Engineering Secure Future Internet Services

Wouter Joosen[1], Javier Lopez[2], Fabio Martinelli[3], and Fabio Massacci[4]

[1] Katholieke Universiteit Leuven
wouter.joosen@cs.kuleuven.be
[2] University of Malaga
jlm@lcc.uma.es
[3] National Research Council of Italy
Fabio.Martinelli@iit.cnr.it
[4] University of Trento
massacci@dit.unitn.it

Abstract. In this paper we analyze the need and the opportunity for establishing a discipline for engineering secure Future Internet Services, typically based on research in the areas of software engineering, of service engineering and security engineering. Generic solutions that ignore the characteristics of Future Internet services will fail, yet it seems obvious to build on best practices and results that have emerged from various research communities.
The paper sketches various lines of research and strands within each line to illustrate the needs and to sketch a community wide research plan. It will be essential to integrate various activities that need to be addressed in the scope of secure service engineering into comprehensive software and service life cycle support. Such a life cycle support must deliver assurance to the stakeholders and enable risk and cost management for the business stakeholders in particular. The paper should be considered a call for contribution to any researcher in the related sub domains in order to jointly enable the security and trustworthiness of Future Internet services.

1 Introduction

1.1 Future Internet Services

The concept named Future Internet (FI) aggregates many facets of technology and its practical use, often illustrated by a set of usage scenarios and typical applications. The Future Internet may evolve to use new infrastructures, network technologies and protocols in support of a growing scale and a converging world, especially in light of smaller, portable, ubiquitous and pervasive devices. Besides such a network-level evolution, the Future Internet will manifest itself to the broad mass of end users through a new generation of services (e.g. a hybrid aggregation of content and functionality), service factories (e.g., personal and enterprise mash-ups), and service warehouses (e.g., platform as a service). One specific service instance may thus be created by multiple service development organizations, it may be hosted and deployed by multiple providers, and may

J. Domingue et al. (Eds.): Future Internet Assembly, LNCS 6656, pp. 177–191, 2011.

be operated and used by a virtual consortium of business stakeholders. While the creative space of services composition is in principle unlimited, so is the fragmentation of ownership of both services and content, as well as the complexity of implicit and explicit relations among participants in each business value chain that is generated. In addition, the user community of such FI services evolves and widens rapidly, including masses of typical end users in the role of *prosumers*(producing and consuming services). This phenomenon increases the scale, the heterogeneity and the performance challenges that come with FI service systems.

This evolution obviously puts the focus on the **trustworthiness** of services. Multiparty service systems are not entirely new, yet the Future Internet stretches the present know how on building secure software services and systems: more stakeholders with different trust levels are involved in a typical service composition and a variety of potentially harmful content sources are leveraged to provide value to the end user. This is attractive in terms of degrees of freedom in the creation of service offerings and businesses. Yet this also creates more vulnerabilities and risks as the number of trust domains in an application gets multiplied, the size of attack surfaces grows and so does the number of threats. Furthermore, the Future Internet will be an intrinsically dynamic and evolving paradigm where, for instance, end users are more and more empowered and therefore decide (often on the spot) on how content and services are shared and composed. This adds an extra level of complexity, as both risks and assumptions are hard to anticipate. Moreover, both risks and assumptions may evolve; thus they must be monitored and reassessed continuously.

1.2 The Need for Engineering Secure Software Services

The need to organize, integrate and optimize the research on engineering secure software services to deal effectively with this increased challenge is pertinent and well recognized by the research community and by the industrial one. Indeed, there is also a growth of successful attacks on ICT-service systems, both in terms of impact and variety. This obviously harms the economic impact of Future Internet services and causes significant monetary losses in recovering from those attacks. In addition, this induces users at several levels to lose confidence in the adoption of ICT-services.

