Journal of the Association for Information Systems
Research Article

# A Design Theory for Secure Information Systems Design Methods[1]

**Mikko Siponen**
Department of Information Processing Science,
University of Oulu, Finland,
*Mikko.T.siponen@oulu.fi*

**Richard Baskerville**
Georgia State University,
baskerville@gsu.edu

**Juhani Heikka**
Department of Information Processing Science,
University of Oulu, Finland,

## *Abstract*

*Many alternative methods for designing secure information systems (SIS) have been proposed to ensure system security. However, within all the literature on SIS methods, there exists little theoretically grounded work that addresses the fundamental requirements and goals of SIS design. This paper first uses design theory to develop a SIS design theory framework that defines six requirements for SIS design methods, and second, shows how known SIS design methods fail to satisfy these requirements. Third, the paper describes a SIS design method that does address these requirements and reports two empirical studies that demonstrate the validity of the proposed framework.*

## Introduction

The increased utilization of information systems (IS) and the Internet has brought security issues to the fore. To ensure that systems are secured, researchers have constructed various SIS design methods (see review articles by Baskerville, 1993; Dhillon and Backhouse, 2001; Siponen, 2005a). But reviewers find a fundamental problem with these SIS design methods: they are theoretically underdeveloped (Baskerville, 1992 p. 410; Willison, 2002 p. 15), "lacking serious research" (Baskerville, 1994 p. 385), or are glutted with "nonsense" (Olnes, 1994 p. 628). As the design theory

[1] Kalle Lyytinen was the Senior Accepting editor of this paper. This paper was submitted on December 3rd 2002 and went through 2 revisions.

approach has proved successful in prescribing better design processes for emergent classes of systems (Walls, et al. 2004), reframing development of SIS design methods as a design science activity is a promising way to advance past this problem.[2]
The purpose of this paper is to describe the development of a fundamental SIS design theory à la Walls et al. (1992). This theory prescribes six requirements, called meta-requirements, for SIS design methods. Through an analysis of the extent to which prior SIS design methods address the six meta-requirements, we learn that none of the existing methods address all six. The SIS design theory enables us to advance a SIS design method, called Meta-notation, which does address the six SIS meta-requirements, and we are able to validate this solution (Meta-notation) through action research. These results contribute to the IS research community with a novel and promising basis for SIS design methods thinking, and the successful action research intervention demonstrates the practical value of the resulting SIS design methods for practitioners.

The paper is organized as follows. The second section discusses the design theory for SIS design methods with six meta-requirements and reviews the extent to which extant SIS design methods meet these requirements. The third section describes the background of the new solution, and the fourth section illustrates how it can be applied to an Information Systems Development (ISD) method. In the fifth section, we describe the research approach used. The sixth section presents the preliminary clinical fieldwork findings, while the seventh section presents the action research findings. Section eight is a discussion, including the implications of the study. Finally, the last section concludes with a summary of the key findings.

## SIS Design Theory And SIS Design Method Meta-Requirements

Walls et al. (1992) envisaged design theories as prescriptive (how to design) and goal-oriented; thus they differ from predictive or explanatory natural science theories. An IS design theory is divided into two aspects; *design* and *product*, and consists of a set of meta-requirements, which are derived from relevant kernel or reference theories (Figure 1). The meta-requirements also inform meta-design principles. A set of testable design product hypotheses is used to test whether the meta-design meets the meta-requirements. In turn, the testable design process hypotheses are utilized to verify whether the design method results in an artifact that is consistent with the meta-design. Both positivistic and interpretive research methods could be used to test these hypotheses (Markus et al., 2002).

[2] Other areas have encountered dissimilar, but fundamental problems in proposed design theory solutions, such as Executive Information Systems (Walls et al., 1992), Decision Support Systems (Kasper, 1996), Knowledge Management (Markus et al., 2002) and Web-based Education (Jones et al., 2003). Working from Simon's (1996) Science of the Artificial, Walls et al. (1992) provided the seminal expression of design theory for IS.

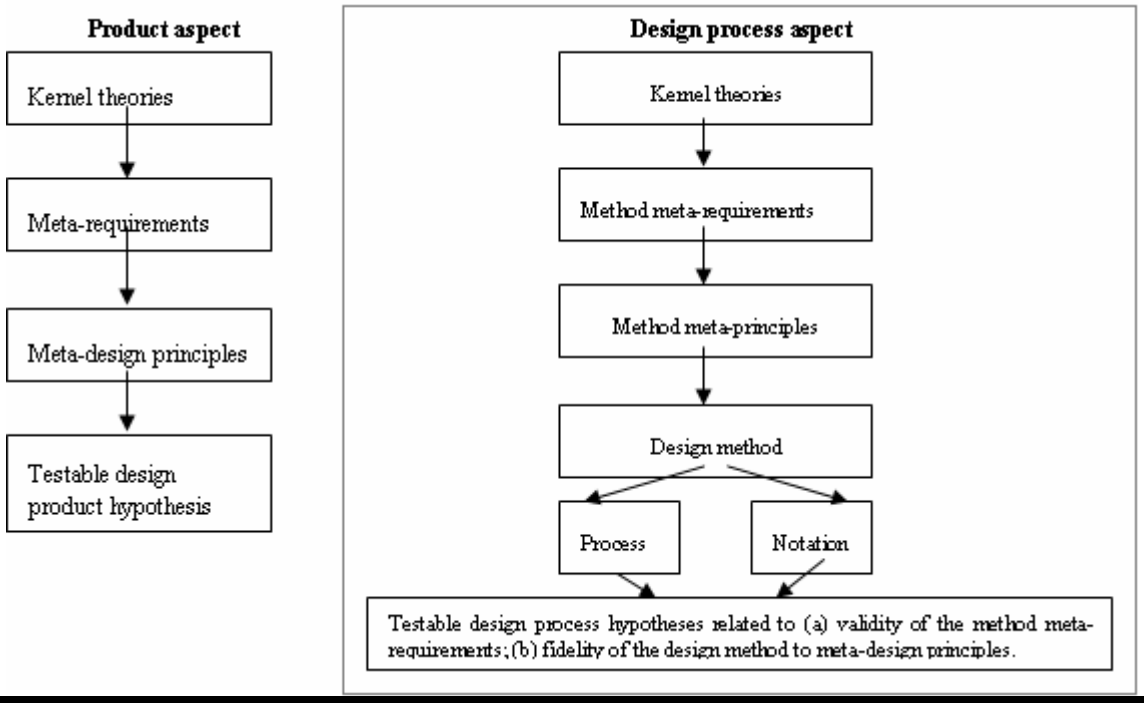


**Figure 1. An IS design theory (adapted from Walls et al., 1992) that elaborates the design method into process and notation. and the design hypotheses into two areas: (a) validity of the method meta-requirements; (b) the design method fidelity to the method meta-design principles.**

### *SIS Design Theory: An Elaboration*

Design theory adapts well to different contexts, but for good fit it requires elaboration or modification in other instances (Kasper, 1996; Markus et al., 2002). We found this to be the case in SIS design, and we elaborated the design theory framework by Walls et al. (1992) in two ways. First, we divided the design method into separate *process* and *notation* aspects, following similar critical work on ISD methods (Hirschheim and Klein, 1992; Gause and Weinberg, 1989). This elaboration is useful because our aims included the development of concepts for creating situated SIS design methods. Second, because our focus is on the design process aspect, we divided the design meta-methodological hypotheses into two areas: (a) validity of the method meta-requirements,[3] and (b) fidelity of the SIS design method to the meta-design principles.

[3] The a-type of hypotheses means that a SIS design method incorporating meta-requirements is assumed to be more successful in practice than a SIS design method that does not have such features. This success is evaluated in terms of practitioners' preferences. So, for example, with respect to modeling support (see MR 2 below), practitioners in general are assumed to prefer SIS design methods that offer modeling support at three levels of abstraction over a SIS that does not provide such support.

We will focus on the design process aspect of design theory. The product aspect, as described by Walls et al. (1992), is most applicable in generic, off-the-shelf software. Such design products must themselves be examined for generality, and thus themselves draw upon kernel theories and meta-principles. However, our focus is on the most widespread SIS design problem: SIS for a particular organization's system or setting. In such design settings, the security requirements of the product (SIS or software) are highly situated, and are determined mainly by the client. Recognizing this, we draw the meta-principles into the design process aspect and concentrate only on the design aspect in our study, represented by the right-hand part of Figure 1.

The design process aspect guides the construction of the product. According to Walls et al. (1992), a design method, based on the idea of the IS design theory, can focus on any stage of ISD (e.g., requirements analysis, implementation, testing). Next, we describe six SIS design method meta-requirements and respective kernel theories, starting from the fundamental kernel theory.

## *SIS Design Theory: Fundamental Kernel Theory*

Clements (1977) developed a theory of SIS based on a model of the relationship between a set of system objects (each with a loss value), a set of threats (each with a likelihood), and a set of IS security features (each with a resistance). The relationship involved the association of potential intrusion activities with each relevant member of the set of system objects (Figure 2).

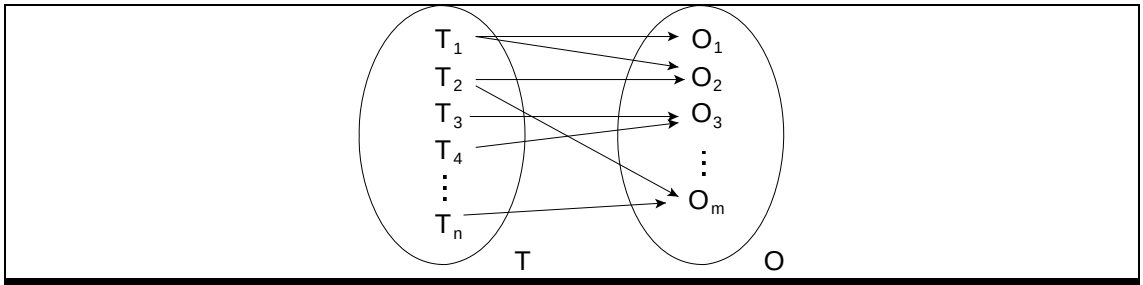


**Figure 2. The threat-object relationships and edges (adapted from Clements 1977).**

These threat-object relations defined a set of edges $T_iO_j$ that represent the components of insecurity or risk in systems. Security features provide a "firewall function" (Clements, 1977 p. 532) that resists penetration of any particular threat through to any particular object. In a protected system, any TO edge is eliminated by inserting a feature to establish a TF edge and an FO edge in place of the TO edge. Thus all edges are prescribed in the forms $T_iF_k$ and $F_kO_j$ that represent the insertion of security features between threats and system objects. In a protected system, all edges $T_iO_j$ are unprotected objects and contribute to the insecurity or vulnerability of the overall system (Figure 3).

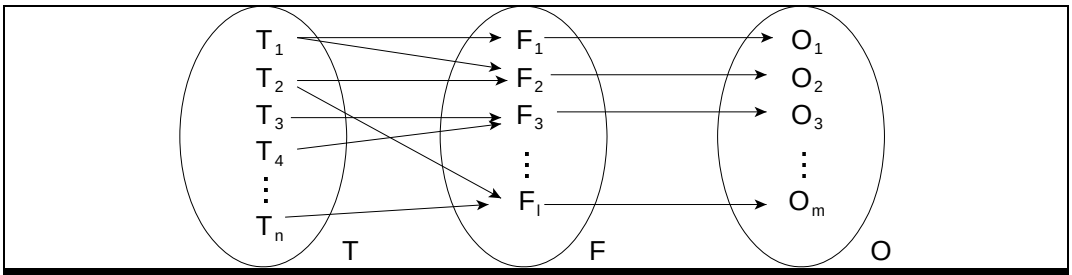


**Figure 3. Clements' basic security system (adapted from Clements 1977).**

Clements' (1977) theory describes the basic relationships between threats and system assets, and the intended role of designed controls or safeguards established to prevent the effects of threats on assets. This is the fundamental theory aimed at describing the heart of SIS design methods, that seeks to specify and deploy safeguards to prevent losses due to threats. This theory provides the most basic set of assumptions among the kernel theories employed in our design theory.

While Clements' (1977) theory is sufficient as a basis for the goals of any SIS design method, it provides for no features beyond the goals in terms of the basic security threat-feature-object triplets. In order to address this issue, we draw from a number of other theories. These other theories are used as kernel theories in developing the requirements for a SIS design method to integrate with other ISD methods. We will describe each of these additional kernel theories in the next section.

**Table 1: Components of the design aspect and respective SIS features.**

| The components of the design aspect | SIS features |
|---|---|
| **1. SIS design method meta-requirements** | **1.0** A SIS design methods must develop security features that resist the impact of threats on system objects.<br>**1.1** Organizations' security requirements must be the point of departure in the development of SIS.<br>**1.2** SIS design methods should provide abstract representation and operations for specifying the three essential elements of secure systems - threats, objects, and security features (safeguards or controls) - for the three levels of abstraction: organizational, conceptual and technical.<br>**1.3** SIS design methods should support its integration into normal ISD methods.<br>**1.4** SIS design methods should maximize the autonomy of developers.<br>**1.5** SIS design methods should be adaptable to forthcoming ISD methods. |
| **2. Kernel theories** | **2.0** Clements' Threat-Feature-Object Model<br>**2.1** Hume's law "no ought from (an) is".<br>**2.2** Metamodels of IS.<br>**2.3** The theory of developmental duality, Ockham's razor and the thesis of ontological dependencies.<br>**2.4** Autonomy in philosophy.<br>**2.5** Emergence: autopoesis and self-reproduction. |
| **3. SIS design method meta-design principles** | **3.0** SIS design methods should incorporate security features that resist the impact of threats on system objects.<br>**3.1** SIS design methods must explicitly and thoroughly explore and account for the customer's unique security requirements and needs. |

| | |
|---|---|
| | **3.2** SIS design methods should provide modeling support at the three levels of ISD (organizational, conceptual and technical) for abstract representation and operations for specifying the three essential elements of secure systems: threats, objects, and security features (safeguards or controls).. **3.3** Any SIS design method must provide for its integration into ISD methods. **3.4** SIS design methods should be adaptable to the ISD methods selected by developers according to their own judgment and preferences. **3.5** SIS design methods should be adaptable to forthcoming ISD methods. |
| **4. Testable SIS design method hypotheses: (a) validity of the method meta-requirements; (b) fidelity of the design method to meta-design principles.** | **4.0** (a) Successful SIS design methods will include a process to protect IS security objects from IS security threats through IS security features. **4.1** (a) Successful SIS design methods include processes and notation for extensively acquiring and documenting security requirements. (b) It is feasible and practicable to start the SIS development with the customers' requirements. **4.2** (a) Successful SIS design methods offer modeling support at the three levels of ISD (organizational, conceptual and technical); (b) It is feasible and practicable to develop a SIS design method to offer modeling support for threats, objects and security features at the three levels of abstraction. **4.3** (a) The more consistent and concordant the processes and notation of a SIS design method is with common ISD methods, the more successful the SIS design method will be in practice. (b) It is feasible and practicable to embed SIS design methods into ISD methods. **4.4** (a) The more consistent and concordant with a wide variety of ISD methods the processes and notation of a SIS design method is, the more successful the SIS design method will be in organizations. (b) It is feasible and practicable for developers to adopt the SIS design methods without constraining their ability to adopt appropriate ISD methods. **4.5** (a) The more the processes and notation of a SIS design method adapts to unpredicted new ISD methods, the more successful the SIS design method will be in organizations. (b) It is feasible and practicable for developers to adopt the SIS design methods to innovative new ISD methods. |

## *SIS Design Theory: Meta-Requirements*

We will discuss these six meta-requirements (MR) below, and relate them to the kernel theories, meta-design principles and the two major sets of hypotheses. Our aim is not to present the exhaustive set of MRs for SIS design methods, but rather to first outline some essential MRs for SIS and to demonstrate the applicability of design theory in SIS.[4]

**MR 0: SIS design methods must develop security features that resist the impact of threats on system objects.**
This essential MR provides the distinctive goal that distinguishes the SIS design methods from other systems development methods with which the SIS design methods must interoperate. SIS design methods must provide mechanisms to identify security

[4] These six meta-requirements were gleaned from the literature because they operate across most SIS design methods. For example, another meta-requirement is support for user participation. However, very few security methods are explicitly participative. Many of these more narrowly focused requirements can be subsumed under MR 3 in the model *"A SIS design method should support its integration into normal ISD methods."*

safeguards, which are IS security features that protect IS security objects from threats in various ways (Baskerville, 1989). In a certain sense, this is the "identity" meta-requirement that makes the outcome of the design a SIS design method as opposed to any other form of ISD method.
**Kernel Theory:** Clements' theory (described in the previous section).
**Meta-Design Principle:** SIS design methods should incorporate security features that resist the impact of threats on system objects.
**Hypotheses:** (a) Successful SIS design methods include a process to protect IS security objects from IS security threats through IS security features.

