss the extensions made in the published literature that have created Secure Tropos [42]. The Tropos methodology [5, 6, 12, 20, 46] was designed to support agent-oriented systems development, with a particular emphasis on the early requirements engineering phase. Recently, it has been recast in the MDA context [45]. In particular, Tropos aims to:

- raise the conceptual level of Requirements Engineering techniques, so that formal and semiformal languages and representations can be used from the very early stages of requirements elicitation and analysis.
- provide and support the system architecture and definition of the system functionality with a set of "social-oriented" notions —to be used alongside the traditional system-oriented concepts— that allows for an easier mapping of the requirements provided in terms of social and organizational needs —as provided by Requirements Engineering— into the characteristics (functional, architectural, and design-oriented) of the system-to-be.

Tropos aims to fulfil these aims by adopting two specific strategies. First of all, it pays attention to the activities that precede the specification of the prescriptive requirements, such as understanding how and why the intended system can meet

the organizational goals (*Late Requirements Analysis*). Even before this phase, it is important to understand and analyse the organizational goals themselves (*Early Requirements Analysis*). In this, Tropos is largely inspired by the Eric Yu's i* framework for requirements engineering, which offers actors, goals and actor dependencies as primitive concepts. The i* framework has been presented in detail by Yu [59] and has been related to different application areas, including requirements engineering, business process reengineering and software processes. Secondly, Tropos uses AI-derived mentalistic notions such as *actors* (or *agents*), *goals*, *soft-goals*, *plans*, *resources* and *intentional dependencies* in all the phases of software development, from the first phases of early analysis down to the actual implementation. Tropos also includes descriptions of Work Products and several Techniques such as Means-End Analysis, useful in requirements engineering [25].

One of the main advantages of the Tropos methodology is that it captures not only the *what* and the *how*, but also *why* a piece of software is being developed. This, in turn, allows for a more refined analysis of the system dependencies and, in particular, for a much better and uniform treatment not only of the system functional requirements, but also of its non-functional requirements.

The Tropos methodology is mainly based on four phases [5]:

Early Requirements Analysis aims at defining and understanding a problem by studying its existing organizational setting.

Late Requirements Analysis describes the system-to be, in the context of its operational environment.

Architectural Design deals with the definition of the system global architecture in terms of subsystems.

Detailed Design specifies each architectural component in further detail, in terms of inputs, outputs, control and other relevant information.

A crucial role is given to the early analysis of requirements that precedes prescriptive requirements specifications. In particular, as noted above, aside from the understanding of *how* the intended system will fit into the organizational setting, and *what* the system requirements are, Tropos also addresses the analysis of *why* the system requirements are as they are, by performing an in-depth justification with respect to the organizational goals.

Thus, the stakeholder intentions are modelled as goals which, through a goal-oriented analysis, eventually lead to the functional and non-functional requirements of the system-to-be. In Tropos, early requirements are assumed to involve social actors who depend on each other for goals to be achieved, tasks to be performed and resources to be furnished. Tropos includes *actor diagrams* for describing the network of social dependency relationships among actors, as well as *goal diagrams* for analysing goals through a means-ends analysis in order to discover ways of fulfilling them. These primitives have been formalized using intentional concepts from AI, such as goal, belief, ability and commitment [50].

2.3 Secure Tropos: security-related concepts

The basic Tropos methodology was not conceived with security in mind and, as a result, it therefore fails to adequately capture security requirements [40]. In particular, the methodology demonstrates limitations with respect to security in its process as well as in its concepts. For example, the process of integrating security and functional requirements throughout the whole range of the development stages is quite ad hoc; the concept of a soft goal used in Tropos to capture security requirements fails to

adequately capture some of the associated security constraints [37, 42]; and the lack of definition of the Tropos concepts with security in mind makes the distinction between security and other requirements difficult. Therefore, the basic Tropos methodology was enhanced to better model security during the development process of a multi-agent system [41] — albeit the resultant Secure Tropos is a single methodological entity not espousing SME concepts. The Secure Tropos extension introduces two types of extensions to the Tropos methodology: extensions related to the concepts of the Tropos methodology and extensions related to the development process of the methodology. As part of the first type of extension, the concept of constraint was introduced and extended with respect to security and the Tropos concepts of dependency, goal, task, resource and capability were extended with security in mind. With respect to the second type of extension, Secure Tropos introduces the notions of a security analysis relevant, firstly, to the system environment and, secondly, to the system itself. The impact of these security issues on design is also a process-focussed addition. At a low level of process detail, Secure Tropos introduces into Tropos tasks relevant to the detailed, security-focussed analysis of actors, dependencies and references, to any necessary delegation and internal consistency checking as well as impacts on architectural style and how to deal with potential attacks. However, implementation of these Tasks by means of Techniques is not explicitly defined (see further discussion in Section 3.4).

The next sub-sections aim to describe briefly these extensions, which are candidates for new method fragments and/or enhancements to pre-existing method fragments.

2.3.1 Constraint and security constraint

Constraints can represent a set of restrictions that do not permit specific actions to be taken or prevent certain objectives from being achieved and more often [55] are integrated in the specification of existing textual descriptions. Because of its importance in the system development, the concept of constraint has been introduced to the Tropos methodology as a separate concept and the metamodel of the Tropos modelling language has been extended by introducing the construct for modelling constraints [37].

A security constraint is captured through a specialization of constraint, defined as a restriction related to security issues, such as privacy, integrity and availability, that can influence the analysis and design of a multiagent system under development by restricting some alternative design solutions, by conflicting with some of the requirements of the system or by refining some of the system's objectives. It is worth mentioning that there are no specific techniques to identify when developers should stop searching for security constraints; this depends on developer and stakeholder satisfaction that all the appropriate security constraints are modelled in the developed models. Nevertheless, the methodology provides information and a number of additional models, such as the security reference diagram model (see later — Figure 17), to assist developers and stakeholders in identifying the appropriate security constraints. Security constraints do not represent specific security protocol restrictions, which should be not be specified until the implementation of the system. However, they do contribute to a higher level of abstraction that allows for a generalized design free of models biased to particular implementation languages.

2.3.2 Secure dependency

A secure dependency [37] introduces security constraint(s) that must be fulfilled for the dependency to be satisfied. Graphically, security constraints are modelled, using i* notation, as illustrated in Figure 6; as clouds within which the description of the (security) constraint is shown. Both the depender and the dependee must agree for the fulfilment of the security constraint in order for the secure dependency to be valid. That means the depender expects that the dependee will satisfy the security constraint(s) and also that the dependee will make an effort to deliver the dependum by satisfying the same security constraint(s). Secure Tropos defines three different types of secure dependency. In a *depender secure dependency* (see Figure 6-a), the depender depends on the dependee and the depender introduces security constraint(s) for the dependency. In a *dependee secure dependency* (see Figure 6-b), the depender depends on the dependee and the dependee introduces security constraint(s) for the dependency. In a *double secure dependency*, the depender depends on the dependee and both the depender and the dependee introduce security constraints for the dependency. Both must satisfy the security constraints introduced to achieve the secure dependency (see Figure 6-c).

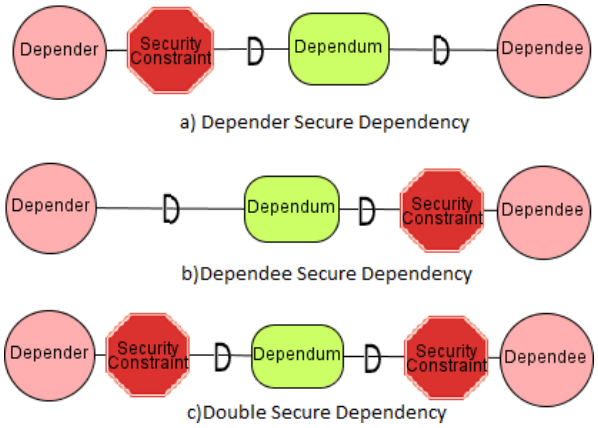
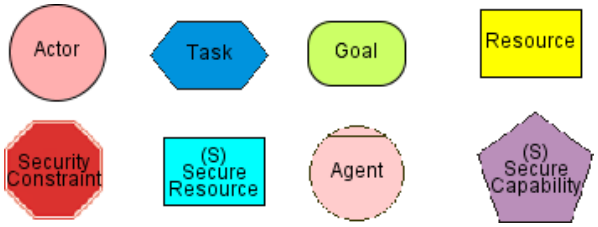


Figure 6 – Secure dependencies.



Legend. Note: This legend is also applicable to Figures 12, 13, 16

2.3.3 Secure entities

The term secure entity is used in Secure Tropos to represent a secure goal, a secure task or a secure resource. A secure goal represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced in order to achieve possible security constraints that are imposed on an actor or that otherwise exist in the system. However, a secure goal does not specifically define how the security constraints can be achieved, since alternatives can also be considered. The precise definition of how the

secure goal can be achieved is given by a secure task. A secure task is defined as a task that represents a particular way of satisfying a secure goal. A secure resource can be defined as an informational entity that is related to the security of the multi-agent system. A secure capability represents the ability of an actor/agent to achieve a secure goal, carry out a secure task and/or deliver a secure resource. The graphical representation of the Tropos entities has been extended to permit modelling of secure entities.