From a business perspective, however, we are now witnessing the emergence of new and unprecedented models for service-oriented computing for the Future Internet: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These models have the potential to better adhere to an economy of scale and have already shown their commercial value fostered by key players in the field. Nevertheless, those new models present change of control on the applications that will run on an infrastructure not under the direct control of the business service provider. For business critical applications this could be difficult to be accepted, when not appropriately managed and secured. These issues are of an urgent practical relevance, not only for academia, but also for industry and governmental organizations. New Internet services will have to be

provided in the near future, and security breaches in these services may lead to large financial loss and damaged reputation.

1.3 Research Focus on Developing Secure FI Services

Our focus is on the creation and correct execution of a set of methodologies, processes and tools for secure software development. This typically covers requirements engineering, architecture creation, design and implementation techniques. However this is not enough! We need to enable **assurance**: approving that the developed software is secure. Assurance must be based on **justifiable evidence**, and the whole process designed for assurance. This would allow the uptake of new ICT-services according to the latest Future Internet paradigms, where services are composed by simpler services (provided by separate administrative domains) integrated using third parties infrastructures and platforms. The need of managing the intrinsic **modularity and compose-ability** of these architectures, traditionally shared with commercial off the shelf components (COTS), should drive the development of corresponding methodologies. Clearly industry needs to prioritize its efforts in order to improve their return of investments (ROI). Thus, embedding **risk/cost analysis** in the SDLC is currently one of the key research directions in order to link security concerns with business needs and thus supporting a business case for security matters.

Our research addresses the early phases of the development process of services, bearing in mind that the discovery and remediation of vulnerabilities during the early development stages saves resources. Thus our joint research activities fall in six areas: (1) *security requirements* for FI services, (2) *creating secure service architectures and secure service design*, (3) supporting *programming environments for secure and compose-able services*, (4) *enabling security assurance*, *integrating the former results in (5) a risk-aware and cost-aware software development life-cycle (SDLC)*, and (6) the delivery of case studies of future internet application scenarios.

The first three activities represent major and traditional stages of (secure) software development: from requirements over architecture and design to the composition and/or programming of working solutions. These three activities interact to ensure the integration between the methods and techniques that are proposed and evaluated. This is a first element that drives to *research integration*.

In addition, the research programme adds two horizontal activities that span the service creation process. Both the security assurance programme and the programme on Risk and Cost aware SDLC will interact with each of the initial three activities, drive the requirements of these activities and leverage upon, even integrate their outcome. This is a second element that drives to research integration.

Finally, notice that all 5 research activities mentioned above will be inspired and evaluated by their application in specific FI application scenarios. This third element complements the overall research programme that leads to integrated research and intensive research collaboration in the area.

In the sequel of this paper we elaborate on the relevant sub domains and techniques that we consider useful for engineering secure Future internet services.

2 Security Requirements Engineering

The main focus of this research strand is to enable the modeling of high-level requirements that can be expressed in terms of high-level concepts such as compliance, privacy, trust, and so on. These can be subsequently mapped into more specific requirements that refer to devices and to specific services. A key challenge is to support dealing with an unprecedented multitude of autonomous stakeholders and devices – probably one of the most distinguishing characteristics of the FI.

The need for assurance in the Future Internet demands a set of novel engineering methodologies to guarantee secure system behavior and provide credible evidence that the identified security requirements have been met from the point of view of all stakeholders. The security requirements of Future Internet applications will differ considerably from those of traditional applications. The reason is that Future Internet applications will not only be distributed geographically, as are traditional applications, but they will also involve multiple autonomous stakeholders, and may involve an array of physical devices such as smart cards, phones, RFID sensors and so on that are perpetually connected and transmit a variety of information including identity, bank accounts, location, and so on. Some of these transactions might even happen transparently to the user; for example, a person's identity could be seamlessly communicated by a personal device to the store she is entering to do the shopping. Addressing concerns about identity theft, unauthorized credit card usage, unauthorized transmission of information by third-party devices, trust, privacy, and so on are critical to the successful adoption of FI applications.