**MR 1: Development of SIS originates in a customer's security requirements.**
A customer's security requirements must be the point of departure when developing SIS. Organizations must ensure that valuable assets (objects) are correctly protected from possible security threats through the introduction of IS security features. Since organizations' assets, business objectives, and operating environments differ, their IS security requirements, ultimately, differ (Baskerville, 1993). Certainly external factors, such as legislation, will have impacts on the development of SIS; however, the customer ultimately decides to what extent any external factors will matter in the development. (In an opposing situation, the customer has little or no power to decide the SIS requirement.)
**Kernel Theory:** Hume's thesis of *"no ought from (an) is"* (Hume's law) means it is illogical to infer "ought" (what an organization should do in terms of security) from "is" (what it is possible to do in terms of what security solutions exist or what others have done to secure their systems). Using Hume's terms, "is"-matters include facts and observations, such as certain security solutions known to have worked. Such knowledge is based on available evidence and situated in a certain environment, and may be useful in the development of SIS. However, such facts (or "is"-matters) cannot logically drive the design. Consequently, we should start the design with the organizational security requirements.
**Meta-Design Principle:** SIS design methods must provide explicit and thorough support for capturing and accounting for the customer's unique security requirements and needs: "What are the valuable assets in an organization that must be protected from IS security threats by introduction of suitable security features?"
**Hypotheses:** (a) Successful SIS design methods include processes and notation for extensively acquiring and documenting SIS requirements. (b) It is feasible and practicable to start the SIS development with the customers' requirements. In particular, SIS design methods shortcut the requirements processes and lead the designer to blindly implement certain predefined and generic security requirements.

**MR 2: SIS design methods should provide abstract representation and operations for specifying the three essential elements of secure systems - threats, objects, and security features (safeguards or controls) - for the three levels of abstraction: *organizational*, *conceptual,* and *technical*.**
SIS design methods should offer modeling support to model at least threats, objects, and security features for the three levels of IS. SIS design methods should not only provide aid on certain development stages or aspects, but also offer modeling support to model at least threats, objects, and security features at the three stages of ISD and SIS development. While an ideal ISD or SIS design method should offer support for all three domains, it does not mean that every ISD project will regard all three (e.g., developers may skip one phase in one ISD project and another phase in another ISD project). MR 2

means that SIS design methods must provide support for these three stages, but not necessarily prescribe that developers need to follow every phase in every project.[5]

**Kernel Theory:** Aside from the basic practical need for designing in the abstract, this meta-requirement stems from research into IS metamodeling, which shows that IS can be divided into three levels of abstraction: organizational, conceptual, and technical (Iivari, 1989).

**Meta-Design Principle:** SIS design methods should provide modeling support at the three levels of ISD (organizational, conceptual, and technical) for abstract representation and operations to specify the three essential elements of secure systems: threats, objects, and security features (safeguards or controls).

**Hypotheses:** (a) Successful SIS design methods offer modeling support at the three levels of ISD (organizational, conceptual and technical); (b) It is feasible and practicable to develop a SIS design method to offer modeling support for threats, objects, and security features at the three levels of abstraction.

**MR3: SIS design methods must be able to integrate to ISD.**

Typically, SIS development and normal ISD have been separate streams of development activities (Baskerville, 1992). However, in order to build security into new systems, SIS design methods cannot operate independently of ISD methods. A SIS design method should support its integration to ISD methods.

**Kernel Theory:** MR 3 arises from the theory of developmental duality, Ockham's razor, and the thesis of ontological dependencies. Developmental duality holds that the separation of normal system development from SIS development inevitably leads to conflicting goals and functional requirements (White and Dhillon, 2005). Separate SIS design methods (i.e., those that cannot be integrated into normal IS development) should be avoided if both system security and system functionality are to be effective (Baskerville, 1992). The problem of developmental duality can be traced back to Ockham's razor, *"Plurality should not be assumed without necessity"* (Ockham, 1990 p. 93). Moreover, Niiniluoto's (1999) thesis of ontological dependency makes the integration of SIS and IS development a pragmatic necessity: If there were no IS, there would be no pragmatic need for IS security. But developers can create an IS without security. IS is an artifact (a human construct), and its security is ontologically dependent on ISD for its existence.

**Meta-Design Principle:** Any SIS design method must provide for its integration into ISD methods.

**Hypotheses:** (a) The more consistent and concordant the processes and notation of a SIS design method are with common ISD methods, the more successful the SIS design method will be in practice. (b) It is feasible and practicable to embed SIS design methods into ISD methods. SIS design methods should not promote a deferral of security considerations for treatment by separate SIS development phases after the IS has been created.

**MR 4: SIS design methods should enable the autonomy of developers.**

According to their situation, developers frequently select from a "toolkit" of methods or method fragments (Kumar and Welke, 1992; Slooten, 1996). For example, according to their judgment of the problem settings, they might choose Spiral Modeling (Boehm, 1988) and Unified Modeling Language (Rumbaugh et al., 1999) for one project and

[5] In fact, MR 4 suggests that the developers should be able to freely to select those techniques, tools and processes of ISD that they prefer to use in an ISD project.

Extreme Programming (Beck, 1999) for the next. Developers should be able to address SIS concerns with their preferred ISD methods, and SIS design methods should respect developers' autonomy[6] by enabling them to select the ISD methods they judge to be correct for the setting. This autonomy extends to organizations that may wish to standardize their in-house methods (including security). Failing MR 4 means requiring developers to switch their traditional ISD methods or forcing them to learn and use separate SIS design methods in addition to their normal methods. SIS design methods meeting MR 4 become more accessible and less costly for organizations.[7]

**Kernel Theory:** MR 4 originates from the need to give proper respect to the autonomy of humans, as recognized by Kant's rule of human dignity, and by Rawls' principle of liberty (*"liberty can be restricted only for the sake of liberty,"* Kukathas and Pettit, 1990 p. 50). This autonomy has the practical effect of improving the feasibility of SIS design for more organizations, giving this MR both ethical and practical value.

**Meta-Design Principle:** SIS design methods should be adaptable to those ISD methods that developers have selected according to their own judgments and preferences.

**Hypotheses:** (a) The more that the processes and notation of a SIS design method are consistent and concordant with a wide variety of ISD methods, the more the SIS design method will be successful in organizations. (b) It is feasible and practicable for developers to adopt the SIS design methods without constraining their ability to adopt appropriate ISD methods. More specifically, SIS design methods meeting MR 4 would make security methods available and feasible for more companies compared to SIS design methods that do not meet MR 4.

**MR 5: SIS design methods should be adaptable to forthcoming ISD methods.**

MR 5 proceeds to a degree from MR 4. If a SIS design method does not take into account the autonomy of developers, i.e., is not adaptable to the existing ISD methods, then the SIS design method is unlikely to be adaptable to forthcoming ISD methods.

**Kernel Theory:** The kernel theory behind MR 5 is emergence theory, according to which ISD methods are known to be emergent (Truex et al., 1999). Not only do new methods spring up every now and then (Jayaratna, 1994), but in practice, methods are hardly executed exactly the same way twice (Truex et al., 2000). It is difficult to predict this emergence or to allow for a predefined universal SIS design method that will match every ISD method and its permutations. Emergence is exemplified in agile or short-

---

[6] In philosophical literature, autonomy has an individualistic viewpoint. It means that people can decide freely and rationally, free from any coercion or manipulation, what to do, or what ISD method to use. Here we use the term in its plural form 'autonomy of developers' in a more collective sense, and by this we recognize that the ISD method used in organizations is hardly autonomously (in the sense autonomy is understood in philosophy) selected by individual developers. Rather the selection may be determined by quality managers, partner organizations, other developers, by superiors in general, or by decisions at a team level or division level, etc.

[7] One may argue that by appealing to autonomy, developers can ignore security issues in practice. This is a risk that comes with autonomy: autonomy increases individual freedom with the result that more responsibility is put on the shoulders of the individual (Hare, 1963). This "risk" encompasses any SIS method in two ways. First, as in this paper, SIS methods can be viewed as tools (roughly speaking). This means that developers of such tools try to ensure that the tools have certain features (the meta-requirements in our case) without regulating the use of that tool in practice in too much detail. And second, even if developers of the tools (SIS methods) regulate the use of the tool in great detail, nothing would stop the practitioners from ignoring these regulations in practice.

cycle-time methods that sometimes change and adapt from project to project (Baskerville and Pries-Heje, 2004). SIS design methods currently emerge long after their related ISD methods (Baskerville, 1993). MR 5 operates with an ideal goal of integrating security quickly and effectively into each (existing, forthcoming and unpredicted) version of an ISD method.

**Meta-Design Principle:** Successful SIS design methods should be adaptable to forthcoming ISD methods.

**Hypotheses.** (a) The more the processes and notation of a SIS design method adapt to unpredicted new ISD methods, the more successful the SIS design method will be in organizations; (b) It is feasible and practicable for developers to adopt the SIS design methods to innovative new ISD methods.

### *Analysis of existing SIS Design Methods*

Based on our earlier review of SIS design methods (Baskerville, 1993; Siponen, 2005a, 2005b), we compiled the SIS design methods into a taxonomy and constructed a meta-requirements analysis of these existing SIS design methods as summarized in Table 2. MR 0 is not represented in the table or discussed below because it is definitional, in the sense that all SIS methods must address this meta-requirement. That is, they must contain a process, according to which security features are installed aimed at ensuring that threats do not get at security objects. Methods that fail to meet MR 0 would not qualify for discussion below. Beyond this, the table indicates the extent to which current SIS design methods meet the other five meta-requirements specified by the design theory above.

**Table 2. The different SIS design methods. The sign – means that SIS design methods do not meet the particular meta-requirement, while + indicates that SIS design methods do meet the particular meta-requirement.**

| **The different paradigms** | **Examples/advocates** | **Method meta-requirements (MR)** | | | | |
|---|---|---|---|---|---|---|
| **Mainstream methods** | | **MR 1** | **MR 2** | **MR 3** | **MR 4** | **MR 5** |
| **SIS Checklists:** List the ideal and generic protection means that organizations should implement | Wood *et al*. (1987) | - | - | - | - | - |
| **SIS Standards:** Prescribes authoritative and generic SIS solutions for all organizations | BS 7799 (1993); GASSP (1999); ITSEC (1990); OECD (1996), ISF (2005); ISO/IEC 17799 (2005) | - | - | - | - | - |
| **Integrative methods** | | | | | | |
| **Information / Database Modeling Methods:** Modeling notations for expressing security constraints based on structured and object-oriented paradigms | Smith (1989); Ellmer *et al*. (1995); Pernul et al. (1998) | + | - | - | - | - |
| **Responsibility Methods:** Identification of workers' role responsibilities is key for designing security in | Backhouse and Dhillon (1996); Dhillon (1997); McDermott and Fox (1999);Sindre and Opdahl (2000) | + | - | + | - | - |

| | | | | | | |
|---|---|---|---|---|---|---|
| organizations | | | | | | |
| **Business Process Security Methods:** Add security constraints in the business processes | Herrmann and Pernul (1999); Röhm *et al*. (1998); Röhm and Pernul (1998); Röhm et al. (1999) | + | + | - | - | - |
| **The Security-Modified ISD methods:** These security methods are influenced by different IS development methods | Baskerville (1988; 1989); Booysen and Eloff (1995); Hitchings (1995); James (1996); Pottas and von Solms (1995) | + | + (only one) | + | - | - |

## Limitations of Mainstream SIS Methods

The mainstream methods are SIS checklists and SIS standards (Table 2).
**MR 1:** *SIS checklists and standards*[8] (e.g., ISO/IEC17799, 2005) attempt to capture the available SIS solutions and best SIS practices in checklists and standards (Baskerville, 1993; Siponen, 2001). The idea here is to promote authoritative and generic checklists and standards that prescribe generic SIS requirements for all organizations (Siponen, 2005b). For example, ISO/IEC17799 (2005) proposes generic SIS principles that "…*apply to most organizations and in most environments*" (BS ISO/IEC17799, 2005) In the case of checklists and standards, the needs of organizations are replaced by ideal protection requirements and techniques. Organizations are often driven to follow these with little rationale. Consequently, for both SIS checklists and standards, this means that the unique needs and problems of organizations are overlooked (e.g., Baskerville, 1988; 1993). Hence, Hypotheses 1(a) and 1(b) fail, and Principle 1 is not addressed: Checklists and information security management standards do not meet MR 1.

**MR 2:** While SIS checklists and standards offer different principles for SIS, they do not provide any modeling support (or means for conceptual modeling). Therefore, these SIS methods do not offer modeling support to the extent that satisfies both Hypotheses 2(a) and 2(b). As a consequence, Principle 2 is not addressed and MR 2 is not met.

**MR 3, 4 and 5:** All mainstream methods (SIS checklists and information security management standards) are targeted for securing a completed IS. The mainstream methods are stand-alone, separate methods, and operate, by definition, independently of ISD methods. Baskerville (1993) and Siponen (2005b) point out that not only do these SIS methods offer no guidance as to how IS developers can integrate them into ISD, but also the methods are intended to be applied to already existing or completed systems. Consequently, Hypotheses 3(a), 4(a), and 5(a) cannot succeed, and Principles 3, 4, and 5 are not addressed. The three meta-requirements are unmet. This means that these SIS design methods are more suitable for securing existing, unsecured, and static systems; and that they are not suitable for use for building new secure systems, or emergent secure systems (Baskerville, 1993; Dhillon and Backhouse, 2001). To summarize, none of the mainstream methods meet any of these five meta-requirements.

[8] One might argue that standards like ISO/IEC17799 are management standards. However, in practice, standards may drive SIS design, as organizations have a need to ensure that their IS meet the criteria presented by standards. Therefore, they are included in the analysis.