2.3.4 Secure Tropos: security modelling processes (activities)

There are three main aims when considering security issues throughout the development stages of a multiagent system: firstly to identify the security requirements of the system; secondly to develop a design that meets the specified security requirements; and thirdly to validate the developed system with respect to security. With the above in mind, the security-oriented process in Secure Tropos is one of identifying the security requirements of the multiagent system, transforming these requirements to a design that satisfies them and validating the developed system with respect to security.

The first step (which takes place during the early and late requirements) in the proposed security-oriented process aims to identify the security requirements of the system. Security requirements are identified by employing modelling processes such as security constraints, secure entities and secure capabilities modelling. In particular, the security constraints imposed on the system and the stakeholders are identified and secure entities, which guarantee the satisfaction of the identified security constraints, are imposed on the actors of the system.

The second step in the process (during architectural and detailed design) consists of identifying a design that satisfies the security requirements of the system, as well as its functional requirements. To achieve this, agents are identified with the aid of the Tropos modelling techniques and then secure capabilities that guarantee the satisfaction of the security entities identified during the previous step are allocated to the agents. It is worth mentioning that, in this stage, different architectural styles might be used to satisfy the functional requirements of the system. However, there should be an evaluation of how each of these architectural styles satisfies the security requirements of the system. Although, in general, this is left to the developers, a process that is based on the measure of satisfiability [19] can be employed to determine whether, for example, a mobile agent or a client server architecture is more likely to satisfy the security requirements of the system under development.

The third step of the process is the validation of the developed solution. The Secure Tropos process allows for two types of validation: a model validation and design validation. The model validation involves the validation of the developed models (for example, the goal diagram or the actor diagram) with the aid of a set of validation rules [37]. It is worth mentioning that the validation rules are divided into two different categories: the inner-model rules and the outer-model rules. The first allows the validation of each model individually, whereas the second allows the validation of the consistency between the different developed models. The inner-model rules allow developers to validate the relationships between the components of the different security-related models, such as the relationship between the security features and the threats in the security reference diagram; to validate the consistency between the same components that appear in more than one model, such as a security constraint that appears in the actors' model as well as in the goal model; and to validate the consistency when delegation of components between actors takes place.

The design validation aims to check the developed solution against the security policy of the system. A key feature of Secure Tropos that allows us to perform such a validation is the fact that the same secure concepts are used throughout the various development stages. Moreover, the definition of these concepts allows us to provide a direct mapping between them, and therefore to be able to validate whether the proposed security solution satisfies the security policy.

3 Method Fragments in Secure TROPOS

In this section, we analyse Secure Tropos by decomposing it (as an existing methodology) into method fragments for process (cycles, phases), work units (processes a.k.a. activities, tasks and techniques) and work products (models and diagrams)[3]. Each of these is first identified from Secure Tropos. For each methodology fragment kind, we analyse the pre-existing Secure Tropos textual descriptions in the context of the fields supplied directly from the attributes of the relevant metamodel element (see example in Figure 7); although there is some subjectivity in delineating fragments, the metamodel attributes impose clear constraints on both content and format. We then evaluate whether the pre-existing support in the OPF repository is adequate or whether the identified method fragment needs to be added. It is highly unlikely that these additional fragments will require any modification to the metamodel, since the elements in that are at a high conceptual level [33].

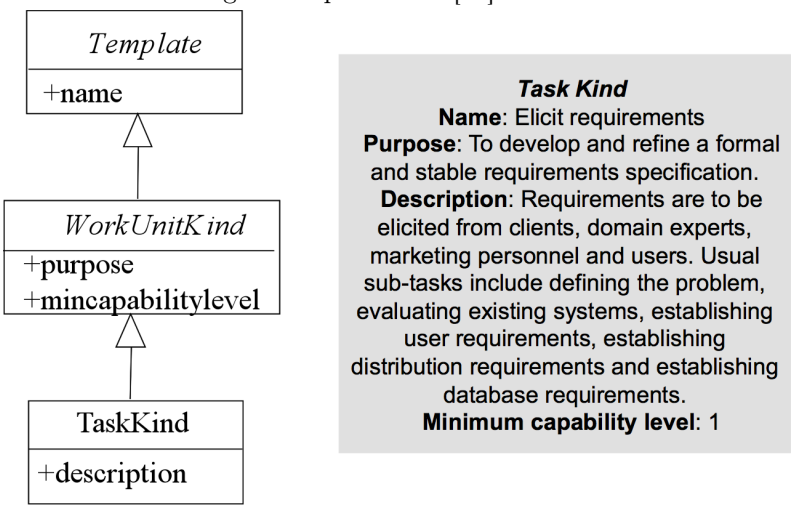


Figure 7 – Metamodel fragment showing attributes (left hand side) which are then given values in the fragment (right hand side) that has been generated from these metamodel elements.

It should be noted that all Work Units (Processes, Tasks and Techniques) have a field to express the minimum capability at which this work unit is appropriate. This is in accordance not only with the metamodel but also the tenets of ISO15504. However, at the time of writing, ISO 15504 does not support security issues so we have omitted this field in the descriptions below. It is likely that the value, once determined, will be at minimum capability level of at least two. We have agreed to contribute and

[3]In this section, for ease of reading we omit the "-kind" suffix on fragment names.

participate in a proposed revision to ISO 15504 in the near future in the determination of appropriate minimum capability levels for such security-related fragments. In terms of space limitations, we list primarily those new fragments and do not list or discuss those already present in the OPF repository (with a few important exceptions).

3.1 Fragments for lifecycle elements

Secure Tropos adopts the Tropos methodology's iterative and refinement lifecycle where a model of the system is incrementally refined and extended from a conceptual level to executable artefacts. This is already fully supported in the OPF repository and, consequently, this research identifies no new fragments in this category.

3.2 Fragments for processes

The metamodel fragment for ProcessKind is shown in Figure 8. However, it should be noted that, typically, the field of minimum capability level is omitted.

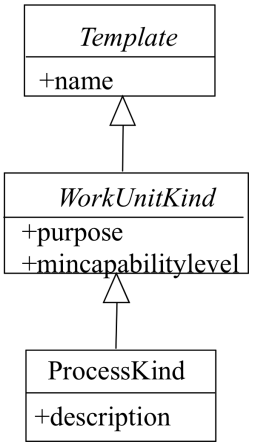


Figure 8 – Metamodel fragment for ProcessKind.

Three additional process fragments are required to support Secure Tropos: *security analysis of system environment*, *security analysis of the system* and *secure system design*. During the first process, the impact with respect to security that the environment has on the system-to-be is analysed. During the second process, the system is analysed with respect to security issues and its security requirements are defined. During the third process, an architecture of the system is defined that satisfies the security requirements of the system. Moreover, the proposed design is tested against a number of attack scenarios to identify whether the components of the system have enough capabilities to withstand such attacks. The definitions of these three fragments are as follows:

Name Security analysis of system environment

Purpose To analyse the impact, with respect to security, that the environment has on the system under development.

Description The environment in which the system under development will be situated is analysed in terms of stakeholders, their intentions and any associated security concerns.

Name Security analysis of the system

Purpose To analyse the system, with respect to security issues, and to define the system's security requirements

Description A thorough analysis of the system (and its internal and external components) takes place and the system's security requirements are identified. (Typically, this process utilizes work products created during the security analysis of the system environment.)

Name Secure system design

Purpose To define the architecture of the system according to its security requirements.

Description The architecture of the system is defined and tested against a number of attack scenarios to identify whether the components of the system have enough "capabilities" to withstand such attacks.

3.3 Fragments for tasks

Each task fragment gleaned from Secure Tropos is ensured to be conformant to the OPF metamodel element called Task (strictly TaskKind: Figure 9), so that consistency of form is achieved both between new fragments and with existing OPF method fragments in the repository. Source information and any information to help the method engineer in the method construction are stored in the method fragment by virtue of the metaclasses of Source and Reference (Figure 10). As there is no current support in the OPF repository for any of the twelve tasks gleaned from Secure Tropos, we propose their addition to the repository. Their definitions are listed below:

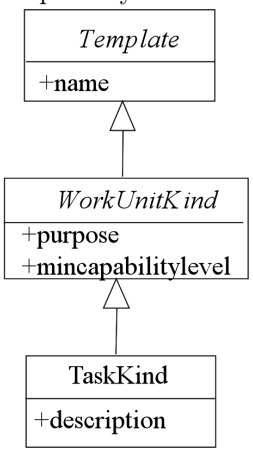


Figure 9 – Metamodel fragment for TaskKind.