Service-orientation and the fragmentation of services (both key characteristics of FI applications) imply that a multitude of stakeholders will be involved in a service composition and each one will have his own security requirements. Hence, eliciting, reconciling, and modeling all the stakeholders' security requirements become a major challenge [5]. Multilateral Security Requirements Analysis techniques have been advocated in the state of the art [14] but substantial research is still needed. In this respect, agent-oriented and goal-oriented approaches such as Secure Tropos [12] and KAOS [8] are currently well recognized as means to explicitly take the stakeholders' perspective into account. These approaches will represent a promising starting point but need to be uplifted in order to be able to cope with the level of complexity put forward by FI applications. Furthermore, it is important that security requirements are addressed from a higher level perspective, e.g., in terms of the actors' relationships with each other. Unfortunately, most current requirements engineering approaches consider security only at the technological level. In other words, current approaches provide modeling and reasoning support for encryption, authentication, access control, non-repudiation and similar requirements. However, they fail to capture the high-level requirements of trust, privacy, compliance, and so on.

This picture is further complicated by the vast number and the geographical spread of smart devices stakeholders would deploy to meet their requirements. Sensor networks, RFID tags, smart appliances that communicate not only with the user but with their manufacturers, are examples of such devices. Such deployments inherit security risks from the classical Internet and, at the same time, create new and more complex security challenges. Examples include illicit tracking of RFID tags (privacy violation) and cloning of data on RFID tags (identity theft). Applications that involve such deployments typically cross organization boundaries.

In light of the challenges and principles highlighted above, we identify the following detailed objectives:

- The definition of techniques for the identification of all stakeholders (including attackers), the elicitation of high-level security goals for all stakeholders, and the identification and resolution of conflicts among different stakeholder security goals;
- The refinement of security goals into more detailed security requirements for specific services and devices;
- The identification and resolution of conflicts between security requirements and other requirements (functional and other quality requirements);
- The transformation of a consolidated set of security requirements into security specifications.

The four objectives listed above obviously remain generic by nature, one should bear in mind though that the forthcoming techniques and results will be applied to a versatile set of services, devices and stakeholder concerns.

3 Secure Service Architecture and Design

FI applications entail scenarios in which there exist a huge amount of heterogeneous users and a high level of composition and adaptation is required. These factors increase the complexity of applications and make it necessary to leverage existing mechanisms and methodologies for software construction as well as researching about new ways to take this complexity into account in a holistic manner. These applications enable pervasive, ubiquitous scenarios where multiple users, devices, third-party components interact continuously and seamlessly, so security enforcement mechanisms are indispensable. The design phase of the software service and/or system is a timely moment to enforce and reason about these security mechanisms, since by that phase one must have already grasped a thorough understanding of the application domain and of the requirements to be fulfilled. Furthermore, at design-time a preliminary version of the application architecture has been produced.

The software architecture encompasses the more relevant elements of the application, providing either a static or/and a dynamic view of the application. A more comprehensive definition can be found in [2], where it is defined as “the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them”.

The security architecture for the system must enforce the visible security properties of components and the relationships between them. All this information makes it feasible to enforce, assess and reason about security mechanisms at an early phase in the software development cycle.

The research topics one must focus on in this subarea relate to model-driven architecture and security, the compositionality of design models and the study of design patterns for FI services and applications. The three share the common ambition to maximize reuse and automation while designing secure FI services and systems.

As for the first element the aim is to support methodologies that utilize several easy-to-understand models to represent the application. According to the model-driven approach, these models may be manipulated and converted automatically into other models, in such a way that they all preserve certain properties. So, it would be possible to specify a first high-level model with some high-level security policies. Then, by automation, this model could be converted into another more specific model, in which the security policies become more detailed, closer to the enforcement mechanisms that will fulfil them. This process should be applied until a basic version of the application architecture can be released.