### Limitations of Integrative SIS Methods

To avoid the shortcomings of the mainstream methods, integrative SIS design methods have been developed that coordinate, SIS design more closely with the social organization, the essential ISD, or the organizational (business) goals. In Table 2, these methods are classified in terms of different method families (Siponen, 2005a): *information modeling*, *responsibility modeling*, *business process modeling*, and *security-modified ISD methods*.

**MR 0:** As mentioned above, all SIS methods meet this requirement as the identity requirement that qualifies them as SIS methods.

**MR 1:** Siponen (2005a) concludes that these integrative SIS design methods (the security-modified IS development, responsibility modeling, information modeling, and business process methods) take into consideration the fact that organizations may have unique SIS requirements, and accordingly, propose different techniques to discover these requirements. The security-modified ISD methods use requirements analysis, influenced by ISD methods, to capture the organizations' SIS requirements. Responsibility methods model responsibilities in organizations, from which organization-specific SIS requirements are derived. Therefore, both the security-modified ISD and responsibility methods address MR 1. Similarly, information modeling methods offer modeling notations for expressing SIS constraints, based on the organization's unique SIS requirements, and hence meet MR 1. Also, business process methods address MR 1 by providing tools to model organization-specific SIS constraints in relation to business processes.
Hence, all of these integrative SIS design methods satisfy Hypotheses 1(a) and 1(b), address Principle 1, and meet the MR 1.

**MR 2:** Within the *security-modified ISD methods*, the Logical Control method (Baskerville, 1988) adds SIS controls to Data Flow Diagrams. Similarly, the SIS Spiral (Booysen and Eloff, 1995) addresses access control issues in data flows. Both the Logical Control and the SIS Spiral methods provide conceptual level modeling support. The Virtual Methodology (Hitchings, 1995) provides a notation at the organizational level to model authorized and unauthorized relations. The risk analysis process of SIS planning methodology (Straub and Welke, 1998) is at the organizational level (Siponen, 2001).

*The information modeling methods* offer notations to extend Entity-Relationship (ER) and data flow diagrams to cover SIS requirements or constraints. Thus, the information modeling methods provide organizational- and conceptual-level modeling support. The business process paradigm offers modeling support at all levels of abstraction, from SIS requirements capture to implementation and coding issues.

*The responsibility modeling methods*, including the responsibility method proposed by Strens and Dobson (1993), the semantic responsibility method (Backhouse and Dhillon, 1996), the task-based authorization method (Thomas and Sandhu, 1994), the abuse case (McDermott and Fox, 1999) and misuse cases (Sindre and Opdahl, 2000), provide conceptual-level modeling support. The exception is MAPS (Pottas and von Solms, 1995), which provides modeling support at all three levels of abstraction. Of the integrative SIS design methods, only MAPS and the business process methods provide

modeling support that satisfies the dictate of Hypothes 2(a). Therefore other SIS design methods fail to address Principle 2 and do not meet MR 2.

In the case of the *business process security methods* (Herrmann and Pernul, 1999; Röhm et al., 1998; Röhm and Pernul, 1999) and the SIS design method by Pottas and von Solms (1995) (which offers modeling support at the three levels of ISD), Hypotheses 2(a) and 2(b) are satisfied. They also address Principle 2 and satisfy MR 2.

**MR 3:** Of the integrative SIS design methods, the Logical Control method (Baskerville, 1988, 1989) can be integrated into structured ISD methods, data flow, and ER diagrams. The SIS Spiral method (Booysen and Eloff, 1995) can be integrated into methods that utilize Boehm's spiral model. The abuse (McDermott and Fox, 1999) and misuse (Sindre and Opdahl, 2000) case methods can be integrated into use case notations in the requirements analysis phase. These SIS design methods satisfy Hypotheses 3(a) and 3(b), addressing Principle 3 and satisfying MR 3.
While the information modeling methods do not address their integration into ISD, the security constraints modeling method (Pernul, 1992) might be integrated to structured methods (Data flow and ER diagrams), and object-oriented security semantics (Ellmer et al., 1995) might be integrated into object-oriented ISD methods.[9] The other integrative SIS design methods are difficult to embed into existing ISD methods and offer no guidance as to how they might be integrated into ISD (Baskerville, 1992; Siponen, 2005a,b). Additionally, the fact that their notations differ from ISD methods is another obstacle to the integration of these SIS design methods into the ISD methods. Thus, the other integrative methods fail Hypotheses 3(a) or 3(b), do not address Principle 3, and fail to satisfy MR 3.

**MR 4:** Although, the logical control method (Baskerville, 1988, 1989), the spiral method (Booysen and Eloff, 1995), and the abuse case (McDermott and Fox, 1999) and misuse case (Sindre and Opdahl, 2000) methods satisfy MR 3, these three SIS design methods are focused on a single ISD method (UML) or a narrow range of ISD methods. For example, the logical control method (Baskerville, 1988) discusses its integration into structured methods and data flow diagrams in particular, but not into other ISD methods. Similarly, the spiral method (Booysen and Eloff, 1995) concerns its integration into Boehm's spiral, and the abuse and misuse case methods concentrate on use cases. As a result, none of the SIS design methods satisfy Hypothesis 4(a), and consequently, Hypothesis 4(b) also fails.

All of these SIS design methods restrict the autonomy of developers in selecting the ISD methods according to their judgment. None of the methods addresses Principle 4 or satisfies MR 4. Such restriction on the integration of these SIS design methods into ISD methods can be explained by the fact that these methods were explicitly designed as a part of certain ISD methods, such as Boehms' spiral, or their specific notation, such as DFDs and use cases. They were not designed to be generic SIS components that can be integrated into different sorts of ISD methods.

**MR 5:** Since the existing SIS design methods constrain the autonomy of developers (failing MR 4), it is unlikely that Hypothes 5(b) will ever succeed for these SIS design

[9] Such integration may be possible, since these methods are based on Data Flow and ER diagrams, and object-modeling.

methods (or at least without modifications to the SIS design methods). Principle 5 is lost, and MR 5 remains unsatisfied.

**Summary.** In light of the meta-requirements of the SIS design theory, none of the current SIS design methods address MR 4 and MR 5, only four of the existing SIS design methods meet MR 3, and in addition to business process methods, only MAPS (Pottas and von Solms, 1995) from among the security-modified IS development methods meets MR 2. To overcome these limitations, we developed the following solution to address these meta-requirements.

### *Proving a SIS Design Theory That Addresses the Six Meta-requirements*

Research into design theory has frequently been limited to the abstract theoretical realm, as has the work above so far. In these cases, the application or "proof" that the design theory operates empirically is deferred to future research (e.g., Walls et al., 1992; Kasper, 1996). Our research goals were primarily focused on the establishment of a major advance in SIS design methodology that proved usable in the field. An empirical field study of design theory entails (1) the development of a SIS method, in our case called Meta-notation, which reflects the design theory, and (2) the application of the resulting methodology in practice. While such studies will not demonstrate universal validity for this design theory, the studies will demonstrate that the theory operates in known settings.

The *first* part of the field study utilizes conceptual-analytical and constructive research to develop a Meta-notation attuned to the SIS design theory. The SIS design problem, in light of the meta-requirements, is abstracted by following the ideas of method engineering (Tolvanen, 1998) and metamodeling (Gigch, 1991). From the perspective of Meta-notation, we infer from the academic literature a general model of SIS design method notation comprising six essential elements. The *second* part of the field study uses clinical fieldwork methods (Schein, 1987) and action research. Figure 4 illustrates the two-stage fieldwork for "proving" the design theory in use.

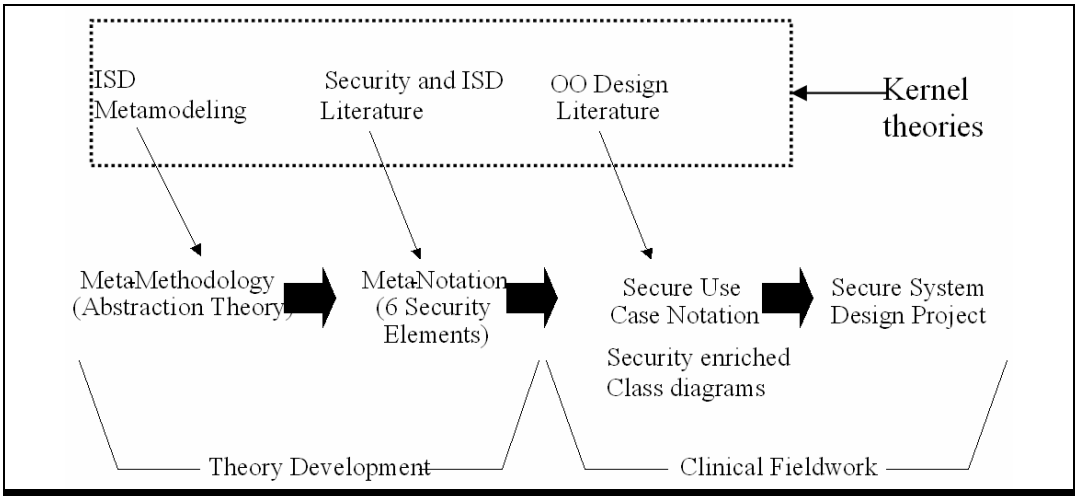


**Figure 4. The development of a SIS design method, namely Meta-notation, to address the six SIS meta-requirements. The figure describes the theory development and clinical field work phases. It also shows the kernel theories in the sense of Walls *et al*. (1992) used to develop the Meta-**

**notation. The kernel theories include ISD metamodeling and modeling in ISD methods, as well as access control models in the field of computer security.**

## The Development Of A SIS Method: Meta-Notation

A comprehensive ISD method includes a process system (guiding how the development is carried out) and a notation system (Hirschheim and Klein, 1992). The three higher-order meta-requirements (MR 3, 4, and 5) indicate a meta-method solution. In order for a design method to be integrated into existing (as required by MR 3) or future (as required by MR 5) ISD methods, SIS design theory demands a higher level of abstraction than existing SIS design methods. Meta-notation involves schemes for the creation and assembly of methods and method fragments. These schemes are of particular interest to method engineering and metamodeling (e.g., Alderson et al., 1998; Brinkkemper et al., 1996; Brinkkemper et al., 1999; Gigch, 1991; Hillegersberg and Kumar, 1999; Nuseibeh et al., 1996; Slooten, 1996; Tolvanen, 1998). Meta-method tools, such as MetaEdit+, provide the means for rapidly developing computer-aided systems analysis and software engineering (CASA/CASE) tools to match development needs and settings (Marttiin et al., 1996).

### *Regularities in ISD and SIS*

Moving to a meta-method level of abstraction yields a perspective of ISD methods that reveals how these are in a constant state of emergence and change. Since ISD settings may be different, ISD methods are often modified to suit different contexts (Mathiassen and Purao, 2002; Truex et al., 2000). However, systems problem settings and development approaches are not in total chaos and relativism. That is, different development situations are not totally disparate. Otherwise, our practical experience with past ISD would have no value for future ISD tasks. Experience helps us recognize regularities in the way problem settings arise and methods emerge. In order to satisfy the three higher-order meta-requirements of our design theory, our meta-method must discern a sufficient range of these regularities in order to be useful.

We used an analytical process for discovering regularities in SIS design elements. This process is described in detail below (Figure 2). First, we looked across ISD and SIS design methods in order to find common core concepts (*subjects* and *objects*). Second, we surfaced the regularities in existing SIS design methods (e.g., Baskerville, 1993; Siponen, 2005a, b) resulting in four additional concepts (*security constraints*, *security classifications*, *abuse subjects and abuse scenarios*, and *security policy*). Finally, we consulted a panel of practitioners for comments about the patterns. This process led to a notation with six elements. While some computer security researchers, particularly cryptographers, distinguish other security elements (e.g., Menezes et al., 1999), the six described below proved sufficient for our purposes to capture most of the common regularities found in SIS development. Additional elements can certainly be added to the Meta-notation on an ad-hoc basis as required.

### *Subjects, Objects, and Classification*

In seeking regularities in ISD methods, notation systems quickly rise in their apparent regularity across many methods and problem settings. More than a thousand different ISD methods are reported to exist (Jayaratna, 1994) with many differences, for example in terms of sociological paradigms (Hirschheim and Klein, 1989), and ontological expressiveness (Wand and Weber, 1993). Yet there are commonalities among these ISD methods, of which two are relevant for our purposes. One objective common to ISD methods is to help developers to understand and model reality with respect to the system-to-be-built (Hirschheim et al., 1995; Iivari, 1989; Lyytinen, 1987). In order to do such analysis and modeling, developers need to agree on how to describe things in different levels of abstraction (organizational, conceptual and technical, see Iivari, 1989). Traditionally, things are classified as 'entities' (e.g., structured methods) or 'objects' (e.g., object-oriented methods) in the ISD literature (Wand et al., 1999). Additionally, some ISD methods use 'actors' (e.g., the UML type of use case diagram) or 'stick figures' (e.g., Checkland's rich picture) at the organizational level (e.g., requirements analysis stage).

Any SIS design method that aims to integrate itself with ISD methods needs to address how the security aspects can be described in term of entities common to ISD methods, such as actors. This may happen, for example, by mapping the SIS design method notation schema to these ISD entities. Thus, given that there exist certain common things in ISD methods (e.g., objects in object-oriented methods, or entities in structured methods), and that we can map SIS notation to these entities, we are able to address the six SIS meta-requirements.

One simple solution addresses this issue. In Meta-notation, the 'actors', 'entities', or 'objects' are classified as security subjects and objects. This is consistent with Clements' theory that establishes the identity of SIS methods. As a result, notational dimensions for security subjects and objects can, in principle, be added into any of the many notational patterns that use actors, entities, or objects as units of analysis. For example, both the structured and object-oriented ISD methods (Lyytinen, 1987) are good candidates for the addition of such security notation. Security subjects denote the different security-relevant entities, i.e., entities that have a relevant security connection to the assets of the organization (security objects). Security subjects may include automated processes, network nodes, employees of the organization, business partners, and third parties. Security subjects include (where they have a security-relevant connection to the assets of the organizations) actors in use cases and stick figures in rich pictures.

The term 'security objects' refers to the assets of the organization that are of relevance in terms of information security. Such assets (security objects) may range from physical things such as paper to electronic entities such as files. The terms 'security subjects' and 'objects' are used in database security literature to describe computer access control policies and models (e.g., Castano et al., 1995; McLean, 1990; Sandhu, 1993). These access control policies and models from the field of database security in turn influence computer security classification schemes. Security classification stems from the need to classify security objects and subjects according to their information security sensitivity.

The abuse subject is a special class of security subjects referring to those that may carry out a security violation. The introduction of *abuse subjects* and *abuse scenarios,* influenced by McDermott and Fox (1999), may be needed for two kinds of situations. First, they may come in handy when there is a need to explore and identify potential threat scenarios. Second, they are relevant for testing purposes. They help to check that

the IS and software under design can cope with unwanted scenarios or attacks by unauthorized people or processes through security features.