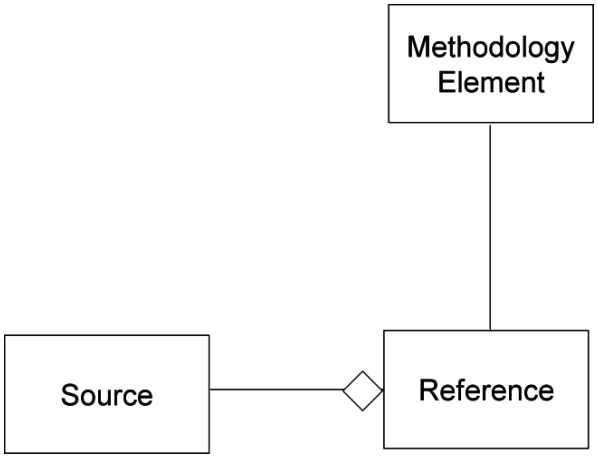


Figure 10 – For each element in the methodology, a reference can be associated, together with details of the source

Name Analyse actor security

Purpose To consider the security of each actor in terms of security constraints and secure entities.

Description This task enables developers to analyse in detail any abstract security constraints documented in the security-enhanced goal diagram as shown in Figure 19. This, in turn, allows an in-depth understanding of the implications of each security constraint on the actor(s), usually resulting in the identification of new security constraints and secure entities.

Name Analyse actor security balance

Purpose To analyse the actors in order to identify any potential bottlenecks with respect to security. Description During this task, critical actors (with respect to security) are identified and a thorough analysis takes place to identify whether actors are overloaded and in danger of not satisfying the security constraints and goals that are assigned to them. This is documented in the security-enhanced goal diagram as shown in Figure 19.

Name Analyse system security

Purpose To analyse in depth the security of the system.

Description This task enables developers to analyse in detail abstract security constraints that have been documented in the system security-enhanced goal diagram. This, in turn, allows an in-depth understanding of the implications of each security constraint on the system, which usually results in identifying new security constraints and secure entities.

Name Analyse system security balance

Purpose To analyse the system in order to identify any potential bottlenecks with respect to security.

Description During this task, the system is analysed (with respect to security) in order to identify if the system is overloaded and if it is in danger of not satisfying the security constraints and goals that are assigned to it.

Name Consider secure dependencies

Purpose To introduce all security constraints that must be fulfilled for the dependency to be satisfied.

Description Both the depender and the dependee must agree for the fulfilment of the security constraint in order for the secure dependency to be valid. That means that the depender expects the dependee to satisfy the security constraint(s) and also that the dependee will make an effort to deliver the dependum by satisfying the security constraint(s). The secure dependencies are documented in the security-enhanced actor diagram as shown in Figure 18.

Name Delegate security constraints

Purpose To delegate security constraints to the system, in order to allow the definition of its security requirements.

Description During the early requirements analysis, a number of security constraints are identified and imposed on the actors. The actors are then responsible for the satisfaction of these security constraints. During the late requirements analysis, a number of these security constraints are delegated, from the actors, to the system. This is mainly the case when an actor cannot satisfy the security constraint (or the corresponding secure dependency restricted by the security constraint) on their own and they depend on the system to assist them in satisfying the security constraint. Therefore, the system assumes responsibility for satisfying them.

Name Develop a secure system structure

Purpose To consider the security of the overall system structure.

Description During this task the developer reasons, with respect to security, about the overall system structure. In particular, the system is decomposed and actors are provided with secure capabilities to allow them to satisfy the secure goals of the system. The main aim is to make sure that the system structure supports all the security requirements. Typically, secure goals, tasks and resources are analysed with the aid of three possible Techniques: means-end analysis, contribution analysis and AND/OR decomposition. In particular, means-end analysis aims at identifying secure tasks and resources that provide a means for achieving a secure goal. Contribution analysis permits developers to identify secure goals that contribute positively or negatively to the secure goal being analysed and AND/OR decomposition provides a division of a secure goal and/or task into sub-goals and sub-tasks respectively.

Name Ensure consistency of security process

Purpose To identify potential inconsistencies and to refine the system in such a way that these inconsistencies are corrected.

Description It is important that the security process employed is in accord with the consistency rules defined by the Secure Tropos methodology [37] and that any inconsistencies are identified and rectified. The Secure Tropos set of consistency rules helps developers to check: (i) the relationships between the components of the different security-related models, such as the relationship between the security features and the threats in the security reference diagram; (ii) the consistency between the same components appearing in more than one model, such as a security constraint that appears in the security-enhanced actors model as well as in the security-enhanced goal model; and (iii) the consistency when delegation of components between actors takes place.

Name Model security references

Purpose To identify the security needs of the system under development; any problems related to the security of the system, such as threats and vulnerabilities; and also possible solutions to the security problems.

Description Developers need to consider the security features of the system under development; the protection objectives of the system; the security mechanisms and also the threats to the system's security features. Security features represent security-related attributes that the system under development must demonstrate. Protection objectives represent a set of principles or rules that contribute towards the achievement of the security features. These principles identify possible solutions to the security problems and usually they can be found in the form of the security policy of the organization. Security mechanisms represent standard security methods towards satisfying the protection objectives. Threats represent circumstances that have the potential to cause loss; or problems that can put the security features of the system in danger.

Name Realize the secure design

Purpose To develop a design that meets the security requirements of a multi-agent system.

Description During this task, developers review the system structure and identify whether extra security-related components are needed to realize the security-related functionalities of the system. Secure Tropos defines a security patterns language (for details of the language see [43]) for MAS that enables developers to select a number of design components and structures that realize the security requirements identified during the requirements stage of the Secure Tropos methodology. The process involves developers identifying a security-related problem, searching the pattern language for a pattern that provides a solution to that problem and applying the pattern to their design. Patterns used can be specified with the aid of the Secure Tropos modelling language and therefore developers are able to include them in their security-focussed models without any modification.

Name Select a secure architectural style

Purpose To explore various architectural designs for the system and select one that supports the required security level.

Description During this task, developers explore various architectural designs for the system by evaluating them against its security requirements. The degree to which different architectural styles contribute to the various system security requirements is established and an evaluation takes place that aims to identify the architectural style that satisfies most of the system's security requirements. The resultant recommended secure architectural style is documented using an architectural style selection diagram. In detail, developers model the various non-functional requirements of the system, its security requirements and a number of secure tasks that the system needs to fulfil in order for the security requirements to be satisfied. Then, a number of architectural styles are defined along with their characteristics and developers evaluate the satisfiability degree (e.g. using the Satisfiability Analysis Technique) to which each characteristic meets the system's security tasks, and therefore its security requirements. The architectural style that mostly satisfies the system's security requirements is selected.

Name Test against potential attacks

Purpose To test the system under development against potential security attacks.

Description During this task, the developed models of the system are evaluated against potential security attacks. In doing so, Secure Tropos defines Security Attack Scenarios. Potential Attackers are analysed and their goals and tasks are identified. Potential attacks are then evaluated against the system's resources (identified during the previous stages of the development process) along with the system's secure capabilities that might provide countermeasure(s) for the identified attacks. Security Attack Scenarios can be documented in Security Attack Scenarios Diagrams as shown in Figure 16.

3.4 Fragments for techniques

The techniques required in Secure Tropos are not explicitly stated. Consequently, again based on the metamodel element (TechniqueKind, Figure 11), we had to identify appropriate technique fragments from the OPF repository that could be modified to suit the security aspects of Secure Tropos or else identify areas where no such fragments pre-exist and therefore had to be formulated from the textual description in Secure Tropos and recast in the form dictated by the metamodel (see Figure 4). Our analysis resulted in indentifying ten new techniques and two existing techniques that required modification (for details see below).

Name Criticality analysis

Purpose To explore the impact that each actor has on the security of the system.

Description Different security constraints can have different impacts on the security of the system. As a result, different actors of the system might impact the security of the system differently according to what security constraints they have had imposed upon them. The developer needs to explore the impact that each actor has on the

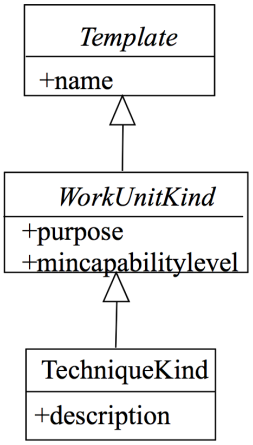


Figure 11 – Metamodel fragment for TechniqueKind.

security of the system. To do so, a technique based on the measure of security criticality is employed that allows developers to define how critical each security constraint is for the overall security of the system. The details of such method can be found in [7] but, in short, the calculation of the criticality of the system starts with consideration of the secure dependencies identified in the security-enhanced actor diagram. A value is then assigned for each security constraint (for example, see Figure 18 and the criticality value "5" assigned to the "Obtain OP Personal Information" secure dependency. Assigned criticality values are derived from a combination of a close study of the system's environment and discussion with the stakeholders. In the case of an open secure dependency (a dependency that has no security constraints attached to it), a value of zero is assigned. The range of values used in this assessment can be defined by the evaluator e.g. values restricted to the range 1-5, where 1 = very low, 2 = low, 3 = medium, 4 = high, 5 = very high. In addition, a maximum value of criticality should be defined for each actor to take into account the actor's abilities, their available time and their responsibilities within the organization [7].