The integration of security aspects into this paradigm is the so-called model-driven security [6], leading to a design for assurance methodology in which every step of the design process is performed taking security as a primary goal. A way of carrying out this integration includes first decomposing security concerns, so that the application architecture and its security architecture is decoupled. This makes possible for architects to assess more easily tradeoffs among different security mechanisms, simulate security policies and test security protocols before the implementation phase, where changes are typically far more expensive.

In order to achieve this, it is first needed to convert the security requirements models into a security architecture by means of automatic model transformations. These transformations are interesting, since whilst requirements belong to the problem-domain, the architecture and design models are within the solution-domain, so there is an important gap to address. In the context of security modeling, it is extremely relevant to incept ways to model usage control (e.g., see [21,22,18]), which encompasses traditional access control, trust management and digital rights management and goes beyond these building blocks in terms of definition and scope. Finally, by means of transformation patterns, it is required to research on new ways to map the high-level policies established at requirements stage into low-level, enforceable policies at run-time. Furthermore note that FI scenarios include Cloud and GRID services and although some work has already been made in the area [23], further research is necessary to find out what kind of security architecture is required in the context and how to carry out the decomposition of such fairly novel architectures.

Until this point in the software and service development process, different concerns – security among them – of the whole application have been separated into different models, each model representing different functional and non-functional concerns that different stakeholder may have about it. However,

in order to grasp a comprehensive understanding of the application as a whole, it is required to integrate all these views into a unified one. This process is called composition [25,11] and, as a recent work suggests [20], it is possible to perform it at run-time, adding a new level of flexibility and adaptation for FI applications. Regarding composition, several topics will be studied. First, it is desirable to define contracts for model composition, in such a way that only correct compositions are allowed, limiting the propagation of design flaws through the models. Second, given that different sub-architectures may exist, each addressing different concerns – even different security sub-architectures for different security requirements – it is required to assure that the composition of all these architectures is accomplished and that all the requirements are met in this composition. Finally, adaptation of composite services is a key area of interest. FI scenarios are very dynamic, so threats in the environment may change along the time and some reconfiguration may be required to adapt to that changes.

The last research focus is on design patterns and on reusable architectural know-how. A design pattern is a general repeatable solution to a commonly occurring problem in software design. Design patterns, once identified, allow reuse of design solutions that have proved to be effective in the past, reducing costs and risks usually arisen by uncertainty, leveraging a risk and cost-aware . There are large catalogues and surveys on security patterns available [26,13], but the FI applications yet to come and the new scenarios enabled by FI need to extend and tailor these catalogues. In this context, the first step is studying the patterns currently available and, what is more important, to analyze the relationships amongst them [17], identifying those which may be useful for FI scenarios. Finally, how to bridge the gap among problem patterns – such as problem frames or KAOS refinement patterns and solution patterns – architectural patterns – must be analyzed, both from a general perspective and from a security perspective for security-critical software systems.

4 Security Support in Programming Environments

Security Support in Programming Environments is not new; still it remains a grand challenge, especially in the context of Future Internet (FI) Services. Securing Future Internet Service is inherently a matter of secure software and systems. The context of the future internet services sets the scene in the sense that (1) specific service architectures will be used, that (2) new types of environments will be exploited, ranging from small embedded devices ("things") to service infrastructures and platform in the cloud, and (3) a broad range of programming technologies will be used to develop the actual software and systems.

The search for security support in programming environments has to take this context in account. The requirements and architectural blueprints that will be produced in earlier stages of the software engineering process cannot deliver the expected security value unless the programs (code) respect these security artefacts that have been produced in the preceding stages. This sets the stage for model driven security in which transformations of architecture and design artefacts is essential, as well as the verification of code compliance with various

properties. Some of these properties have been embedded in the security specific elements of the software design; other may simply be high priority security requirements that have articulated – such as the appropriate treatment of concurrency control and the avoidance of race conditions – in the code, as a typical FI service in the cloud may be deployed with extreme concurrency in mind.