### Security Constraints

Security constraints are the rules being enforced in relation to security objects. These may include write access, read access, etc. We can discover security constraints by analyzing the security requirements (confidentiality, integrity, availability, and non-repudiation) for each security object. The theoretical background of the constraints concept stems from "security requirements." In the security literature, there is wide agreement on four security requirements (e.g., Parker, 1981, 1998): confidentiality or non-disclosure (prevent/detect improper disclosure of information), integrity (information should not be modified by unauthorized personnel), availability (information should only be available to authorized personnel), and "non-repudiation" (Chokhani, 1994; Röhm et al., 1998). Non-repudiation is important for enabling financial transactions and data interchange, which are keys to electronic commerce. It is needed to make valid large-scale contracts between vendors and customers. Non-repudiation means that a contracting party cannot subsequently deny its actions. These constraints can be derived from customers or from their information security policy.
In summary, the Meta-notation includes six dimensions: 1) *security subjects*, 2) *security objects*, 3) *security constraints*, 4) *security classifications*, 5) *abuse subjects and abuse scenarios*, and 6) *security policy*. These dimensions are optional depending on the exact security setting.

### A Meta-notation Illustration

To clarify the operation of this Meta-notation, we illustrate how it can be applied using a simple demonstration of a use case notation and a class diagram common in object-oriented ISD methods. We chose use cases for demonstrating the Meta-notation because most object-oriented methods employ use cases in the requirements analysis phase to collect and analyze requirements. Use case notation demonstrates what a user should be able to do with the system; use cases are graphical or textual presentations of the usage of a system (Jacobson et al., 1992).
Figure 5 is a traditional textual use case description. The use case is 'booking'.

*Use case:* Booking.
*Version:* 1.0
*Functional Summary:* A booking clerk books journeys for customers.
*Frequency:* several times a day
*Usability requirements:* Any database query and booking must be able to complete in less than 30 seconds.
*Actor:* a clerk.
*Preconditions:* Booking and customer databases exist.
*Exceptions:* If information on certain journey is not available an appropriate error message is produced.

**Figure 5. A traditional use case (adapted from Jaaksi, 1998).**

Now presume that there is a need to address security considerations in ISD. Figure 4 describes the use case presented in Figure 3 enriched by five security dimensions of the

Meta-notation. The normal use case semantics are presented in italics, while the security semantics are illustrated in italics and boldface. The example demonstrates an implementation of a security policy that states booking clerks can only access objects labeled as confidential within the booking department. The security policy constraints are expressed in the Meta-notation enriched use case in Figure 6 as the Security policy/Specific security restrictions.

*Use case:* Booking.
*Version:* 1.0
*Functional Summary:* A booking clerk books journeys for customers.
*Frequency:* Several times a day
*Usability requirements:* Any database query and booking must be able to complete in less than 30 seconds.
*Actor/**security subject:*** A clerk.
***Security classification of the subject:*** confidential
***Security objects and access types to security objects:*** Object: customer file (the clerk must be able to read, update and delete the customer information); Object: booking database (the clerk must be able to read, update and delete the customer information on the database)
***Security policy/Specific security restrictions:*** The clerk is only allowed to access security objects classified as confidential with the booking department.
*Preconditions:* Booking and customer databases exist. The identity of the booking clerk/security subject has been validated.
*Exceptions:* If information on a certain journey is not available, an appropriate error message is produced.

**Figure 6. A Meta-notation enriched use case.**

The security subject in Figure 6 is the booking clerk. The clerk's security classification clearance is confidential (classification of security subjects). The relevant security objects with respect to this use case are the customer file and the booking database. The types of access to these objects are create, read, delete, and update.

Continuing this example, we can illustrate how abuse cases are connected to respective use cases. Abuse cases, like the other five security elements, are included only as deemed relevant by the designers. Figure 7 describes an abuse case associated with the booking use case, including possible abuse scenarios, assumed frequency of the action, presumed abuse subject, and the security object that is the target of that abuse.[10] There are also total costs of recovering from the abuse ($ 1500). The cost information, together with the frequency of the abuse scenario, is useful for prioritizing the abuse scenarios.

[10] Abuse subjects are not applied to normal use cases, as they do not concern normal use, but unauthorized use.

*Abuse case:* Booking.
*Version:* 1.0
*Possible abuse scenario:* A malicious user carries out a denial-of-service attack halting (use case) booking, i.e., halting a clerk in booking journeys for customers.
*Frequency:* Several times a day
***Abuse subject:*** A malicious user
***Security objects/target of abuse:*** Customer file, booking database.
***Total costs of recovering:*** $ 1500

**Figure 7 presents an abuse case scenario (cf., McDermott and Fox, 1999).**

Figure 8 shows the security level of each class. The security levels of classes are added into the class diagram. The user in Figure 8 is a security subject, while document is a security object.

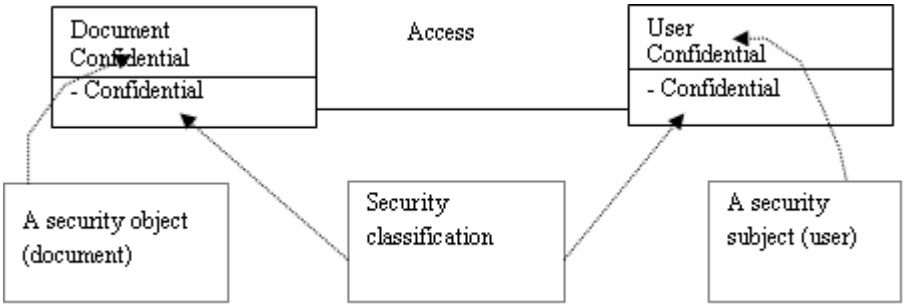


**Figure 8. A Meta-notation enriched class diagram.**

## Research Approach

While the conceptual analysis and constructive research used to develop the design theory and Meta-notation provide a degree of logical rigor, fieldwork is needed for an empirical validation of the concepts. The most effective means for empirically validating ISD or SIS design methods entails action research, in other words, researchers must *"go into the world and try them out"* (Wood-Harper, 1989). Proper evaluation of a design theory requires that we apply the resulting artifacts in their intended settings, providing empirical evidence that the prescribed means are a promising route to the desired ends. Therefore, we designed action research to apply the design theory and its proposed Meta-notation in actual practice, and to evaluate the results.

Action research is an interventionist and interpretive method (Klein and Myers, 1999; Walsham, 1996). From the perspective of action research, theories are validated through successful use in solving practical problems; successful use being defined through the social reflection of the collaborators in the research. Action research has also been demonstrated as a valuable research strategy for investigating ISD methods (Avison *et al*., 2001; Baskerville and Pries-Heje, 1999). In fact, it has been argued that action research is ideal for studying IS methods in a practical setting because it expands scientific knowledge while simultaneously putting theories to work in practice to help participating organizations solve concrete problems (Baskerville and Wood-Harper, 1998).

Action research is typically represented as an iterative process of theory-based diagnosis followed by practical interventions in the form of action planning, action taking, and evaluation in a field setting (Figure 9). Based on the specific learning that arises from evaluation, the theory is changed (if necessary) and the cycle repeats until a theory is developed that leads to satisfactory problem resolution.

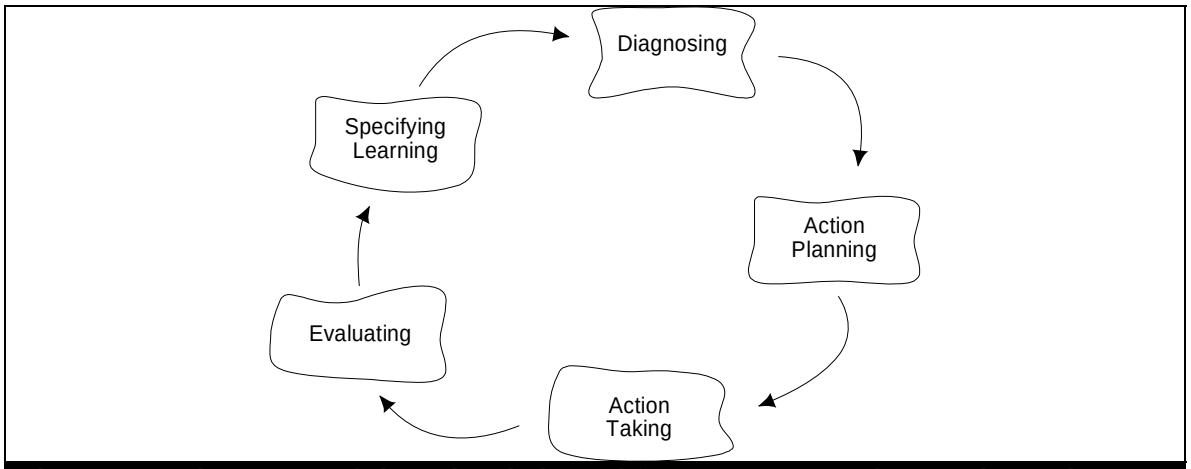


**Figure 9. Action research iteration (adapted from Susman and Evered, 1978).**

Our research used staged action research methods involving two stages and three iteration cycles (Figure 10). Stage one was a preliminary study that followed a form of action research called "clinical fieldwork" (Schein, 1987). Clinical action research is distinguished by its precise focus on the organizational problem and expectations for a single pass (iteration 1) through the action research cycle. Based on the results of the preliminary study, we proceeded to an expanded action research setting (Stage 2) that involved two iterative cycles. The outer iteration 2 operated at the level of the software development organization, while the nested inner iteration 3 operated within a particular ISD project. All three action research iterations involved introducing the Meta-notation into the organization as a solution to poor security development methods. In each iteration, the involved actors appraised the results through social reflections.

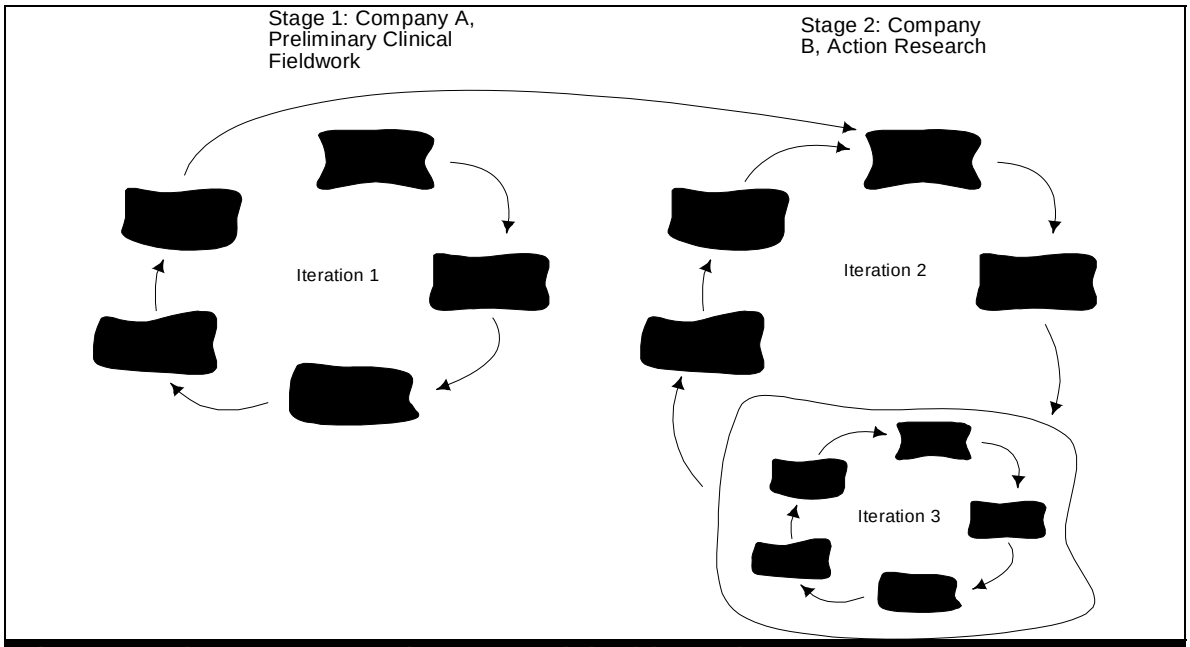


**Figure 10. The stages of action research in this study.**

Appendix C summarizes the criteria for evaluating such action research designs and discusses how the features of this design satisfy these criteria.

## Preliminary Clinical Fieldwork

Following the development of the Meta-notation from the design theory, we sought a software development organization with SIS development problems. Company A was involved in building a web-based database for films and movies that raised serious security concerns in protecting the intellectual property.

Working with one of the software companies engaged by Company A, the researchers helped introduce the Meta-notation into the existing modeling notation. We recorded the experiences, opinions, and evaluations of the practitioners in interview sessions and reviewed design documents. The Meta-notation operated as predicted, and the modified methods were applied in designing and building the Company A software. In the opinion of the participants, the Meta-notation worked as expected via the design theory.

Clinical researchers would advocate that this single loop is sufficient to validate the underlying theory in practice (iteration 1 in Figure 8, see Appendix A for further details on the preliminary clinical fieldwork). While the generality of the design theory will not necessarily be improved by additional action research, we can increase the confidence in the design theory by expanding the application to other settings. Accordingly, we proceeded to develop a full action research study to introduce the design theory and the Meta-notation into a more complex design setting.

## Proving Meta-Notation In Company B

Company B is a large software house that develops customized systems and offers consulting services in all aspects of IS and software development. It uses an in-house

ISD method, sells this ISD method as a product, and provides related consultation services. The most important clients of the company include government agencies, insurance companies, banks, and industrial companies.

A chief developer, responsible for security matters in ISD, contacted one of the authors. The chief developer believed that the company's ISD method didn't offer adequate support for the SIS aspects of ISD. As a result, this author worked with the chief developer to explore the possibility of an action research project that would integrate relevant security aspects into their ISD processes. This meeting led, in turn, to a meeting among the chief developer, the manager of the ISD department, and the development manager of the company. After reviewing the Meta-notation, the chief developer and the two managers sought to set up the action research project for the purpose of integrating the Meta-notation into company's ISD method and subsequently move the result into practice. We present the implementation of the research phases (iteration 2 in Figure 10) and the schedule in Figure 11.

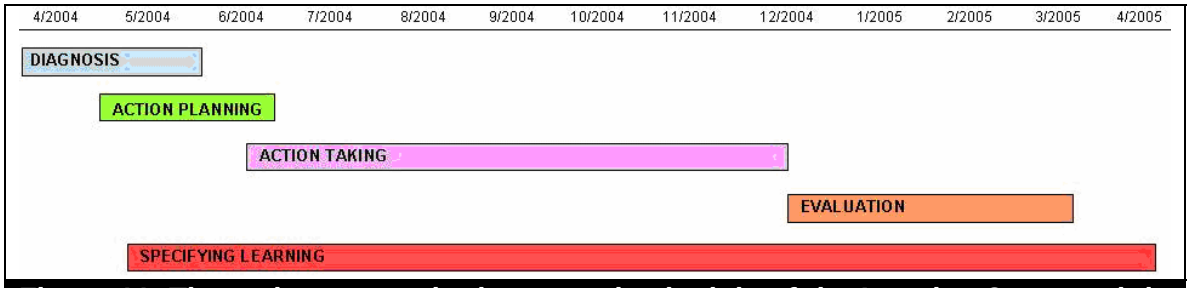


**Figure 11. The action research phases and schedule of the Iteration 2 research in Company B.**

## *Iteration 2 Research in Company B*

The action research team for iteration 2 consisted of two researchers, the chief developer, two managers, and several software developers. The researchers integrated Meta-notation into Company B's ISD method, which was based on the Rapid Application Development (RAD) process and UML notation. We integrated the six security elements of the Meta-notation into the process documents through three integration iterations with close cooperation between researchers and practitioners. The practitioners on the team were consulted to ensure that the integration was successful in each phase. After the integration phase, the method was evaluated empirically. We describe these iterations and the empirical evaluation next.