Name Complexity analysis

Purpose To analyse how complex it is for each actor to achieve the security constraints he/she has been imposed.

Description Security complexity helps to design sub-systems to support actors that might be in danger of not achieving some security constraints that have been imposed on them; an undesirable situation that endangers the overall security of the system. Complexity analysis is based on the measures of security complexity and system complexity, where the former is the effort required by an actor to achieve a particular security constraint and the latter is the effort required from the dependee for achieving the dependum [18]. These two cases are differentiated since an actor's security complexity may be high, even if the system complexity is low. On the other hand, there might be cases where an actor's security complexity might be low but the system complexity is high. In such a case, it may not be possible to achieve all the security constraints imposed on the system. Thus, by taking into consideration both

system and security complexity, the developer can identify more precisely the degree of achievement of the security complexity. Similar to criticality analysis, complexity analysis assumes that complexity (system and security) can obtain integer values within the range 1-5, where 1 = very low, 2= low, 3=medium, 4=high, 5=very high. Also similarly to criticality, a maximum value of (overall) complexity is defined for each actor [7]. Complexity analysis is documented in the security enhanced goal diagram (see for example Figure 19).

Name Security Models Consistency

Purpose To check the security process and the produced security models for consistency.

Description Consistency validation may be undertaken by manual cross-referencing, i.e. developers manually evaluate the security aspects of the models and compare them against a number of consistency rules that have been defined (e.g. [37]). The rules are expressed in a natural language and they can be applied more than once when checking the models and the process. Consistency rules can be divided into inner-model rules, which help to check the consistency within a model, and outer model rules, which help to check the consistency across the different models of a process. In particular, developers use the identified set of consistency rules to: (i) check the relationships between the components of the different security-related models, such as the relationship between the security features and the threats in the security reference diagram; (ii) check the consistency between the same components thar appear in more than one model, such as a security constraints that appear in the actors' models as well as in goal models; (iii) and check the consistency when delegation of components, between actors, takes place.

Name Satisfiability analysis

Purpose To select among alternative architectural styles using as criteria the non-functional requirements of the system under development.

Description Satisfiability analysis is based on an independent probabilistic model that uses the measure of satisfiability proposed in [19]. Satisfiability represents the probability that a non-functional requirement will be satisfied. Therefore, the analysis involves the identification of specific non-functional requirements and the evaluation of different architectural styles against these requirements. The evaluation results in contribution relationships from the different architectural styles to the probability of satisfying the non-functional requirements of the system. To express the contribution of each style to the satisfiability of each non-functional requirement of the system, a weight is assigned. Weights take a value between 0 and 1.

Name Security Constraints modelling

Purpose To identify security constraints imposed on the actors and add to existing actor diagrams.

Description Security constraints imposed on the actors are identified and documented using security-enhanced actor diagrams (see for instance Figure 18). Security constraints can be categorized into actor-imposed and environment-imposed. The first category includes security constraints imposed by one actor on another as part of a secure dependency. Such constraints are typically identified by analysing the security-related concerns that an actor might have as part of the dependency. For instance, when actor A depends on actor B to obtain some information, if that information is sensitive to actor B, then most likely actor B would introduce a security constraint to ensure that the information is not abused. The second category involves security constraints imposed by the environment. Such security constraints are typically identified by analysing security policies, laws, rules or regulations that are relevant to the system and its stakeholders. The results of the security constraints identification analysis is documented in a security-enhanced actor diagram (see Figure 18 for an example of such diagram).

Name Secure goal introduction

Purpose To identify secure goals related to security constraints.

Description Developers examine the security constraints imposed on individual actors and documented in the security-enhanced goal diagram, and identify any related secure goals that assist in satisfying those security constraints. The process of identifying secure goals is similar to the process used in goal-oriented approaches and involves techniques such as means-end analysis [5]. However, such techniques are combined with a number of security-related techniques such as attack trees [52] and security reference diagrams [37]. The Secure Goal Introduction analysis enables developers to refine the goals of an actor to allow the satisfaction of a security constraint. In some cases it is necessary to decompose security constraints into more detailed security constraints. In doing so, the AND decomposition technique is employed. The decomposed constraint is called the "root" constraint, and its satisfaction is implied if and only if all the security sub-constraints are satisfied. Identified secure goals are documented in a security-enhanced goal diagram.

Name Identification of security attack scenarios

Purpose To identify the goals and intentions of possible attackers; to identify a set of possible attack scenarios to the system; to determine whether the system copes with such attacks.

Description Security attack scenarios aim to identify the goals and the intentions of possible attackers. This information allows developers to identify a set of possible attack scenarios to the system. These scenarios can then be subjected to simulated attacks to determine how the system copes. By analysing the goals and the intentions of the attackers, the developer obtains valuable information that helps in the understanding not only how the attacker might attack the system but also why an attacker wants to attack the system. This leads to a better understanding on how possible attacks can be prevented. In addition, the application of a set of attacks to the system contributes towards the identification of attacks that the system might not be able to cope with, thus leading to the re-definition of the agents of the system and the addition of new secure capabilities to the system to assist in the protection from these

attacks. Documentation is in the form of an attack testing diagram and/or enhanced secure Tropos actor diagrams [39]. "In particular, an attacker is depicted as an actor who aims to break the security of the system. The attacker intentions are modelled as goals and tasks and their analysis follows the same reasoning techniques that the Tropos methodology employs for goal and task analysis. For the purpose of a security attack scenario, a differentiation takes place between internal and external actors of the system. Internal actors represent the core actors of the system whereas external actors represent actors that interact with the system. Such a differentiation is essential since it allows developers to identify different attacks to resources of the system that are exchanged between external and internal actors of the system." (direct quotation from [39]). An example of an attack testing diagram is shown in Figure 16.

Name Security constraint assignment

Purpose To identify the goals of an actor that are restricted by security constraints.

Description Using the information documented in the security-enhanced actor diagram(s), developers further analyse each security constraint that has not been delegated and assign these security constraints to goals of the actor that are restricted by it (the security constraint). This is done in three steps. Firstly, developers need to further analyse the goals of an actor using the Tropos/i* goal analysis techniques [5]. Secondly, they identify the security constraints (from those that an actor has been imposed) that restrict one or more goals of the actor. There is no specific way recommended to assist developers to identify what goals each security constraint restricts since this mainly depends on the developer. However, support is provided, if required by the developers, in the form of the security reference diagram [37] and security patterns [43] Thirdly, developers document that assignment with the aid of restriction links (e.g. using the Secure Tropos modelling language). The resultant from these three steps can be illustrated in a security constraint assignment diagram as shown in Figure 12.

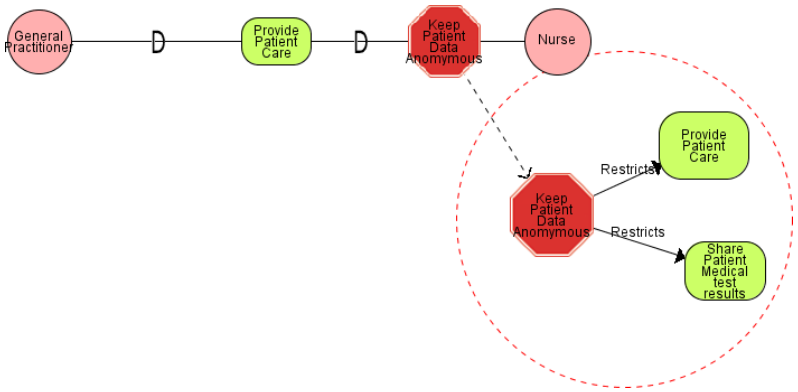


Figure 12 – Security constraint assignment.

Name Security constraint delegation

Purpose To identify cases where constraint delegation takes place.

Description Security constraints are imposed on an actor either during the security-enhanced actor analysis or during the security-enhanced goal analysis. When a security constraint is imposed on an actor, that actor is responsible for the security constraint satisfaction. However, it might be the case that an actor delegates a security constraint that has been imposed on it to another actor. Therefore, during the security constraint delegation technique, developers identify cases where constraint delegation takes place. There are two possible cases where a security constraint delegation is necessary: either because the actor that imposed the security constraint cannot satisfy the security constraint on its own or because one of the goals of the actor that a security constraint restricts is delegated to another actor and therefore the corresponding security constraint also needs to be delegated. Hence, the developer needs to consider both the security-enhanced actor diagram and the security-enhanced goal diagrams for each actor and identify any of the above cases. An example of the second case is shown in Figure 13 where a Patient actor depends on his/her general practitioner to Receive Care. A security constraint is imposed on the General Practitioner to Keep Patient's Data anonymous as part of the secure dependency "receive care". However, the General Practitioner delegates responsibility for one of the sub-goals corresponding to the "receive care" goal to the Nurse through the "Provide Patient Care" dependency. As a result, the Keep patient's data anonymous security constraint is delegated to the Nurse.