Supporting security requirements in the programming – code – level requires a comprehensive approach. The service *creation* means must be improved and extended to deal with security needs. Service creation means both aggregating and *composing* services from pre-existing building blocks (services and more traditional components), as well as *programming* new services from scratch using a state-of-the-art programming language. The service creation context will typically aim for techniques and technologies that support compile and build-time feedback. One could argue that security support for service creation must focus on and enable better static verification. The service *execution* support must be enhanced to deal with hooks and building blocks that facilitate effective security enforcement at run-time. Dependent on the needs and the state-of-the-art this may lead to interception and enforcement techniques that "simply" ensure that the application logic consistently interacts with underpinning security mechanisms such as authentication or audit services. Otherwise, the provisioning of the underpinning security mechanisms and services (e.g. supporting mutual non repudiation, attribute based authorization in a cloud platform etc.) will be required as well for many of the typical FI service environments. Next we further elaborate on the needs and the objectives of community wide research activities.

4.1 Secure Service Composition

Future Internet services and applications will be composed of several services (created and hosted by various organizations and providers), each with its own security characteristics. The business compositions are very dynamic in nature, and span multiple trust domains, resulting in a fragmentation of ownership of both services and content, and a complexity of implicit and explicit relations among the participants.

Service Composition Languages. One of the challenges for the secure service composition is the need for new formalisms to specify service requests (properties of service compositions) and service capabilities, including their security policies, and tools to generate code for service compositions that are able to fulfil these requirements based on the available services. In addition to complying with the requested functional and quality-of-service-related characteristics, composition languages must support means to preserve at least the security policy of those services being composed. The research community needs to consider the cases where only partial or inadequate information on the services is available, so that the composition will have to find compliant candidates or uncover the underspecified functionality.

Middleware Aspects. The research community should re-investigate service-oriented middleware for the Future Internet, with a special emphasis on

enabling deployment, access, discovery and composition of pervasive services offered by resource-constrained nodes.

4.2 Secure Service Programming

Many security vulnerabilities arise from programming errors that allow an exploit. Future Internet will further reinforce the prominence of highly distributed and concurrent applications, making it important to develop methodologies that ensure that no security hole arises from implementations that exploit the computational infrastructure of the Future Internet. The research community must further investigate advances over state-of-the-art in fine-grained concurrency to enable highly concurrent services of the Future Internet, and will improve analysis and verification techniques to verify, among others, adherence to programming principles and best-practices [10].

Verifiable Concurrency. Lock-free wait-free algorithms for common software abstractions (queues, bags, etc.) are one of the most effective approaches to exploit multi-core parallelism. These algorithms are hard to design and prove correct, error-prone to program, and challenging to debug. Their correctness is crucial to the correct behaviour of client programs. Research should now focus on build *independently* checkable proofs of the absence of common errors, including deadlock, race conditions, and non-serialize-ability [16].

Adherence to Programming Principles and Best-Practices. Programming support must include methods to ensure the adherence of a particular program to well-known programming principles or best-practices in secure software development. Emphasis will be put on language extensions that guarantee adherence to best-practices, and verified design patterns that can be used during development. The research community might investigate and re-visit methods from language-based security, in particular type systems, to enforce best-practises currently used in order to prevent cross-site scripting attacks and similar vulnerabilities associated with web-based distributed applications. Obviously, the logical rationales underlying such best-practises must be articulated, enabling he development of type systems enforcing these practises directly – thus allowing users to deviate from rigid best-practices while still maintaining security.

4.3 Platform Support for Security Enforcement

Future Internet applications span multiple trust domains, and the hybrid aggregation of content and functionality from different trust domains requires complex cross-domain security policies to be enforced, such as end-to-end information flow, cross-domain interactions and usage control. In effect, the security enforcement techniques that are triggered by built-in security services and by realistic in the FI setting, must address the challenge of *complex interactions* and of *finely grained control* [15]. Research should therefore focus on the enforcing cross-domain barriers in the interaction among different cross-domains, and on the enforcement of fine-grained security policies via execution monitoring.