***Diagnosis:*** *identification of problems in the company's existing ISD practice*

While Company B does not apply any particular SIS method, it does use its own proprietary information systems security checklist in ISD. The chief developer felt that, at present, the feasibility of adding security features into IS or a software product rests on the competence of individual developers and designers. In particular, he called for a systematic and holistic SIS method that would be integrated into all stages of the company's existing ISD method: The chief developer asked,*"How can information security features be handled in every stage of the ISD process, because* [in a large

organization such as Company B] *the developers in the different development phases are different persons?"*

The chief developer noted that misunderstandings between different developers (system analysts, designers, programmers and testers) are too frequent. He wanted a SIS design method that allows the developers to communicate with each other using the same concepts throughout the different phases of ISD. Such problems demanded a methodological solution in which SIS issues are integrated as a normal part of the ISD method.[11] Finally, he stressed that such a SIS design method should be as easy to use as possible.

Both the chief developer and the managers felt that the company's proprietary method (a variant of the RAD process) would be a good choice for this research project, since RAD teams are small. With small teams, the intervention would be easier to study and control. RAD projects are shorter than traditional projects, so the results are achieved more quickly.

To obtain a deeper understanding of the problems of SIS in relation to the company's ISD method, we interviewed four carefully selected practitioners: a method practitioner, a business manager, a software architect and the manager of the ISD department. Because Company B also sells its own ISD methods and related consultation services, these practitioners believed a Meta-notation enriched ISD method would be extremely relevant. The practitioners also pointed out three specific SIS problems in ISD: (1) the lack of a systematic approach to address SIS issues, (2) outdated and separate SIS instructions that were not part of the ISD methods and (3) wasted resources (security is added too late, costing money and resources, either directly or indirectly). The following comments illustrate these problems:

*"The method used lacks a systematic approach to security issues,"* – software architect.
*"Security instructions have so far been incomplete…The instructions are too narrow,"* – method practitioner.

All the practitioners maintained that *"security instructions have, up until now, been incomplete. We have had some instruction, but now* [during the research project] *the instructions are finally going to be developed further."* In addition, the business manager said that adding security afterward and separately from the ISD process leads to waste of resources, and, *"In some projects, there is not enough knowledge of information security, so the responsibility for adding security rests on the auditors' shoulders. Unfortunately, the auditors often lack the necessary information security skills."*
The practical diagnosis in this setting aligned well with the drivers underlying the design theory, e.g., outdated SIS instructions, lack of SIS method resulting in waste of resources in ISD, etc. The loyal adherence to an in-house ISD method that lacked security considerations further aligned the setting to the design theory. We proceeded

[11] According to the chief developer, where there are different developers in different phases of ISD, either all developers need to have security expertise (otherwise it is likely that the SIS issues may be taken into account in one stage, but not in others), or the ISD method has to provide SIS features, leading to a situation in which the SIS issues are "naturally" taken into account in each phase of ISD.

with planning for the integration of the Meta-notation into the proprietary software ISD method of Company B.

***Action planning:*** *planning the integration of the Meta-notation*

We scrutinized the company's in-house ISD method and the RAD process to see how SIS Meta-notation could be integrated into the RAD process. The RAD process involves 44 process documents including functionality specification, class description, architectural designs, test plan, and reports. These process documents are created by the ISD department of Company B, to inform Company B's developers how to use the RAD process (documents are needed to ensure that all developers use consistent notations and concepts). For example, a class description document describes the methods and attributes of classes and explains how to make class diagrams.

In planning, we realized that part of the Meta-notation dimensions could be easily integrated into these process documents. The RAD process documents included actors that can be characterized as security subjects. Also, the concept of the "information entities to be processed" in these process documents can be characterized as security objects, while information processing rules can be regarded as security constraints. Security classifications, security policy, abuse scenarios, and abuse cases proved more difficult to integrate into the process documents because there were no obvious equivalent structures in the methodology. Nevertheless, the in-house ISD method was expanded (or enriched) with these new security dimensions.

After analyzing all 44 process documents, one of the authors made four Meta-notation enriched process documents by integrating the security dimensions of the Meta-notation to four requirement specification documents. The purpose of these documents was to demonstrate how the integration of the Meta-notation into the RAD process affects the original RAD process and how smoothly SIS aspects can be integrated into these documents.

While the company was satisfied with the four process documents in general, they highlighted issues that needed critical attention during the integration process. The chief developer stressed that the whole ISD process must be covered. Moreover, the ISD department manager stressed that *"It would be very useful to develop a document which can be used to demonstrate the security issues and risks to the clients' managers."* In that way, the company's clients would also obtain a general picture of the system's risks and security requirements.

The chief developer and the ISD department manager noted the promising nature of both the analytical and more casually expressed anticipation that the process of integrating the Meta-notation's security dimensions into their in-house method would yield a solution. The team agreed to continue the research process by integrating SIS dimensions into the remaining process documents.

***Action taking:*** *The integration of the Meta-notation into the RAD process*
The integration of the Meta-notation took place in a nested series of action research iterations (Iteration 3 in Figure 10). The action research team for this part of the research included one researcher, an architect, a subject matter expert in the application domain, and an engineer. This part of the research ran the inner action loop cycle through three iterations. The practitioners in Company B evaluated the results of the integrations after

each iteration. The process documents modified during the integration process are listed in Table 3. We discuss the details of each of these iterations in more detail in Appendix B.

Table 3. Meta-notation enriched documents by development stage.

| *RAD process documents* | *Iteration 1* | *Iteration 2* | *Iteration 3* |
|---|---|---|---|
| Requirement specification | 4 documents | 5 documents | 6 documents |
| Requirement analysis | 2 documents | 5 documents | 5 documents |
| Architectural verification | 5 documents | 5 documents | 5 documents |
| Design and implementation | 3 documents | 6 documents | 6 documents |
| Testing | 2 documents | 3 documents | 3 documents |
| New documents | 0 documents | 3 documents | 0 documents |
| **Total** | **16** documents | **24** documents | **25** documents |

**Development of the process model**

We present the Meta-notation enriched RAD process model that emerged from the action taking iterations, based on the original RAD process model in Figure 12. The Meta-notation enriched RAD project also takes clients' SIS requirements as one of the points of departure for the ISD. The information security policy can set security constraints for data processing and define the classification policy used in the client organizations. Access control mechanisms and security constraints, in terms of the Meta-notation, can be inferred from the information security requirements.

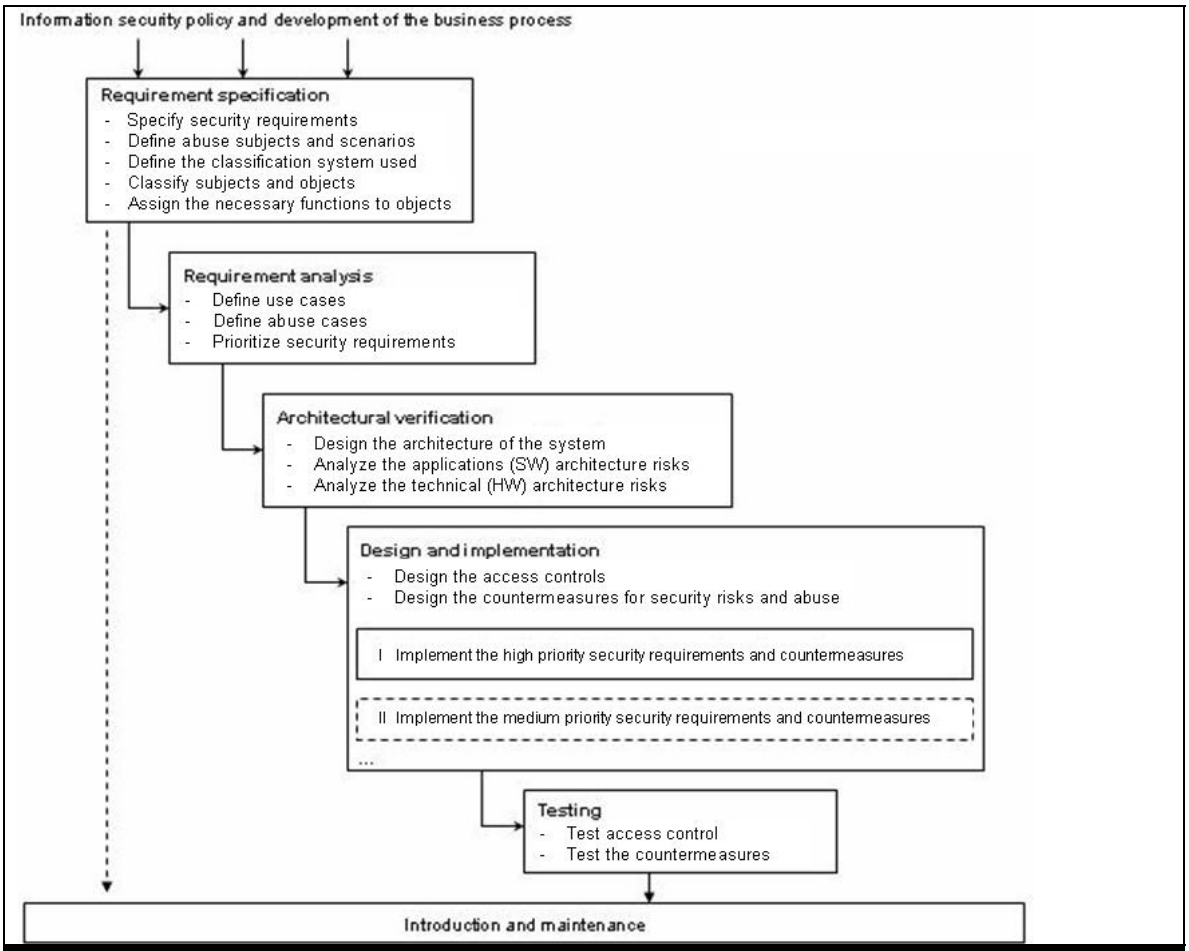


**Figure 12. The Meta-notation enriched RAD process model after integration of the Meta-notation.**

In the requirements analysis phase, the analyst defines use cases by using the security constraints and information security requirements from the requirements specification. After the use cases are created, the analyst defines abuse cases by basing these on abuse scenarios, the priority of the SIS requirements, and risk analysis.

In the architectural verification stage, the architect designs the IS architecture by analyzing the architectural risks. By analyzing these architectural risks, and by considering the alternative architectural options, the architects and designers become aware of the risks of the particular application, and can decide how these risks should be addressed.

In the design and implementation phase, the designer designs access control mechanisms and the developers implement these in order of requirements priority. The developers implement high-priority requirements before the medium priority requirements, and so on. Furthermore, countermeasures to abuse scenarios are designed and implemented in this phase. After the implementation, the access control mechanisms and countermeasures are tested.

We integrated the Meta-notation dimensions in all phases of the RAD process to ensure that the SIS requirements would be implemented as planned. After the modifications, the

practitioners stated that the process documents and the Meta-notation enriched RAD method were ready to be tested in practice.

**Evaluation:** Testing the Meta-notation enriched RAD process in practice.

We empirically tested the Meta-notation enriched RAD method in a project aimed at developing an IS that the customers of Company B could use to update their purchased software products on-line. The project resources included three experienced software practitioners (a software architect, a technical expert and a software engineer). At the beginning of the project, the project team decided not to use the security classification element, because the implementation of such a system might be too heavy for a small application. As the software architect noted, *"The security classifications may be difficult to implement as an access control mechanism. The implementation and design of such an access control mechanism may be too heavy to be done in one project."*

Instead, the practitioners implemented access control was implemented by using security constraints. From a design theory viewpoint, this decision exercises the autonomy meta-requirement, and tests whether developers adapt SIS methods in an on-the-fly, emergent manner. In this case, the classification element is excluded (indeed, it is supposed to be optional depending on the setting).

The design team began by defining three abuse subjects and six abuse scenarios to demonstrate the harm the abusers might cause. From these scenarios, the team was able to recognize several security objects, such as user passwords and the update packages. After defining abuse scenarios, the team members separately prioritized the risks regarding each abuse scenario. After discussing these risks as a team, the team decided that three of these abuse scenarios had to be taken into account in the development of the IS.

In the requirements analysis phase, the team defined the abuse cases relating to use of the system. In defining the abuse cases, they added color to the notation, using different colors in the use and abuse case diagram. They found color was effective for highlighting the SIS aspects of the IS. The team used red to highlight the abuser and related harmful actions and yellow to highlight the countermeasures that mitigated these harmful actions. White represented normal use cases (Figure 13).

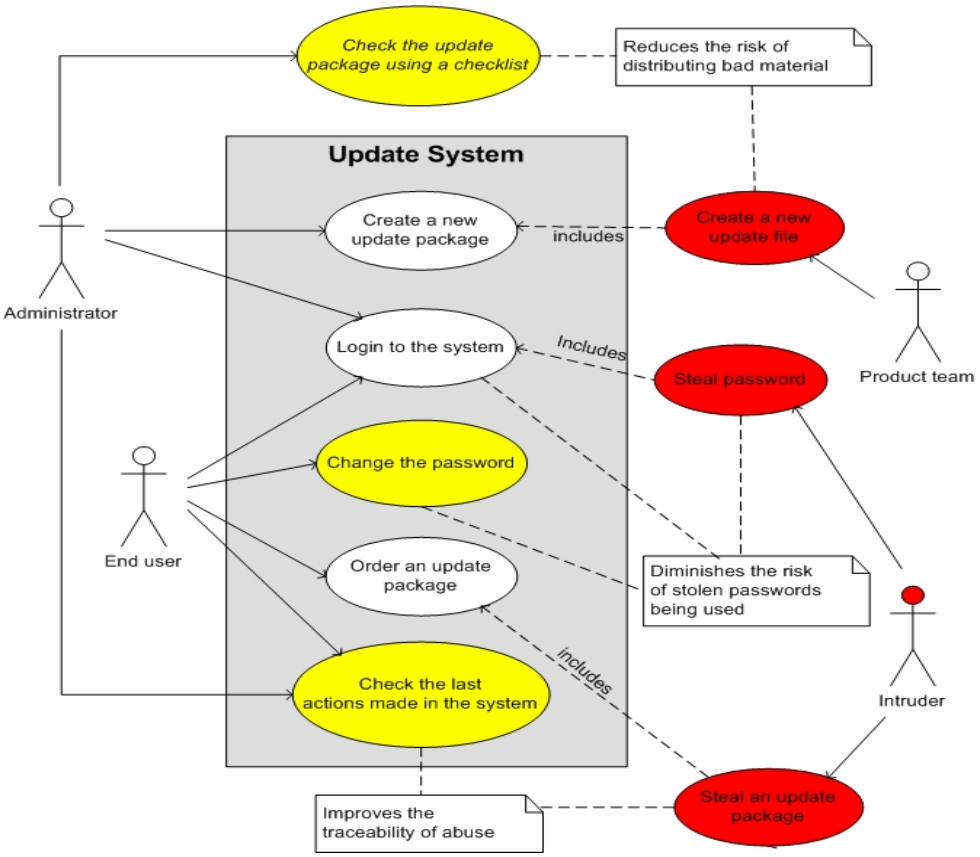


**Figure 13. A use case diagram for the IS. The product team is an example of an insider threat and the intruder represents threat from outside. Four elements are red in color: "create a new update file," "steal password,, "intruder," and "Steal an update package." Three elements are yellow in color: "Check the update package," Change the password," and "Check the last actions."**

**Specifying learning:** *Evaluation of the Meta-notation work in Company B*
We collected feedback from the experiences of the Meta-notation use in this setting through interviews with the practitioners (software architect, technical expert, and software engineer).