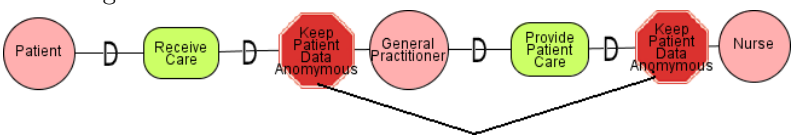


Figure 13 – Security constraint delegation.

Name Security reference diagram construction

Purpose To construct, manually or automatically, a security reference diagram.

Description A security diagram transformation system can assist developers to construct effectively a security reference diagram. It is based on the graph transformation system introduced by [1] and the analysis proposed for Tropos' actor and goal diagrams by [4]. The graph transformation system supports the progressive derivation of the security reference diagram through subsequent, increasingly precise versions of it, according to the application of a set of rules to the diagram. To automate the construction of the diagram, an algorithm has been defined that identifies the nodes of the security reference diagram by applying the defined rules. In particular, the algorithm initially identifies the nodes related to the security features of the diagram; the threats related to the drawn security features; the protection objectives applicable to the security features; the security mechanisms for the drawn protection objectives and then any identified security sub-mechanisms [37].

For the two technique fragments that require modification, we propose:

Name Secure capability modelling

Purpose To identify secure capabilities of the system's actors to guarantee the satisfaction of the security constraints.

Description Secure capabilities can be identified by considering dependencies that involve secure entities in the extended, security-enhanced actor diagram. When identified, the secure capabilities are further specified in terms of plans of particular actors of the system. This is an extension to the pre-existing OPF Technique fragment "Capabilities identification" [26].

Name Secure patterns employment

Purpose To employ a pattern language to develop a secure design.

Description To assist developers in developing a secure design, a pattern language that contains a number of security patterns can be employed. The pattern language consists of a roadmap that shows the dependencies between the patterns and points from one pattern to other patterns that the developer might want to consult once the first pattern has been applied. As patterns are applied, parts of the system are defined and later refined with the application of consecutive patterns of the language. This is an extension to the pre-existing OPF Technique: Pattern recognition [31].

3.5 Fragments for work products

All work products of Secure Tropos are represented either using i* or using UML notation with minor extensions. Five work products are identified resulting in three new work product fragments, detailed below and conformant to the metamodel fragment of Figure 14, being added to the repository: security reference diagram, security-enhanced actor diagram and architectural style selection diagram. Two existing work products in the repository require extension: security-enhanced actor diagram and security-enhanced goal diagram.

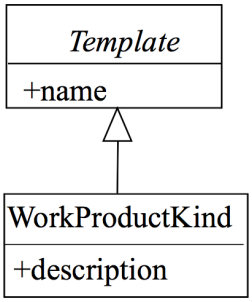


Figure 14 – Metamodel fragment for WorkProductKind.

Name Architectural style selection diagram Description: This diagram (Figure 15) uses Tropos-like notation and is used to model architectural styles, security properties and security requirements of the system under development and the different contributions that each architectural style has on the security properties and the security requirements of the system [42]. In this diagram, a hexagon represents a security solution, while an emboldened cloud represents a non-functional requirement of the system. Links represent contributions and weights represent the degree of satisfiability [37] of the architectural style (for example Client/Server — Mobile Agents) to the various nodes.

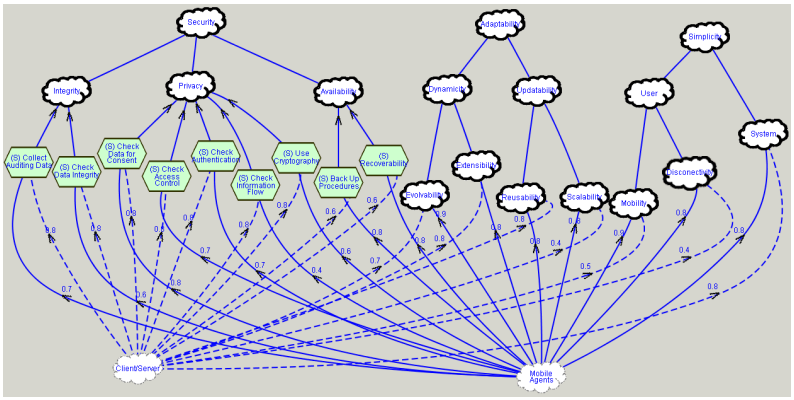


Figure 15 – Example Architectural Style Selection Diagram (from [37]).

Name Attack testing diagram

Description This diagram models possible attackers, the resources that are attacked and the agents/actors of the system related to the attack (Figure 16). In particular, an attacker is modelled as an agent/actor and its intentions are modelled as goals and tasks. Attacks are depicted as dash-lined links, called attack links, which contain the "attacks" tag, starting from one of the attackers' goals and ending on the attacked resource. Moreover, the system's agent/actor secure capabilities are modelled and links are employed to indicate which of these capabilities help towards the prevention of the attackers' goals.

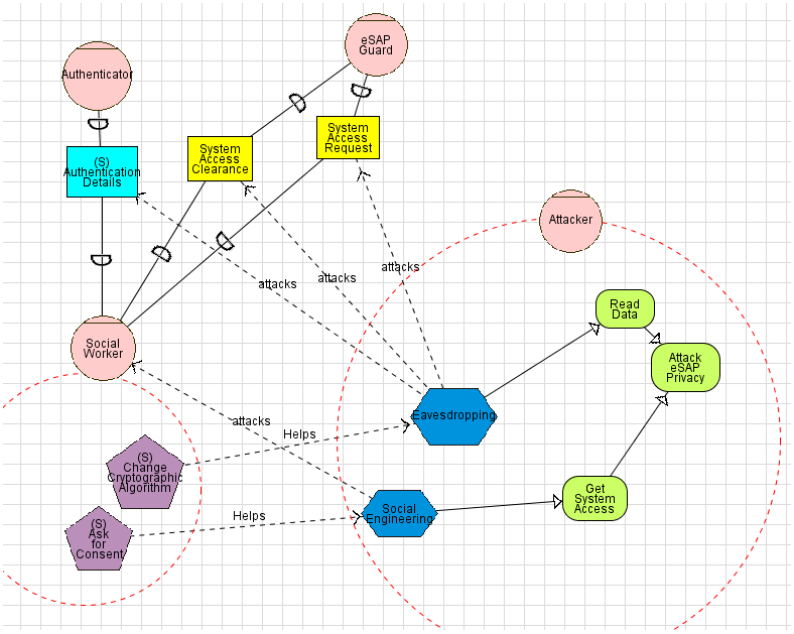


Figure 16 – Example Security Reference Diagram.

Name Security reference diagram

Description The security reference diagram is a graph that consists of a set of labelled nodes and a set of labelled directed edges, each of which connects a pair of nodes (Figure 17). Formally, this is represented as a special case of a labelled directed diagram. To control the non-deterministic derivation process during the construction of the security reference diagram, priority rules have been defined [37] and should be used by the developer.

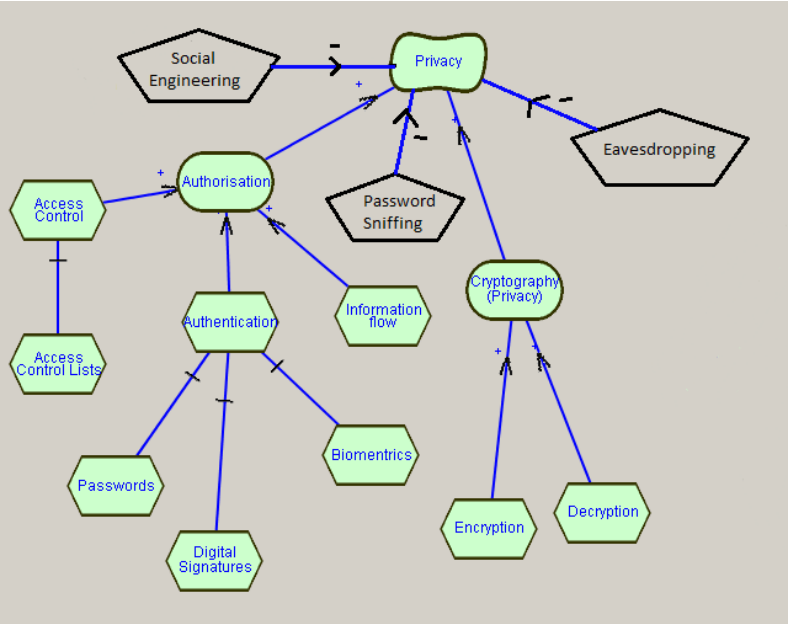
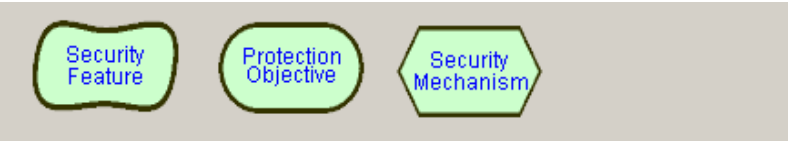


Figure 17 – Example Attack Testing Diagram (from [37]).