Secure Cross-Domain Interactions. Web technology inherently embeds the concept of cross-domain references, and applications are isolated via the Same-Origin-Policy (SOP) in the browser. From a functional perspective, the SOP puts limitations on compose-ability and cooperation of different applications, and from a security perspective, the SOP is not strong enough to achieve the appropriate application isolation.

Finely Grained Execution Monitoring. Trustworthy applications need run-time execution monitors that can provably enforce advanced security policies [19,3] including fined-grained access control policies usage control policies and information flow policies [24].

Supporting Security Assurance for FI Services. Assurance will play a central role in the development of software based services to provide confidence about the desired security level. Assurance must be treated in a holistic manner as an integral constituent of the development process, seamlessly informing and giving feedback at each stage of the software life cycle by checking that the related models and artefacts satisfy their functional and security requirements and constraints. Obviously the security support in programming environments that must be delivered will be essential to incept a transverse methodology that enables to manage assurance throughout the software and service development life cycle (SDLC). The next section clarifies these issues.

5 Embedding Security Assurance and Risk Management during SDLC

Engineering secure Future Internet services demands for at least two traversal issues, security assurance and risk and cost management during SDLC.

5.1 Security Assurance

The main objective is to enable assurance in the development of software based services to ensure confidence about their trustworthiness. Our core goal is to incept a transverse methodology that enables to manage assurance throughout the software development life cycle (SDLC). The methodology is based on two strands: A first sub-domain covers early assurance at the level of requirements, architecture and design. A second sub-domain includes the more conventional and complementary assurance techniques based on implementation.

Assurance during the Early Stages of SDLC. Early detection of security failures in Future Internet applications reduces development costs and improves assurance in the final system. This first strand aims at developing and applying assurance methods and techniques for early security verification. These methods are applied to abstract models that are developed from requirements to detailed designs.

One main area of research is step-wise refinement of security, by developing refinement strategies, from policies down to mechanisms, for more complex

secure protocols, services, and systems. This involves the definition of suitable service and component abstractions (e.g., secure channels) and the setup of the corresponding reasoning infrastructure (e.g., facts about such channels). Moreover, we need to extend the refinement framework with *compositional* techniques for model-based secure service development. Model decomposition supports a divide-and-conquer approach, where functional and security-related design aspects can be refined independently. Model composition must preserve the refinement relation and component properties. Our aim is to offer developers support for smoothly integrating security aspects into the system development process at any step of the development.

Enabling rigorous and formal analysis processes. There is an increasing demand of models and techniques to allow the formal analysis of secure services. The objective is to develop methodologies, based on formal mappings from the constraint languages, to other formalisms for which theorem proving and/or (semi-)decision procedures are available, to support formal (and, when possible, automated) reasoning about the security policies models.

The methodologies must be supported by automatic protocol verification tools, such as the AVISPA [1] tool set and the Scyther tool [7], for the verification of Future Internet protocols. The planned extensions require not only significant efficiency improvements, but also the ability to deal with more complex primitives and security properties. Moreover, the Dolev-Yao attacker model [9] used by these tools needs to be extended to include new attack possibilities such as adaptive corruptions, XSS attacks, XML injection, and guessing attacks on weak passwords. In addition, for assurance, there is the need to extend model checking methods to enable automatic generation of protocol correctness proofs that can be independently verified by automated theorem proving.

Security Assurance in Implementation. Several assurance techniques are available to ensure the security at the level of an implementation. Security policies can be implemented correctly by construction through a rigorous secure programming discipline. Internet applications can be validated through testing. In that case, it is possible to develop test data generation that specifically targets the integration of services, access control policies or specific attacks. Moreover, implementations can be monitored at run-time to ensure that they satisfy the required security properties.