Did the notation help acquire and document the security requirements (Hypothesis 1(a))? Comments from the participants appear supportive. The three practitioners saw several benefits in the use of the Meta-notation. According to the software engineer:

> *"The biggest advance that comes from using the Meta-notation is that the security issues are described at the beginning of the development and that these issues are not modeled too technically at the beginning of the development process."* [In addition, they stated that the Meta-notation was a useful tool for modeling the systems' requirements.]

Do the practitioners regard the method as feasible and practicable (Hypotheses 1(b), 2(b), 3(b), 4(b) and 5(b))? Comments from the participants also appear supportive. For example, the practitioners commented that the Meta-notation was easy to learn in

general, although security classifications and terminology were difficult to understand. According to the software engineer:

> *"Regarding ease of use, it makes no difference whether one employs use cases or abuse cases, and the abuse scenarios and abuse cases fitted well in the* [RAD] *process."* The technical expert commented: *"It is always difficult to use new methods, especially when you are the first* [in the organization] *to do it, but I do not think that the use of this method is any harder than the use of some other method."* The software architect stated. *"The abuse scenarios and abuse cases felt easy to learn […]. I find the security classifications difficult to understand and to realize how they should be used."* [The other dimensions (security policy and constraints) were easy to understand and apply in practice.]

Was the modeling comprehensive (Hypothesis 2(a))? Comments from the field study appear supportive. For example, according to the technical expert:

> *"It* [the Meta-notation] *works in the modeling done in the requirement stage. One has to stretch the standards* [e.g., UML] *a little, but that is not a problem."* According to the software architect: *"The method works well in the requirements stage and helps to move the collected information to the design stage. In the design stage the method* [Meta-notation] *provides a good tool for modeling the technical mechanisms of the countermeasures to be implemented."*
> And, again, according to the technical expert: *"In those development stages where we used the Meta-notation, it worked well. Of course, the use of the Meta-notation required a bit of adapting and I'm sure that this will also be required in the future in different kinds of projects."*

Were the developers able to use the method within their preferred approach (Hypotheses 3(a) and 4(a))? the software architect noted:

> *"Generally the integration* [of the Meta-notation] *to our method was OK. … The information security policy defines different kinds of processes which are good to take into consideration during the development process. … The Meta-notation helps in describing the features which might become risks if they are ignored."*

Did the developers feel that they could use the method in future, perhaps in other preferred methods or even methods yet to come (Hypotheses 4(a) and 5(a))? Comments from the participants are generally supportive.

> All developers agree that they would consider using this information security use case notation in future, as indicated by a developer: *"Of course, with certain reserve. I would probably not use all of the six dimensions in every project. On the other hand, the dimensions* [all six] *are available when needed. It all depends on how much the client is willing to pay for security features."* Another developer commented: *"Yes, I am sure that I will be using the method in future."*

The developers found the abuse scenarios and cases to be good tools for communicating with customers and top-level managers. As one developer commented:

> *"In the requirement analysis stage, the abuse scenarios help in describing the operational environment in such a way that later in the process the* [information security] *requirements can be modeled and prioritized. The abuse subjects and abuse scenarios create a realistic vision of the mandatory security mechanisms which facilitate design."*

Furthermore, other developers observed that the use of the Meta-notation improves the communication between the developers of Company B and its clients. The technical expert said:

> *"Nowadays the test cases are planned based on use cases, and there are situations where it is not clear between the client and supplier* [Company B] *if the feature is a bug or an undefined feature. This can be avoided when the test cases are planned using abuse cases."*

### *Areas for Future Improvement*

The use of the Meta-notation derived from the design theory was not without issues for Company B. While in general the notation succeeded and the design project reached a successful conclusion, further research might focus on several potential adjustments. The developers did recognize some features that the Meta-notation failed to cover. For example, one of the developers said:

> *"It would be good to have a risk analysis technique which helps to filter the achieved results* (abuse cases, etc.) *It would help one to concentrate on the right aspects and we could prepare for those aspects."* Another developer reported that, *"Many information security features come from the architecture phase so it would be good to highlight the architectural aspects more clearly."*

The practitioners saw that security classifications might be difficult to utilize in certain projects. The software engineer stated:

> *"The classification of the information* [security object] *is at too low a level. The security classifications are useful in some specific industries, for example in defense forces; but in many cases they are redundant."* According to the software architect, *"The method* [Meta-notation] *is good for specific use, but is not usable in all projects. I personally would use the classification of information only when required."*

Ideas also surfaced after the project for further development of the Meta-notation. The software architect noted:

> *"The* [Meta-notation enriched RAD] *process model should include information about responsibilities, what kind of skills the task requires, and what information is needed to complete that task, and define what the results are."*

That is, the Meta-notation should provide more guidance in the architectural verification stage. In addition, the technical expert saw a need to *"…crosscheck the use cases and abuse cases* [in order to improve the risk analysis process]." To minimize this gap, the practitioners created a matrix for that purpose. The practitioners also added their own in-house risk management calculations methods to estimate risk, on the basis of which the abuse cases were prioritized.

## Discussion And Implications

The meta-method contributes to the SIS literature by addressing the six meta-requirements of our SIS design theory. First, most of the existing SIS design methods lack modeling support at the three levels of abstraction as required by meta-requirement 2. Following the higher level of abstraction, we are led to propose the Meta-notation. In the example illustrated in this paper, the notation consists of six dimensions, which can be applied to different development situations at different levels of abstraction (and hence address the second SIS meta-requirement—to offer modeling support at the three levels of abstraction). However, this does not mean that one has to apply all six elements *pro forma*: developers can select from these six elements the relevant ones for the design setting.

Second, only four prior SIS design methods are able to address the third meta-requirement. That is, many of the existing SIS design methods cannot be integrated into ISD methods (the developmental duality problem). Our work indicates that the Meta-notation can be added to existing notations for modeling IS or software. Within the new framework, it is possible to contextualize security-specific notation in existing and proven modeling notations. The theoretical framework indicates the combined modeling approach will allow security and functionality to be consistently modeled. The practical success of the clinical research solution supports this claim.

Third, the framework and Meta-notation model take into consideration the autonomy of developers (the fourth meta-requirement). Developers need to independently exercise their contextualized preferences in selecting and using methods as a basis of ISD. Existing SIS design methods constrain this autonomy in developing SIS by limiting developers to independent secure development methods. The new framework adapts security design to the preferred methodology. The practical success collected from two cases also supports this facet of the theoretical frame.

Fourth, there seems to be value in enabling systems design in organizations known to be emergent and evolving (the fifth meta-requirement). The Meta-notation framework may also address the constant need for novel methods, not just for security, but also for shifting purposes. Such methods need to be routinely modified by practitioners to fit regularly changing situations. In our fieldwork, the settings provide support for this claim.
Finally, our general use of the design theory is at a level that Walls et al. (2004) characterize as "third level" usage (new insights are discovered as a result of the systematic use of design theory). After reviewing 26 articles that apply their idea of design theory, Walls et al. (2004 p. 56) point out that no previous study exemplified stage four. In our case, though, three observations/modifications involve elaboration of the original design theory. These include the observation that product meta-requirements are difficult to see in customized IS, the necessity for dividing design process hypotheses into two domains, and the necessity for adding process and notation to the design process aspects. This contribution may be seen to align with the

highest of four levels of design theory application, a level at which gaps are discovered in, and improvements are made to, the original design theory. The work above approaches, if not achieves, this fourth level of design theory work.

## Conclusions

IS and software development methods do not address security aspects seriously enough. As a result, many SIS design methods, from checklists to more advanced integrated IS design methods, have been established. However, these methods typically fail to meet five of the six SIS method meta-requirements, discovered through design theory. As a result, different SIS design methods cover different levels of IS, but each lacks the comprehensiveness needed for wider application. Most of the methods for designing SIS are difficult to integrate into ISD methods. Moreover, existing SIS design methods impair the autonomy of developers.

A design theory leads to a Meta-notation that provides a framework for addressing these six meta-requirements and thus helps resolve the shortcomings of the known SIS design methods. This approach includes a Meta-notation for adding security into existing and future ISD methods. The Meta-notation has been validated in two practical settings with exactly these issues. The experience gained from these two action research settings demonstrates that the proposed Meta-notation is relevant, feasible, and easy to embed into normal ISD in a real-life setting. While the research shows that further work is needed, the design theory, and its Meta-notation framework, provides a promising solution to a current and critical security problem in ISD.

## *References*

Alderson, A., Hull, M.E.C., Jackson K. and Griffiths, L.E., (1998) "Method engineering for industrial real-time and embedded systems," *Information and Software Technology* (40) 8, pp. 443–454.

Avison, D., Baskerville, R. and Myers, M. (2001) "Controlling action research projects". *Information Technology and People* (14) 1, pp. 28–45.

Backhouse, J. and Dhillon, G. (1996) "Structures of responsibility and security of information systems". European Journal of Information Systems. (5) 1, pp. 2–9.

Baskerville, R. (1988) *Designing Information Systems Security.* John Wiley Information Systems Series.

Baskerville, R. (1989) "Logical Controls Specification: An approach to information system security", In H. Klein and K. Kumar (eds.) *Systems Development for Human Progress*. Amsterdam: North-Holland.

Baskerville, R. (1992), "The developmental duality of information systems security," *Journal of Management Systems* (4) 1, pp. 1–12.

Baskerville, R. (1993) "Information systems security design methods: Implications for information systems development," *ACM Computing Surveys* (25) 4, pp. 375–414.

Baskerville, R. (1994) "Research notes: Research directions in information systems security," *International Journal of Information Management* 14 (5), pp. 385–387.

Baskerville, R. and Pries-Heje, J. (2004) "Short cycle time systems development," *Information Systems Journal* (14) 2, pp. 237–264.

Baskerville, R. and Pries-Heje, J. (1999). Grounded action research: A method for understanding IT in practice. *Accounting, Management and Information Technology*, 9, 1-23.

Baskerville, R. and Wood-Harper, T. (1998) "Diversity in information systems action research methods," *European Journal of Information Systems* 7, pp. 90–107.

Beck, K. (1999) *Extreme Programming Explained*. Massachusetts: Addison Wesley Longman, Inc.

Boehm, B.W. (1988). "A spiral model of software development and enhancement," *IEEE Computer* (21) 5, pp. 61–72.

Booysen, H.A.S. and Eloff, J.H.P. (1995) "A methodology for the development of secure application systems," In Jan H.P. Eloff, Sebastiaan H. von Solms (eds) *Information Security - the Next Decade, Proceedings of the IFIP TC11 Eleventh International Conference on Information Security*. New York: Chapman & Hall, pp. 255–269.

Brinkkemper, S., Lyytinen, K., and Welke, R. (Eds.). (1996). *Method Engineering*. London: Chapman and Hall.

Brinkkemper, S., Saeki M. and Harmsen F. (1999) "Meta-modelling based assembly techniques for situational method engineering," *Information Systems* (24) 3, pp. 209–228.

BS7799 (1993) *Code of Practice for Information Security Management,* Department of Trade and Industry. British Standard Institution, London, UK.

BS7799-1 (2000) *Code of Practice for Information Security Management,* Department of Trade and Industry. British Standard Institution, London, UK.

Castano, S., Fugini, M., Martell, G. and Samarati, P. (1995) *Database Security*. Addison-Wesley.

Chokhani, S. (1994) "Toward a national public key infrastructure," *IEEE Communications Magazine* (32) 9, pp. 70–74.

Clements, D. (1977) *Fuzzy Ratings for Computer Security Evaluation*. Unpublished PhD thesis, University of California, Berkeley, 1977, cited from Hoffman, L., Michelman, E., and Clements, D. "SECURATE - Security evaluation and analysis using fuzzy metrics," in: *AFIPS National Computer Conference Proceedings*, 1978, pp. 531–540.

Cole, R., Purao, S., Rossi, M. and Sein, M. K. (2005) "Being Proactive: Where Action Research Meets Design Research," In D. Avison, D. Galletta & J. I. DeGross (Eds.), *Proceedings of the Twenty-Sixth International Conference on Information Systems*, pp. 325–336

Dhillon, G. and Backhouse, J. (2001) "Current directions in IS security research: Toward socio-organizational perspectives," *Information Systems Journal* (11) 2, pp. 127–153.

Dhillon, G. (1997) *Managing Information Systems Security*. MacMillan Press LTD, UK.

Ellmer, E., Pernul, G. and Kappel, G. (1995) "Object-Oriented Modeling of Security Semantics," In *Proceedings of the 11th Annual Computer Society Applications Conference (ACSAC'95)*. IEEE Computer Society Press, pp 52–61.

GASSP (1999) "Generally Accepted System Security Principles (GASSP)". Version 2.0. Cited from Poore, R. S. (1999) "Release for public comment," *Information Systems Security* (8) 3.

Gause, D. and Weinberg, G. (1989) "*Exploring Requirements: Quality Before Design*" New York: Dorset House.

Gigch van, J.P. (1991) *Systems Design Modeling and Metamodeling* Plenum Press, New York.

Hare, R. M. (1963) *Freedom and Reason*, Oxford University Press, UK.

Herrmann, G. and Pernul, G. (1999) "Viewing Business-Process Security from Different Perspectives," *International Journal of Electronic Commerce* (3) 3, pp. 89–103.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004) "Design science in information systems research," *MIS Quarterly* 28 (1), 75–105.

Hillegersberg, J.V and Kumar, K. (1999) "Using metamodeling to integrate object-oriented analysis, design and programming concepts," *Information Systems* (24) 2, pp. 113–129.

Hirschheim, R. and Klein, H. (1992) "A Research Agenda for Future Information Systems Development Methodologies," In: in W.W. Cotterman and J.A. Senn (eds): *Challenges and Strategies for Research in Systems Development*, pp. 235–269.

Hirschheim, R., and Klein, H.K. (1989) "Four paradigms of information systems development,"*Communications of the ACM* (32) 10, pp. 1199–1216.

Hirschheim, R., Klein, H. K. and Lyytinen, K. (1995) *Information Systems Development and Data Modelling: Conceptual and Philosophical Foundations*. Cambridge University Press, UK.

Hitchings, J. (1995) "Achieving an integrated design: The way forward for information security" In Jan H.P. Eloff, Sebastiaan H. von Solms (eds) *Information Security - the Next Decade, Proceedings of the IFIP TC11 Eleventh International Conference on Information Security*. New York: Chapman & Hall.

Iivari, J. (1989) "Levels of abstraction as a conceptual framework for an information system," In E. D. Falkenberg and P. Lindgreen (eds): *Information System Concepts: An In-depth Analysis*. North-Holland, Amsterdam.

ISF (2005). International Security Forum. "The Forum's Standard of Good Practice for IS security" January 2005, Available at http://www.isfsecuritystandard.com/index-_ie.htm

ISO/IEC. (2005). *ISO/IEC 17799: Information technology -- Security techniques -- Code of practice for information security management* (International Standard No. ISO/IEC 17799:2005(E)). Geneva: International Standards Organization.