Legend

Name Security-enhanced actor diagram

Description This is a Tropos style diagram. There is already a work product fragment in the OPF Repository for the (Tropos) actor diagram [26]. This, however, needs extension in order to support the additions suggested by Secure Tropos. In particular, the security-enhanced actor diagram models any secure dependencies and the appropriate security constraints imposed on the network of actors modelled in the actor diagram. An example of a security-enhanced actor diagram is shown in Figure 18.

Name Security-enhanced goal diagram

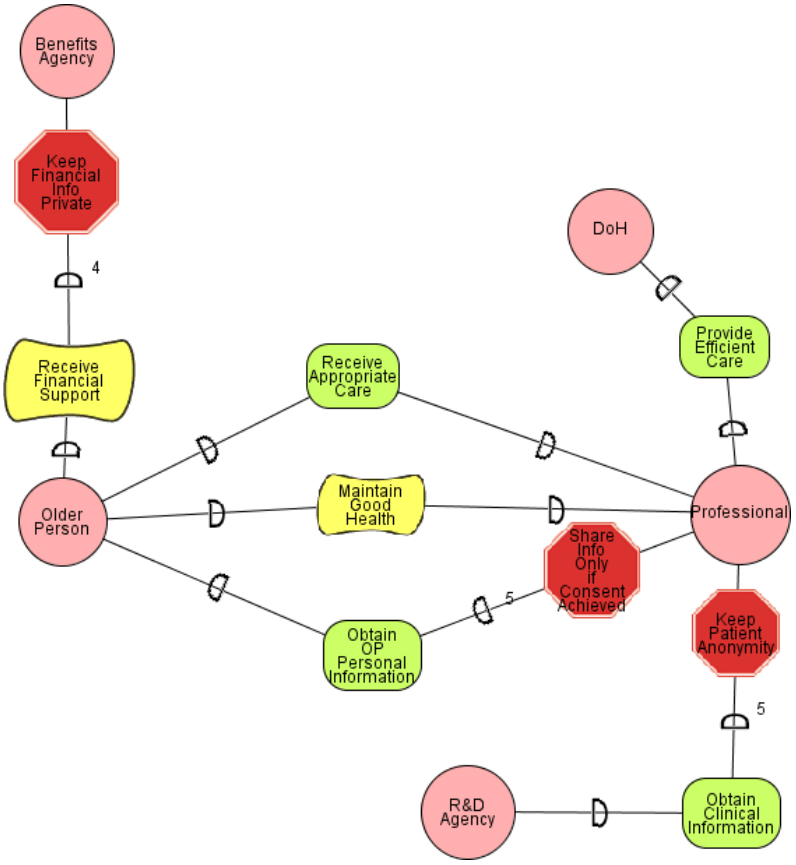


Figure 18 – Example of a security-enhanced actor diagram (after [7].)

Description This is a Tropos style diagram. It extends the Tropos goal diagram (previously incorporated into the OPF repository: [26]) by modelling the analysis of security constraints, the introduction of secure goals and secure tasks according to the techniques described above. An example of a security-enhanced goal diagram is shown in Figure 19.

4 Discussion

As we noted in Section 1, situational method engineering revolves around the duality of a metamodel and instances generated from and/or conformant with one of the elements in that metamodel. These instances are the method fragments that are stored in a method repository. Since the underpinning metamodel is standardized [33], it is unlikely that any changes will be necessary to the metamodel when we add new technologies such service orientation or the like — so long as we keep within the overall software development domain. (Clearly if we move to system development or manufacturing processes, the methodology metamodel required will necessarily be different.)

In contrast to the metamodel, the repository of fragments is highly likely to

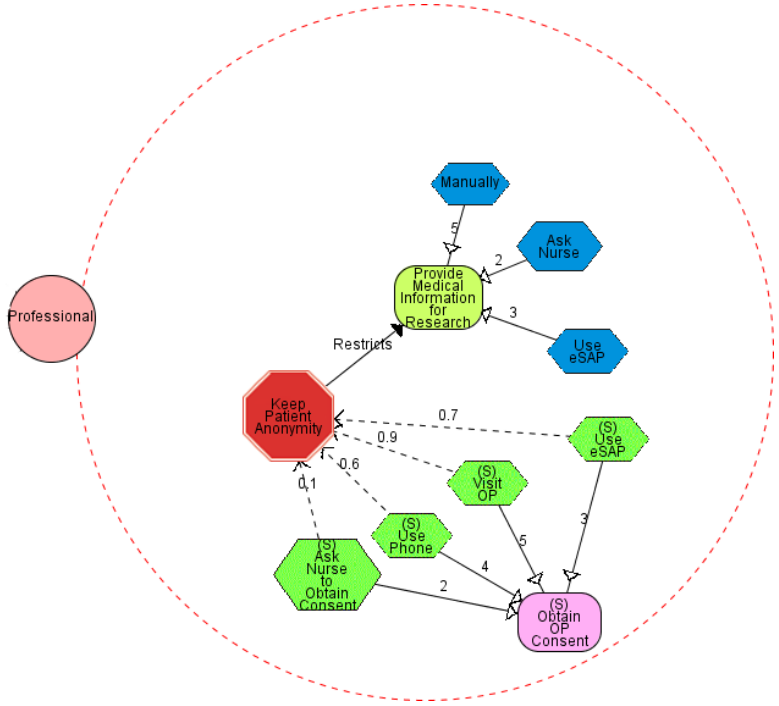


Figure 19 – Example of a security-enhanced goal diagram (after [7]).

need further additions when new methodologies are analysed from the viewpoint of fragments and SME. However, as more and more methodologies are examined, the incremental additions to the repository are anticipated to decrease — within a specific paradigmatic view (e.g. object-oriented methodologies, agent-oriented methodologies). This means that there is likely to be a reasonable chance of achieving a near closure such that further analysis will not reveal any new method fragments [23]. With OPF, this has already been done for several areas of computing including agent technology. Analysis of non-security-related agent-oriented methodologies is nearly complete. This paper is the first attempt to add security concerns to those agent-related fragments already stored in the repository.

The concerns of this paper have been to analyse how the contents of the existing OPF repository of software development fragments needs to be enhanced when new ideas regarding security are introduced from the Secure Tropos methodology. This analysis has led to our identification of three new processes, 12 new tasks, 10 new techniques and three new work products, together with some recommended extensions to existing method fragments in the OPF repository. When other security-related methodologies (or part-methodologies) can be analysed (see below), we may anticipate reaching some sort of completeness with regard to security-related fragments for constructing agent-oriented methodologies. As demonstrated [22] for non-security AOSE methodologies, this is reflected in the decreasing number of new fragments identified in the analysis of subsequent AOSE methodologies i.e. when the repository is complete, the analysis of any additional methodological approaches identifies zero new candidates for addition to the methodbase.

It should be noted that, within the repository, method fragments are essentially independently defined. Upon abstraction of these from the repository as part of the construction of a new, situationally-specific methodology, they need to be connected in a meaningful way. While the formal underpinning theories to do this are still lacking in the SME literature (S. Brinkkemper, p.c., 2005), there is some assistance by means of classes in the metamodel that provide such linkages. For example, the OPF approach has a metaclass called Action to permit the linking of task fragments with work product fragments, a metaclass called WorkPerformance to link work unit fragments and producer fragments etc. A second gap in the literature is that of quality assessment [24, 54] — both for the constructed methodology in terms of internal consistency and appropriate fit to the contextual situation (the "S' in SME) and also in terms of how it works on real software development projects. In the latter case, there is some minimal literature on industrial case studies e.g. [27].

As a further step towards a true "validation", it becomes possible, with these method fragments newly added to the OPF repository, not only to reconstruct Secure Tropos but, much more importantly, to include some, or all, of the newly identified security-related method fragments into other agent-oriented methodologies making their identification more widely useful. This kind of enhancement has been done in the non-secure agent environment by adding Tropos method fragments to the Prometheus approach (see Chapter 13 in [25]). It also means that the newly added Secure Tropos method fragments are available for users of other methodologies built from OPF method fragments, thus permitting the addition of security concerns to any of the other, already supported agent-oriented methodologies. Below, we discuss how two well known and important agent-oriented software engineering methodologies, Gaia and MaSE, could benefit from using the fragments identified in this paper.