Complementing activities are related to **secure programming**. This strand addresses a comprehensive solution for program verification, while adding a particular focus on session management in concurrent and distributed service compositions.

In addition, an important set of research activities must address **testing**. This strand covers the testing activities which complement programming and coding. We can consider three aspects, that although not comprehensive, present characteristic for service-oriented applications in the future Internet: penetration testing that leverages on the high-level models that are generated in early stages of the software life cycle, automated generation in XML-based input data to maximize the efficiency in the security testing process, and testing of policies

that are the typical high-level front end of a complex service composition. The latter part will focus on access control policies. i

Finally, an important set of activities relates to **run-time verification**. This strand concludes the trilogy of implementation-level testing: run-time *verification* must complement programming-level verification and testing in order to provide the final assurance that the latter cannot deliver, be it for scientific and technological reasons, be it for reasons of organizational complexity. The latter may frequently occur in a multi-organizational context, typical for service compositions in Future Internet. We will study approaches for run-time monitoring of data flow, as well as technologies for privacy-preserving usage control.

Towards a Traverse Methodology. Security concerns are specified at the business-level but have to be implemented in complex distributed and adaptable systems of FI services. We need comprehensive assurance techniques in order to guarantee that security concerns are correctly taken into account through the whole SDLC. A chain of techniques and tools crossing the above areas is planned.

Security Metrics. Measurements are essential for objective analysis of security systems. Metrics can be used directly for computing risks (e.g., probability of threat occurrence) or indirectly (e.g., time between antivirus updates). Security metrics in the Future Internet applications become increasingly important. Service-oriented architectures demand for assurance indicators that can explicitly indicate the quality of protection of a service, and hence indicate the effective level of trustworthiness. These metrics should be assessed and communicable to third parties. Clients want to be sure that their data outsourced to other domains, which the clients cannot control, are well protected. We need to define formal metrics and measurements that can be practically calculated. Compositional calculation approaches will be studied in this context. Many of the proposed metrics will be linked to and determined by the various techniques in the Engineering process.

5.2 Risk and Cost Aware SDLC

There is the need of the creation of a methodology that delivers a risk and cost aware SDLC for secure FI services. Such a life cycle model aims to ensure the stakeholders' return of investment when implementing security measures during various stages of the SDLC. We can envision several aspects of this kind of SDLC support (see also [4]).

Process: The methodology for risk and cost aware SDLC should be based on an *incremental and iterative process* that is accommodated to an incremental software development process. While the software development proceeds through incremental phases, the risk and cost analysis will undergo new iterations for each phase. As such the results of the initial risk and cost analyses will propagate through the software development phases and become more refined. In order to support the propagation of analysis results through the phases of the SDLC

one needs to develop methods and techniques for the refinement of risk analysis documentation. Such refinement can be obtained both by refining the risk models, e.g. by detailing the description of relevant threats and vulnerabilities, and by accordingly refining the system and service models.

Aggregation: In order to accommodate to a modular software development process, as well as effectively handling the heterogeneous and compositional nature of Future Internet services, one needs to focus on a modular approach to the analysis of risks and costs. In a compositional setting, also risks become compositional and should be analysed and understood as such. This requires, however, methods for aggregating the global risk level through risk composition which will be investigated.

Evolution: The setting of dynamic and evolving systems furthermore implies that risk models and sets of chosen mitigations are dynamic and evolving. Thus, in order to maintain risk and cost awareness, there is a need to continuously reassess risks and identify cost-efficient means for risk mitigation as a response to service or component substitution, evolving environments, evolving security requirements, etc., both during system development and operation. Based on the modular approach to risk and cost analysis one needs methods to manage the dynamics of risks. In particular, the process for risk and cost analysis is highly iterative by supporting updates of global analysis results through the analysis of only the relevant parts of the system as a response to local changes and evolvements.