ITSEC (1990). *Information Technology Security Evaluation Criteria (ITSEC), Provisional Harmonized Criteria, Version 1.2.* Brussels, Belgium: Commission of European Communities, Directorate--General.

Jaaksi, A. (1998) "Our cases with use cases," *Journal of Object-Oriented Programming* (10), pp. 58–65.

Jacobson, I., Christerson, P. Jonsson, P. and Övergaard, G. (1992) *A Use Case Driven Approach.* Addison-Wesley Publishing Company.

James, H.L. (1996) "Managing information systems security: a soft approach," *In Proceedings of the Information Systems Conference of New Zealand,* pp. 10–20.

Jayaratna, N. (1994) *Understanding and Evaluating Methodologies: NIMSAD, A Systemic Framework*. London: McGraw-Hill.

Jones, D., Gregor, S. and Lynch, T. (2003) "An Information Systems Design Theory for Web-based Education," *Proceedings of the IASTED International Symposium on Web-based Education*,

Kasper, G. M. (1996) "A theory of decision support system design for user calibration," *Information Systems Research* (7) 2, pp. 215–232.

Klein, H.K. and Myers, M.D. (1999) "A set of principles for conducting and evaluating interpretive field studies in information systems," *MIS Quarterly* 23, pp. 67–94.

Kumar, K. and Welke, R.J. (1992) "Methodology engineering: A proposal for situation-specific methodology construction" In W.W. Cotterman and J.A. Senn (eds): *Challenges and Strategies for Research in Systems Development*, pp. 257–269.

Kukathas, C. and Pettit (1990) *Rawls: A Theory of Justice and Its Critics.* Stanford University Press, CA, USA.

Lyytinen, K. (1987) "Two Views on Information Modeling," *Information and Management* 12, pp. 9–19.

Markus, M.L., Majchrzak, A. and Gasser, L. (2002) "A design theory for systems that support emergent knowledge process," *MIS Quarterly* (26) 3, pp. 179–213.

Marttiin, P., Harmsen, F. and Rossi, M. (1996) "A functional framework for evaluating method engineering environments: The case of Maestro II/Decamerone and MetaEdit+". In S. Brinkkemper, K. Lyytinen and R. J. Welke (Eds.), *Method Engineering: Principles of Method Construction and Tool Support*. London: Chapman and Hall, pp. 63-86.

Mathiassen, L. and Purao, S. (2002) "Educating reflective systems developers," *Information Systems Journal* (12), pp. 81–102.

McDermott, J. and Fox, C. (1999) "Using abuse case models for security requirements", *In Proceedings of the 15th Annual Computer Security Applications Conference*, pp. 55–64.

McLean, J. (1990) "The specification and modelling of computer security," *IEEE Computer* (23) 1, pp. 9–16.

Menezes, A.J., van Oorschot, P.C. and Vanstone, S.C. (1999) *Handbook of Applied Cryptography*. CRC Press, USA.

Niiniluoto, I. (1999) *Critical Scientific Realism*, Oxford University Press, Oxford, UK.

Nuseibeh, B., Finkelstein and A, Kramer, J. (1996) "Method engineering for multi-perspective software development," *Information and Software Technology* (38) 4, pp. 267–274.

Ockham, W. (1990) *Philosophical Writings: A selection.* Hackett Publishing Company, Indianapolis, USA.

OECD, (1996) "Guidelines for the Security of Information Systems" OECD, Paris, France.

Olnes, J. (1994) "Development of Security Policies," *Computers & Security* (13), pp. 628–636.

Parker, D. B. (1998) *Fighting Computer Crime - A New Framework for Protecting Information*. Wiley Computer Publishing. USA.

Parker, D. (1981) *Computer Security Management.* Reston, Virginia: Reston Publishing.

Pernul, G. (1992) "Security constraint processing during multilevel secure database design," In *Proceedings of the 8th Annual Computer Security Applications Conference,* pp. 349–370.

Pernul, G., Tjoa A. M. and Winiwarter, W. (1998) "Modelling data secrecy and integrity," *Data and Knowledge Engineering* (26), pp. 291–308.

Pottas D. and von Solms, S.H. (1995) "Aligning information security profiles with organizational policies," In Jan H.P. Eloff, Sebastiaan H. von Solms (eds) *Information Security - the Next Decade, Proceedings of the IFIP TC11 Eleventh International Conference on Information Security*. New York: Chapman & Hall.

Rumbaugh, J., Jacobson I. and Booch, G. (1999) *The Unified Modeling Language Reference Manual.* Addison Wesley.

Röhm, A.W., Pernul, G. and Herrmann, G. (1998) "Modelling secure and fair electronic commerce," In Proceedings of the 14th Annual Computer Security Applications Conference, pp. 155–164.

Röhm, A.W. and Pernul, G. (1999) "COPS: A model and infrastructure for secure and fair electronic markets," In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences*, p. 8021.

Sandhu, R.S. (1993) "Lattice-based access controls," *IEEE Computer* (26) 11, pp. 9–19.

Schein, E. (1987) *The Clinical Perspective in Fieldwork*, Sage, Newbury Park.

Simon, H. A. (1996) *The Science of the Artificial* (3rd ed.). Cambridge, Mass.: MIT Press.

Sindre, G. and Opdahl, A.L. (2000) "Eliciting security requirements by misuse cases," *Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems*, pp. 120–131.

Siponen, M.T. (2001) "An analysis of the recent IS security development approaches: descriptive and prescriptive implications," In: G. Dhillon (eds:) *Information Security Management - Global Challenges in the Next Millennium*, Idea Group, pp. 101–124.

Siponen, M.T. (2005a) "Analysis of modern IS security development approaches: towards the next generation of social and adaptable ISS methods," *Information and Organization* (15) 4, pp. 339–375.

Siponen, M.T. (2005b) "An analysis of the traditional is security approaches: implications for research and practice," *European Journal of Information Systems* (14) 3, pp. 303–315.

Slooten van, K. (1996) "Situated method engineering," *Information Resources Management Journal* (9) 2, pp. 23–31.

Smith, G.W. (1989) "Multilevel secure database design: A practical application" In *Proceedings of the 5th Annual Computer Security Application Conference*, pp. 314–321.

Straub, D.W. and Welke, R.J. (1998) "Coping with systems risk: Security planning models for management decision making," *MIS Quarterly* (22) 4, pp. 441–464.

Strens, R. and Dobson, J. (1993) "How responsibility modelling leads to security requirements," *In Proceedings on the Workshop on New Security Paradigms*, pp. 143–149.

Susman, G. and R. Evered (1978) "An assessment of the scientific merits of action research." *Administrative Science Quarterly* 23(4), pp. 582–603.

Thomas, R.K. and Sandhu. R.S. (1994). "Conceptual foundations for a model of task-based authorizations," In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pp. 66–79.

Tolvanen, J-P. (1998) *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence*. Academic Dissertation, University of Jyväskylä.

Truex, D.P., Baskerville, R. and Klein, H.K. (1999) "Growing systems in an emergent organization," *Communications of The ACM* (42) 8, pp. 117–123.

Truex, D., Baskerville, R. and Travis, J. (2000) "Amethodical systems development: The deferred meaning of systems development methods," *Accounting, Management and Information Technology* (10), pp. 53–79.

Walls, J.G., Wildmeyer, G.R. and El Sawy, O.A. (1992) "Building an information systems design theory for vigilant EIS," *Information Systems Research* (3) 1, pp. 36–59.

Walls, J. G., Widmeyer, G. R., and El Sawy, O. A. (2004). Assessing information system design theory in perspective: How useful was our 1992 initial rendition? *JITTA: Journal of Information Technology Theory and Application*, (6)2, pp. 43-58.

Walsham, G. (1996) "The emergence of interpretivism in IS research," *Information Systems Research* (6), pp. 376–394.

Wand, Y. and Weber, R. (1993) "On the ontological expressiveness of information systems analysis and design grammars," *Journal of Information Systems* (3) 4, pp. 217–237.

Wand, Y., Storey, V. and Weber, R. (1999) "An ontological analysis of the relationship construct in conceptual modeling," *ACM Transactions on Database Systems* (24) 4, pp. 494–528.

White, E.F.R. and Dhillon, G. (2005) "Synthesizing information system design ideals to overcome developmental duality in securing information systems" In R. H. Sprague (Ed.), *Proceedings of the 38 th Annual Hawaii International Conference on System Sciences.* Los Alamitos, California: IEEE Computer Society, pp. 186–195.

Willison, R. (2002) "Opportunities for computer abuse: Assessing a crime specific approach in the case of barings bank," Unpublished PhD Thesis, The London School of Economics, UK.

Wood, C.C., Banks, W.W., Guarro, S.B., Garcia, A.A., Hampel, V.E. and Sartorio, H.P. (1987) *Computer Security: A Comprehensive Controls Checklist*. John Wiley and Sons.

Wood-Harper, T. (1989) "Comparison of Information Systems Definition Methodologies: An action research multiview perspective". Unpublished Ph.D. Thesis, University of East Anglia.

# APPENDIX A: PRELIMINARY CLINICAL FIELDWORK DETAILS

Company A is a film and production center aiming at facilitating regional film and media production. In 1999 Company A recognized the potential of the Web to support their business activities. Company A wanted to build a web-based database for scouting and infrastructure for the distribution of digital media to their customers through the Internet. To further improve their ISD practice as a means of achieving the aforementioned goals, Company A participated in a collaboration project between the University of Oulu and several other industrial organizations. Company A hired two software houses to develop parts of the functionality of the IS. Because Company A aimed to handle their business-related activities (scouting database, distribution of digital products), through the Internet, there was a crucial need for addressing respective security concerns.

The role of researchers in this part of the development was to support both Company A directly and one of the software companies hired by Company A to develop Company A's systems. This software company followed an object-oriented modeling notation - Unified Modeling Language (UML) – as the basis of their ISD. However, based on the interviews, and observations on the design process and design documents in this company, the use of ISD methods and notation were rather informal, minimalist and solution-oriented. This means that developers were using only those parts of UML that they really needed in each project for the development of the IS or software. The software company was unable to address the security concerns by their current ISD method (UML) and practices:

> *"We did not have such security methods as we have for normal ISD design and modeling … but the design and modeling techniques we are currently using ... concentrate on doing software. We don't have a method which includes security aspects."* – a technical project manager of the company

The company's incentive for adaptation of this method was to fill this gap (the lack of notation support with respect to security aspects) in their ISD development practice. The experiences, opinions, and evaluations of the practitioners were recorded in interview sessions and design documents were reviewed. The action research followed a facilitative-clinical methodology without multiple cycles (Schein, 1987).

**Intervention**

First, the Meta-notation SIS method was introduced to the company. The idea was to use the Meta-notation enriched ISD method in the project. The project started with the creation of use cases: The researcher helped the practitioners convert the IS requirements into use cases. Then their experiences, opinions, and evaluations were recorded in an interview session. The next figure presents an example of the use of the notation in this project (Figure 14).

| | |
|---|---|
| **Use case** | Searching for work force. |
| **Version** | 1.0 |
| **Functional summary** | Registered user access work force database in order to search for a suitable work force. |
| **Frequency** | Several times during the production planning. |
| **Usability requirements** | - |

| | |
|---|---|
| **Actor/security subject** | A registered user from a production company |
| **Security classification of the subject** | Confidential |
| **Security objects and access types to security objects** | Security objects: Database ("Person" persistent objects)<br>Access types: Actor/security subject can read the entire content of the work force database. |
| **Security policy/specific security restrictions** | Actor/security subject is not allowed to write or modify the contents of the work force database. |
| **Preconditions** | Work force database exists. Actor has been authenticated. |
| **Exceptions** | - |

**Figure 14. A use case example from the project.**

In the example presented in Figure 14, registered users access a web-based database to search for the proper work force (functional summary). The actor/security subject is a registered user and the security classification is confidential. The actor/security subject can read the entire contents of the work force database (access types to security object). Figure 15 presents a modified example taken from the project. The notation is called "a second level use case" (as named by the software house). In Figure 15, there are two groups of security subjects: registered user and anonymous user.

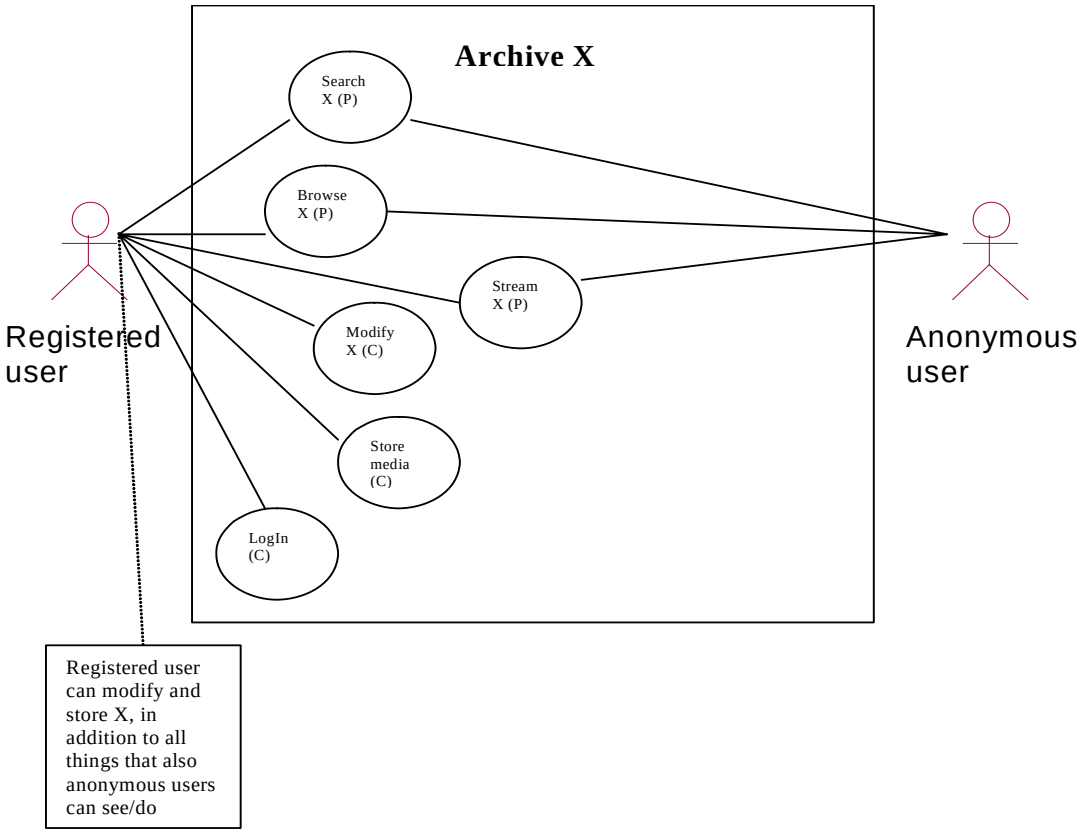


**Figure 15. Modified example of "a second level use case" from the project.**

The elements of the archive X are security objects, and in this case, security objects are classified as confidential and public. Registered users are also allowed to modify and store X (security policy), while anonymous users are not.