During the analysis stage of the Gaia methodology [58, 60], the security reference diagram could be constructed to assist in the identification of permissions and the security requirements of the system could be identified taking into consideration the identified roles and their permissions. In doing so, the "Security Reference Diagram Construction" and the "Analyse Actor Security" task can be employed. Then during the design stage, the "Realize the secure design" task can be used to help in the aggregation of the roles to agents, and the architectural style of the system could be identified using the proposed "Select an architectural style" task. Finally, security attack scenarios could be developed using the agents, services and acquaintance models of Gaia and with the aid of the "Test against potential attacks" task and "Identification of security attack scenarios" technique.

The MaSE methodology [14, 15] starts its requirement analysis by capturing the system's goals. As such, a number of the identified fragments can be employed to assist developers to develop a security requirements identification activity, which can be integrated within the analysis stage of the MaSE methodology. For instance, the "Secure goal introduction" technique could be employed together with the "Security constraint assignment" technique. The fragments that support the "Security analysis of the system" process could be employed during the assembling agent classes activity, whereas the fragments that support the selection of the architectural style according to the security requirements of the system could be integrated within the system design activity.

4.1 Related Work

Although security issues have been addressed in general (e.g. application to SOA in [8]), to the best of our knowledge, this is the first effort to identify and document a set of security-related fragments for multi-agent systems. Moreover, there is little reported research[4] into defining security-related agent-oriented methods and methodologies. As stated earlier, Secure Tropos is the only agent-oriented methodology that enables developers to consider security throughout the full development lifecycle from requirements to design and testing. However, the literature provides some agent-oriented methods that only focus on specific development stages, and especially in requirements. We now briefly review this related work with a view to identifying additional security-related fragments. Liu et al. [36] have presented work to identify security requirements during the development of multiagent systems. In this work, security requirements are analysed as relationships amongst strategic actors, such as users, stakeholders and potential attackers. These authors propose three different kinds of analysis techniques: agent-oriented, goal-oriented and scenario-based analysis. Agent-oriented analysis is used to model potential threats and security measures, whereas goal-oriented analysis is employed for the development of a catalogue to help towards the identification of the different security relationships on the system. Finally, scenario-based analysis is considered to be an elaboration of the other two kinds of analysis. This work is based on existing modelling techniques, and security issues are represented and reasoned about without special notations and constructs. As such, we cannot identify security specific fragments.

The SI* approach [61] has also been proposed as a security-related extension to i*. That work uses the concepts of trust and delegation to analyse security issues. Nevertheless, it is only applicable to the Requirements Engineering stage and therefore it does not fully support the development of multi-agent systems. Indeed, very recent work [38] has tried to integrate the SI* approach with the Secure Tropos methodology. This integration is not yet finalized. However, once the integration is finalized, we plan to investigate this work further and explore the possibility of identifying additional security fragments, derived from non-agent contexts yet potentially useful to extend the support for security issues in AO methodologies such as Secure Tropos. Huget [32] has proposed a new agent-oriented methodology called Nemo, which is claimed to support security. However, security is not considered as a specific model but it is included within the other models of the methodology. From the current description of this methodology in the literature, security seems to be considered quite superficially, and no specific security-related processes, techniques and/or work products are defined. Therefore, unless the methodology is further developed and more attention is paid to its security dimension, no additional security-related fragments can be identified.

5 Conclusions and Future Work

In the work presented here, we have analysed the Secure Tropos methodology and we have identified a number of method fragments to be added into the OPF repository and a number of existing method fragments that need enhancement. Secure Tropos provides a new domain beyond the agent-oriented methodologies already analysed and

[4] Although a number of efforts have been reported in the literature that consider security as part of the software engineering development process, these are not specialized for agent-oriented systems and therefore we do not review them in this section. Interested readers can review [38].

thus it is not surprising that such a significant number of new method fragments have been found in this study.

Future work includes the ratification of a secure, agent-oriented methodology constructed using these new security-related fragments (together with other pre-existing ones, of course) on a real project. Such empirical work is challenging and relies on a research methodology such as action research [2] and the slow build-up of a number of such "case studies" — unfortunately, statistical approaches are completely impossible and uptake by industry of secure agent-oriented systems is extremely rare.

Clearly there are other agent-related domains that are outside of standard AO methodologies. As well as security, another obvious domain extension is in methodologies for the design of mobile agent systems. This will be a topic of future work, as will assessing the likely closure in both this and the security domain analysed in this paper.

Acknowledgments This is contribution number 09/06 of the Centre for Object Technology Applications and Research.

References

[1] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schurr, G. Taentzer, Graph Transformation for Specification and Programming, Science of Computer Programming 34 (1999) 1-54.

[2] D. E. Avison, F. Lau, M. Myers, P. A. Nielsen, Making academic research more relevant, Communications of the ACM 42(1) (1999) 94-97.

[3] M. Bajec, D. Vavpoti?, M. Krisper, Practice-driven approach for creating project-specific software development methods, Information and Software Technology 49(4) (2007) 345-365.

[4] P. Bresciani, P. Giorgini, The Tropos Analysis Process as Graph Transformation System, Workshop on Agent-oriented methodologies at OOPSLA 2002 (Seattle, WA, USA, 2002).

[5] P. Bresciani , P. Giorgini, F. Giunchiglia, J. Mylopoulos, A. Perini, TROPOS:An Agent Oriented Software Development Methodology, Journal of Autonomous Agents and Multi-Agent Systems 8(3) (2004) 203-236.

[6] P. Bresciani, P. Giorgini, H. Mouratidis, On Security Requirements Analysis for Multi-Agent Systems, 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems SELMAS 2003 (Oregon, USA, 2003).

[7] P. Bresciani, P. Giorgini, H. Mouratidis, G. Manson Multi-Agent Systems and Security Requirements Analysis. in: C Lucena, A Garcia, A Romanovsky, J Castro, P Alencar (eds) Advances in Software Engineering for Multi-Agent Systems, LNCS2940 (Springer-Verlag, Berlin,2004) 35-48.

[8] R. Breu, M. Hafner, F. Innerhofer-Oberperfler, F. Wozak Model-Driven Security Engineering of Service Oriented Systems. in: Information Systems and e-Business Technologies, 5 (Springer Berlin Heidelberg,2008) 59-71.

[9] S. Brinkkemper, Method Engineering: Engineering of Information Systems Development Methods and Tools, Information and Software Technology 38(4) (1996) 275-280.

[10] S. Brinkkemper, M. Saeki, F. Harmsen, Assembly techniques for method engineering, Advanced Information Systems Engineering, Proceedings 10th International Conference, CAiSE'98 (Pisa, Italy, 1998) 381-400.

[11] S. Brinkkemper, M. Saeki, F. Harmsen, Meta-Modelling Based Assembly Techniques for Situational Method Engineering, Information Systems 24(3) (1999) 209-228.

[12] J. Castro, M. Kolp, J. Mylopoulos, Towards Requirements-Driven Information Systems Engineering: The Tropos Project, Information Systems 27 (2002) 365-389.

[13] M. Cossentino, S. Gaglio, A. Garro, V. Seidita, Method fragments for agent design methodologies: from standardisation to research, Int. J. Agent-Oriented Software Eng. 1(1) (2007) 91-121.

[14] S. A. Deloach, Multi-agent Systems Engineering: A Methodology and Language for Designing Agent Systems, Proceedings Agent-Oriented Information Systems '99 (AOIS'99) (Seattle, WA, USA, 1999).

[15] S. A. Deloach, M. Kumar Multi-agent Systems Engineering: An Overview and Case Study. in: B Henderson-Sellers, P Giorgini (eds) Agent-Oriented Methodologies, (IDEA Group Publishing,2005) 236-276.

[16] P. Devanbu, S. Stubblebine, Software Engineering for Security: A roadmap, Proceedings of the 22nd International Conference on Software Engineering. Track on the Future of Software Engineering (Limerick -Ireland, 2000).

[17] D. G. Firesmith, B. Henderson-Sellers, The OPEN Process Framework (Addison-Wesley, London, 2002).

[18] M. Garzetti, P. Giorgini, J. Mylopoulos, F. Sannicolo, Applying Tropos Methodology to a real case study: Complexity and Criticality Analysis, Proceedings of the Second Italian workshop on "WOA 2002 dagli oggetti agli agenti dall'informazione alla conoscenza" (Milano-Italy, 2002).

[19] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani, Reasoning with Goal Models, Procs. 21st International Conference on Conceptual Modelling (ER2002) (Tampere, Finland, 2002) 167-181.

[20] P. Giorgini, A. Perini, J. Mylopoulos, F. Giunchiglia, B. P., Agent-Oriented Software Development: A Case Study, Thirteenth International Conference on Software Engineering and Knowledge Engineering (SEKE01) (Buenos Aires, Argentina, 2001).

[21] C. Gonzalez-Perez, B. Henderson-Sellers An ontology for software development methodologies and endeavours. in: C Calero, F Ruiz, M Piattini (eds) Ontologies in Software Engineering and Software Technology, (Springer-Verlag,2006) 123-152.

[22] A. F. Harmsen, Situational Method Engineering. Moret Ernst & Young, Amsterdam, The Netherlands, 1997

[23] B. Henderson-Sellers, Evaluating the feasibility of method engineering for the creation of agent-oriented methodologies, Proceedings Multi-Agent Systems and Applications IV. 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005 (Budapest, Hungary, 2005) 142-152.

[24] B. Henderson-Sellers, Some quality issues for conceptual models, Dagstuhl seminar 08181, http://kathrin.dagstuhl.de/08181/Materials2/ (2008).

[25] B. Henderson-Sellers, P. Giorgini (eds) Agent-Oriented Methodologies. (Idea Group, Hershey, USA, 2005).

[26] B. Henderson-Sellers, P. Giorgini, P. Bresciani Enhancing Agent OPEN with concepts used in the Tropos methodology. in: A Omicini, P Pettra, J Pitt (eds) Engineering Societies in the Agents World IV. 4th International Workshop, ESAW 2003, LNAI 3071 (Springer-Verlag, Berlin,2004) 328-345.

[27] B. Henderson-Sellers, A. Qumer, Using method engineering to make a traditional environment agile, Cutter IT Journal 20(5) (2007) 61-74.

[28] B. Henderson-Sellers and J. Ralyté, Situational Method Engineering: State-of-the-Art Review, Journal of Universal Computer Science, 16(3), 424-478, http://www.jucs.org/jucs_16_3/situational_method_engineering_state, (2010).

[29] B. Henderson-Sellers, M. Serour, T. Mcbride, C. Gonzalez-Perez, L. Dagher, Process construction and customization, J. Universal Computer Science 10(4) (2004) 326-358.

[30] B. Henderson-Sellers, M. K. Serour, Creating a dual agility method - the value of method engineering, J. Database Management 16(4) (2005) 1-24.

[31] B. Henderson-Sellers, A. Simons, H. Younessi, The OPEN Toolbox of Techniques (Addison-Wesley Longman, Harlow (Essex), UK, 1998).

[32] M.-P. Huget, Nemo: An Agent-Oriented Software Engineering Methodology, OOPSLA Workshop on Agent-Oriented Methodologies (Seattle, USA, 2002).

[33] ISO/IEC 24744. Software Engineering – Metamodel for Software Development Methodologies, ISO, Geneva, 2007.

[34] N. Jayaratna, Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework (McGraw-Hill, New York, 1994).

[35] K. Kumar, R. J. Welke Methodology Engineering: a Proposal for Situation-Specific Methodology Construction. in: Ww Cotterman, Ja Senn (eds) Challenges and Strategies for Research in Systems Development, (John Wiley & Sons, Chichester, UK,1992) 257-269.

[36] L. Liu, E. Yu, J. Mylopoulos., Analyzing Security Requirements as Relationships Among Strategic Actors, 2nd Symposium on Requirements Engineering for Information Security (SREIS'02) (Raleigh, North Carolina, 2002).

[37] H. Mouratidis, A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of Health and Social Care Needs of Older People in England, Thesis, Department of Computer Science, University of Sheffield, 2004.

[38] H. Mouratidis, P. Giorgini, Integrating Security and Software Engineering: Advances and Future Vision (IDEA Group Publishing, 2006).

[39] H. Mouratidis, P. Giorgini, Security Attack Testing (SAT)–testing the security of information systems at design time, Information Systems 32(8) (2007) 1166-1183.

[40] H. Mouratidis, P. Giorgini, G. Manson, Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems, 15th International Conference on Advance Information Systems (CAiSE) (Velden - Austria, 2003) 63-78.

[41] H. Mouratidis, P. Giorgini, G. Manson, Modelling Secure Multiagent Systems, AAMAS'03 (Melbourne, Australia, 2003) 859-866.

[42] H. Mouratidis, P. Giorgini, G. Manson, When security meets software engineering: A case of modelling secure information systems, Information Systems 30(8) (2005) 609-629.

[43] H. Mouratidis, G. Weiss, P. Giorgini, Modelling Secure Systems Using An Agent Oriented Approach and Security Patterns, International Journal of Software Engineering and Knowledge Engineering (IJSEKE) 16(3) (2006) 471-498.

[44] J. Pavon, J. Gomez-Sanz, R. Fuentest The INGENIAS Methodology and Tools. in: B Henderson-Sellers, P Giorgini (eds) Agent-Oriented Methodologies, (IDEA Group Publishing, 2005) 236-276.

[45] L. Penserini, A. Perini, A. Susi, J. Mylopoulos, High variability design for software agents: Extending Tropos, ACM Transactions on Autonomous and Adaptative Systems 2(4) (2007) 16-27.

[46] A. Perini, P. Bresciani, P. Giorgini, G. Giunchiglia, J. Mylopoulos, A Knowledge Level Software Engineering Methodology for Agent Oriented Programming, Proceedings of the Fifth International Conference on Autonomous Agents (Montreal, Canada, 2001).

[47] J. Ralyte, Towards situational methods for information systems development: engineering reusable method chunks, Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education (Vilnius, Lithuania, 2004) 271-282.

[48] J. Ralyté, C. Rolland, An Approach for Method Reengineering, Proceedings of the 20th International Conference on Conceptual Modelling, ER2001 (Yokohama, Japan, 2001) 471-484.

[49] J. Ralyté, C. Rolland An assembly process model for method engineering. in: Kr Dittrich, A Geppert, Mc Norrie (eds) Advanced Information Systems Engineering, LNCS2068, (Springer-Verlag, Berlin,2001) 267-283.

[50] A. S. Rao, M. P. Georgeff, BDI agents: from theory to practice, First International Conference on Multi Agent Systems (San Francisco, CA, USA, 1995) 312-319.

[51] C. Rolland, N. Prakash, A. Benjamen, A Multi-Model View of Process Modelling, Requirements Engineering Journal 4(4) (1999) 169-187.

[52] B. Schneier, Secrets and Lies: Digital Security in a Networked World (John Wiley & Sons, New York, NY, 2000).

[53] V. Seidita, M. Cossentino, S. Gaglio, Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies, AOSE2008 (Estoril, Portugal, 2008).

[54] V. Shekhovtsov, On conceptualization of quality, Dagstuhl seminar 08181, `http://kathrin.dagstuhl.de/08181/Materials2/` (2008).

[55] E. Steegmans, J. Lewi, M. D'haese, J. Dockx, D. Jehoul, B. Swennen, S. Van Baelen, P. Van Hirtum EROOS Reference Manual Version 1.0. in:(Department of Computer Science, K.U.Leuven. 1995) 176.

[56] K. Van Slooten, B. Hodes, Characterizing IS development projects, IFIP TC8 Working Conference on Method Engineering: Principles of method construction and tool support (London, 1996).

[57] V. Waller, R. B. Johnston, S. K. Milton Development of a situation information systems analysis and design methodology: a health care setting. in: Z Irani, Od Sarikas, J Llopis, R Gonzalez, J Gasco (eds) European and Mediterranean Conference on Information Systems 2006 (EMCIS2006). Brunel University, West London. 2006) 8pp.

[58] M. Wooldridge, N. R. Jennings, D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems (The Netherlands, 2000) pp 285-312.

[59] E. Yu, Modelling Strategic Relationships for Process Reengineering, PhD Thesis, Department of Computer Science, University of Toronto, 1995.

[60] F. Zambonelli, N. Jennings, M. Wooldridge, Developing Multiagent Systems: the Gaia Methodology, ACM Transactions on Software Engineering and Methodology 12(3) (2003) 417-470.

[61] N. Zannone, A Requirements Engineering Methodology for Trust, Security, and Privacy, PhD Thesis, Department of Information and Communication Technology, University of Trento, 2007.

About the authors

Graham Low is Professor of Information Systems at The University of New South Wales. Graham's research focuses on the implementation and adoption of new technologies by the IS/IT industry such as methodological approaches to agent oriented information systems design; and management of the information systems design and implementation process. E-Mail: `g.low@unsw.edu.au`

Haralambos Mouratidis is Principal Lecturer in Secure Systems and Software Development at the School of Computing, IT and Engineering ad the University of East London. His research interests focus on secure information systems development and agent oriented software engineering. He is Editor in Chief of the International Journal of Computer Science and Secure Systems and he has published more than 80 papers. Email: `haris@uel.ac.uk`

Brian Henderson-Sellers is Director of the Centre for Object Technology Applications and Research and Professor of Information Systems at University of Technology, Sydney (UTS). He is author of ten books on object technology and is well known for his work in OO methodologies (MOSES, COMMA and OPEN) and in OO metrics. He was recently awarded a DSc degree by the University of London for his work in object-oriented methodology. E-Mail: `brian@it.uts.edu.au`