**Results**

The preliminary study indicates that the design theory and the Meta-notation would hold in practice. Reflection represented by recorded comments of the collaborators indicated that the Meta-notation approach was feasible and practical (but not without limitations):

> The meta-notion was found very simple: "*It was clear, some new lines were added into the diagrams*". In fact, an analyst expected the Meta-notation to be "*much more difficult*" as "*usually methods by academics are complex and hard to understand, but this use case notation was surprisingly clear*". All developers seemed to understand the terms: "*Generally yes,"* according to a developer, *"but the concept of 'Security Policy/Specific Security' restrictions was the hardest"*.
> Also one developer found the concept/dimension of 'access types to security objects' difficult at first sight: "*At first it was hard to understand ... but after considering the example in the paper everything became clear in the end"*.

There were indications in the preliminary study that the modeling was comprehensive:

> The developers did not recognize that anything was missing from the Meta-notation. For example, one developer said, "*Nothing was missing. I did not see anything lacking"*, while another reported, "*Well, I can't really say that anything was missing"*, and "*I can't think of anything to add*."

It appeared the notation helped acquire and document the security requirements, and the developers were able to use the method within their preferred approach:

> The developers regarded the meta-notion as very useful and relevant: "*This is much smarter than the normal use case, whether for security use or not*. *These are of real use.*" According to another developer, the Meta-notation was "*OK. Useful extra fields were added to it.*"

Also, the developers saw potential for future use of the method even in other preferred methods:

> All developers agree that they would consider use of this information security use case notation in future: "*Yes, if there were need for security*".
> Another developer commented that: "*Yes I would use it … immediately when we meet something with this kind of security requirements. This can be quite handy*." Another developer said *"… it felt simple so I'm sure I would use it again if there is a need to address security aspects."*

These representative results from the preliminary setting suggested strong support for the validity of the underlying SIS design theory:

> The overall evaluation of the Meta-notation by the technical project manager of the software company responsible for the development of the system was also positive, saying he *"would give it eight out of ten."* An analyst in the same interview evaluated the Meta-notation as *"10 out of 10. We just add some lines into the notation and put the rest in there"*.

# APPENDIX B: ITERATION LOOP 3 IN COMPANY B

The research design involved a nested set of action research iterations within the action taking phase of the Action Research study in Company B. Thus there were two sets of iterations in Stage 2 of the overall research project (See Figure 10). In this appendix, we will describe the action taking phase of Iteration 2 in more detail. This action taking phase involved an independent action research cycle all on its own, one in which the developers integrated the design theory's Meta-notation into the company's proprietary development method. There were three iterations of this inner action research loop (Iteration 3 in Figure 10). We refer to these as iterations 3.1 – 3.3.

The action research team for this part of the research was different from the encompassing (Iteration 2) action research study in Stage 2. The team consisted of a researcher, an architect, a subject matter expert, and an engineer.

*Integration iteration 3.1*

In the first integration iteration, the dimensions of the Meta-notation were integrated into 16 RAD process documents by the researchers. The integration was carried out one development phase at a time, starting from the requirement specification phase. In this iteration, attention was paid to ensuring that the transformation of information between different ISD stages was accurate and effective. For example, the security classifications of subjects and objects, the security policy and security constraints were integrated into a use case template (Figure 16). The abuse case document was created by using the use case template of the company and adding the abuse case specific fields (e.g., abuse subject, countermeasure, etc.) to this template. The abuse case template was then integrated into the same process document as the use cases. Additionally, the security classifications and constraints were integrated into the class description document and also into the architectural documents. To facilitate the use of the Meta-notation enriched process documents, instructions on the use of the Meta-notation were also added to all documents.

| | |
|---|---|
| **Purpose:** | What is the purpose of this use case? |
| **Priority:** | High / Medium / Low |
| **Frequency:** | How often this feature is used? |
| **Actor:** | User and user's role in the system *(=security subject).* |
| ***Classification of the actor:*** | *For example: Top secret, Secret, Confidential, Public* |
| **Preconditions:** | What are the preconditions for this use case? |
| **Result:** | What is the result of this use case? |
| **Definition:** | The success scenario of this use case. |
| **Processed Information:** | The information entity processed in this use case *(= security object).* |
| ***Classification of the object:*** | *For example: Top secret, secret, confidential, public* |
| **Exceptions:** | Define alternative scenarios and exceptions. |
| **User's rights:** | What are the user's rights in this use case? *Note that the user's rights, also seen as security constraints, are more specific than the security classifications.* |
| ***Security policy:*** | *The security policy regarding this use case.* |
| **Requirements:** | Links to specified requirements of this use case. |
| **Test cases:** | Links to specified test cases of this use case. |

**Figure 16. The Meta-notation enriched use case. The new SIS relevant fields (after iteration 1) are shown in *italic* font.**

*Evaluation of the first integration iteration*

The authors collected feedback on these 16 Meta-notation enriched RAD process documents from seven practitioners: a technology manager, a business manager, a method specialist, two software architects and two managers. The practitioners wanted to separate the abuse cases from use cases in their own process documents. The practitioners also felt that the security classifications were not usable in their proposed form and location in the process documentation, and that the classifications were at the wrong level of abstraction. The R&D manager and the software architect requested the addition of more guidance on the use of Meta-notation, and the participants agreed that a process map could be useful in solving this guidance problem. Based on the ideas collected in this meeting, researchers continued to address these problems and proceeded with the integration in the second integration iteration.

*Integration iteration 3.2*

Based on the feedback, we modified 24 existing process documents and defined three new process documents. These new process documents were: unwanted functionality specification (abuse cases), the instruction on the use of Meta-notation in ISD, and the process map. The instruction document involved a simple example of the use of Meta-notation in practice, and a proposal to collect the security classifications of subjects and objects in one matrix (Table 4). This matrix shows the classifications of subjects and objects and their security constraints. The unwanted functionality specification describes the abuse cases, based on the abuse scenarios developed in the RAD process.

Table 4. The security classifications and constraints of an eCommerce system.

| Role | Web site **classification: public** | File 1 **classification: secret** | Database **classification: confidential** | Customer info **classification: top secret** |
|---|---|---|---|---|
| **Administrator** classification: top secret | read / write | read / write | read / write | read / write |
| **Employee** classification: secret | read / write | Read | read | read |
| **End user** classification: confidential | read | - | read | - |
| **Outside visitor** classification: public | read | - | - | - |

Based on the comments in the first iteration, we modified the process documents by removing the security classifications from the class level. We also integrated security classifications into the database design document. The abuse scenario and abuse subject were integrated into the operational environment document, and the security requirements were highlighted by categorizing them in their own security category in the requirements catalogue. Furthermore, the classification of the security subjects and

objects was moved to the information security design specification document. These modifications also led to a change in the way information was transformed between the process documents[12], and updates were required in the auditing documents to correspond to the current version of the Meta-notation enriched RAD.

During the second iteration, a document was developed to enable communication between the client organization's management and Company B's sales and project personnel. Abuse subjects and scenarios were integrated into the operational environment specification, because that document was already used (prior to this project) for communication purposes between Company B and its clients. The abuse scenarios provided, for example, written stories of possible compromises and how these could damage the IS. In accordance with critical comments from the ISD department manager, the orientation of the abuse scenarios was such that they could be understood by Company B's sales personnel and clients, in order to communicate a general picture of SIS issues[13].

*Evaluation of the second integration iteration*

After these modifications, a method specialist, a software architect and a system architect provided feedback on the modified process documents. They felt that the Meta-notation terminology was rather academic and they wanted to change the terminology to correspond to the terms already used in the company: *"We want to get rid of the academic terminology, and make the terminology more understandable for all developers."* – method specialist. The software architect continued: *"The unwanted functionality document should be coherent with the functional specification* [in terms of the language]*. The abuse cases should be understandable and the metaphors should be the same as in the traditional use cases"*.

The practitioners also saw that the same information was presented twice in different documents. They saw that it is easier to update the documents when a developer only has to change the information in one process document. The practitioners were also satisfied with separating the security requirements into their own requirements category to highlight SIS features. Several previous problems were evaluated as solved: the treatment of abuse subjects, abuse scenarios and the instruction document on using Meta-notation enriched RAD and Meta-notation in software development. After collecting the feedback, analyzing the comments, we planned how to address the stated problems in the third integration iteration.

[12] For example, the abuse subjects and scenarios are defined in the requirement specification phase. Later, in requirement analysis, that information is used to prioritize the information security requirements and to define the abuse cases. In the design and implementation phase, the information defined above is needed in designing the access control mechanisms and countermeasures to the abuse scenarios, and finally the SIS countermeasures are tested in the testing phase. All other information on SIS features (e.g., security constraints) must be associated similarly throughout the whole RAD process to ensure that the features will be implemented and tested correctly.

[13] The document, including the abuse subjects and scenarios, works as a basis for describing abuse cases and prioritizing security requirements later in the ISD process.

*Integration iteration 3.3*

In the third integration iteration, we modified 25 process documents, based on the feedback provided by the company's practitioners in the second iteration. In this iteration, most of the changes were terminology-related: the terminology was harmonized with terminology used in the company. For instance, the practitioners of the company felt that it would better to use the term "role" or "actor" instead of "security subject", to make the method easier to learn.

The second concern mentioned during the second iteration concerned the maintenance of the documentation. We used a matrix to compile information from several documents, e.g., security classification of the subjects and objects (Boyseen and Eloff, 1995). By collecting the classification information into the information security plan document, it becomes easier to modify the information (Table 5). The practitioners objected to the security classifications and constraints matrix (Table 4) because it included too much information. When the IS is large, this table becomes unreadable. For example, as few as 100 objects and 20 actors would make this matrix (Table 4) so complex it would be difficult to understand. They recommended that it should be divided into two matrices (Table 5 and Table 6). This two-matrix solution requires fewer resources, since the modification of the information in the process documents would be centralized in a limited number of main process documents (e.g., the information security design specification). This centralization of information also improves the coherence within the overall documentation. After these modifications, we updated the links between the process documents.

Table 5. The security classifications of the roles.

| **User role** | **Description** | **Classification** |
|---|---|---|
| **Role 1** | Description of this user role | The classification of the role |
| **Role 2** | … | (e.g., top secret / secret / confidential / public) |
| **Role …** | | |

The security policy was modified during the third iteration. It was clear to the practitioners that, at this point in the development, it is better to refer to a specific security requirement in the requirement catalogue than a client organization's information security policy. According to the developers, the security policy may be generic, for example, drawn from BS7799. This modification is consistent with the Meta-notation, since the aim of the Meta-notation was not to refer to an organizations' high-level security policy in use cases, but relevant sub-polices.

The access matrix for the roles in the system was developed in this iteration to make the design and maintenance of access rights easier (Table 6). Through this matrix, one can see what operations certain roles (security subjects) need to perform on security objects.

Table 6. The access matrix for the roles (C = create, R = read, U = update, D = delete).

| Processed information | Role 1 | Role 2 | Role 3 | Role … | The needed operations |
|---|---|---|---|---|---|
| **object 1** | CR | RUD | R | R | **CRUD** |
| **object 2** | CRUD | - | - | - | **CRUD** |
| **object 3** | RU | RU | R | R | **RU** |

After these modifications (Figure 17), we updated the transformation of information between the different process documents and delivered the documents to the company for evaluation.

| | |
|---|---|
| **Purpose:** | What is the purpose of this use case? |
| **Priority:** | High / Medium / Low |
| **Frequency:** | How often is this feature used? |
| **Actor:** | User's role in the IS (=security subject). *Security classification defined in IS plan (see Table 3).* |
| **Preconditions:** | What are the preconditions for this use case? |
| **Result:** | What is the result of this use case? |
| **Definition:** | The success scenario of this use case. |
| **Processed Information:** | The information entity processed in this use case (= security object). *The functions needed are defined in IS plan (access matrix, see Table 4).* |
| **Exceptions:** | Define alternative scenarios and exceptions. *Possible links to abuse cases.* |
| **User rights:** | What are the user's rights in this use case? *The functions needed are defined in IS plan (access matrix, see Table 4).* |
| **Requirements:** | Links to specified requirements of this use case. |
| **Test cases:** | Links to specified test cases of this use case. |

**Figure 17. The final version of the Meta-notation enriched use case. Part of the information, such as security classifications of subjects and objects, is moved to Tables 5 and 6.**

*Evaluation of the third integration iteration*

At the conclusion of this third iteration, the proposed method was harmonized with the terminology and practices used by the company. These modifications affected 25 process documents. The developers were satisfied with the modifications and they decided that the method is ready for field test.

# APPENDIX C: VALIDITY CRITERIA FOR IS ACTION RESEARCH

The research described in this paper tests the practical success (relevance, feasibility and applicability) of the idea of Meta-notation by testing notation in practice in clinical and action research settings. A clinical research approach was selected initially because it is known to be the highly suitable for initial testing and possible adjustment of ISD methods.

Seven validity criteria for IS action research results have been proposed (Baskerville and Wood-Harper 1998): (1) the research should be set in a multivariate social situation; (2) the observations should be recorded and analyzed in an interpretive frame; (3) there is researcher's action that intervened in the research setting; (4) the method of data collection includes participatory observation; (5) changes in the social setting are studied; (6) the immediate problem in the social setting must have been resolved during the research; and (7) the research should illuminate a theoretical framework that explains how the actions led to a favourable outcome. These criteria are applied in evaluating the results of the intervention in the two companies.

The research in this paper conforms to the seven validity criteria above. Regarding to the *first* criterion, the research in both cases was set in a multivariate social situation. The Company A case involved a complex relationship between an ISD contractor and their client. The client explicitly required a SIS. In the Company B case, the research was also carried out in a multivariate social situation, which included interaction between different stakeholders (e.g., practitioners of the company, management, researchers, etc.).

Regarding *second* criterion, for both cases in companies A and B, the observations were recorded and analyzed within an interpretive research framework. Windows into this data and the interpretation have been provided in this report. With respect to *third* criterion, the researchers actively intervened in both companies; in the case of Company A, the researcher worked directly with the ISD contractor to introduce the Meta-notation into their practices. Similarly, in Company B case, the researchers' actions intervened in the research setting by introducing a new method (Meta-notation) in the company.

For the *fourth* criterion: the method of data collection in Company A included participatory observation as well as interviews. In the case of Company B, the data collection also included participatory observation in integration phase (action taking), but not in the empirical testing phase (evaluation). In this phase, the practitioners informed the researcher through e-mails, meetings, and by providing the documentation (e.g., specifications, design documents) from the project.

As to the *fifth* criterion: in both cases, the outcome of the ISD process was assessed in terms of the collaborators' reviews of the usability and success of the modified ISD method and the ISD modification process in different phases of the study.

The *sixth* criterion is satisfied because the immediate problem in the research was resolved through integrating the Meta-notation to the ISD practices of companies A and B, according to the evaluation of the collaborators in the field.

Finally, regarding the *seventh* criterion, the actions in the settings were tightly linked to the Meta-notation theoretical framework. This framework defined the actions and explains clearly how the actions led to the favorable outcome.

To summarize, the research experiences from two cases suggest that the Meta-notation was successful in practice. It was perceived to be relevant in use, feasible, and easily applied for extending normal ISD methods for secure purposes. This experience supports the design theory framework that drove the creation of the Meta-notation and its application for the purpose of creating SIS. While we do not specifically examine the satisfaction of the foregoing work in terms of criteria for design theory (e.g., Hevner, et al., 2004), there is evidence that work satisfying criteria for action research will additionally satisfy the criteria for design theory, and vice-versa (Cole et al., 2005).

Journal of the Association for Information Systems
ISSN: 1536-9323