Interaction: The methodology of this strand spans the orthogonal activities of security requirement engineering, secure architecture and design, secure programming as well as assurance and the relation to each of these ingredients must be investigated. During security requirements engineering risk analysis facilitates the identification of relevant requirements. Furthermore, methods for risk and cost analysis offer support for the prioritization and selection among requirements through e.g. the evaluation of trade-off between alternatives or the impact of priority changes on the overall level of risks and cost. In the identification of security mechanisms intended to fulfil the security requirements, risk and cost analysis can be utilized in selecting the most cost efficient mechanisms. The following architecture and design phase incorporates the security requirements into the system design. The risk and cost models resulting from the previous development phase can at this point be refined and elaborated to support the management of risks and costs in the design decisions. Moreover, applying cost metrics to design models and architecture descriptions allows early validation of cost estimates. Such cost metrics may also be used in combination with security metrics for the optimization of the balance between risk and cost. The assurance techniques can therefore be utilized in providing input to risk and cost analysis, and in supporting the identification of means for risk mitigation based on security metrics.

6 Conclusion

We have advocated in this paper the need and the opportunity for firmly establishing a discipline for engineering secure Future Internet Services, typically based on research in the areas of software engineering, security engineering and of service engineering. We have clarified why generic solutions that ignore the characteristics of Future Internet services will fail: the peculiarities of FI services must be reflected upon and be addressed in the proposed and validated solution.

The various lines of research and the strands within each of research line have been articulated while founding the NESSoS Network of Excellence (www.nessos-project.eu). Clearly, the needs and challenges sketched in this paper reach beyond the scope and capacity of a closed consortium. The topics listed above should and will be shared and tackled by an entire and open research community.

Acknowledgments. We would like to thank the anonymous reviewers for the helpful comments. Work partially supported by EU FP7-ICT project NESSoS (Network of Excellence on Engineering Secure Future Internet Software Services and Systems) under the grant agreement n.256980.

References

1. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
2. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley, Boston (2003)
3. Bauer, L., Ligatti, J., Walker, D.: Composing security policies with polymer. SIGPLAN Not. 40, 305–314 (2005)
4. Braber, F., Hogganvik, I., Lund, M.S., Stølen, K., Vraalsen, F.: Model-based security analysis in seven steps — a guided tour to the coras method. BT Technology Journal 25, 101–117 (2007)
5. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8, 203–236 (2004)
6. Clavel, M., da Silva, V., de O. Braga, C., Egea, M.: Model-driven security in practice: An industrial experience. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 326–337. Springer, Heidelberg (2008)
7. Cremers, C.J.: The scyther tool: Verification, falsification, and analysis of security protocols. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 414–418. Springer, Heidelberg (2008)

8. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. 20, 3–50 (1993)
9. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, Washington, DC, USA, pp. 350–357. IEEE Computer Society Press, Los Alamitos (1981), doi:10.1109/SFCS.1981.32
10. Erlingsson, U., Schneider, F.B.: Irm enforcement of java stack inspection. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 246–255. IEEE Computer Society Press, Los Alamitos (2000)
11. France, R., Fleurey, F., Reddy, R., Baudry, B., Ghosh, S.: Providing support for model composition in metamodels. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, Washington, DC, USA, p. 253. IEEE Computer Society Press, Los Alamitos (2007)
12. Giorgini, P., Mouratidis, H., Zannone, N.: Modelling security and trust with secure tropos. In: Integrating Security and Software Engineering: Advances and Future Vision, IDEA (2006)
13. Group, O.: Security design pattern technical guide, `http://www.opengroup.org/security/gsp.htm`
14. Gürses, S.F., Berendt, B., Santen, T.: Multilateral security requirements analysis for preserving privacy in ubiquitous environments. In: Proc. of the Workshop on Ubiquitous Knowledge Discovery for Users at ECML/PKDD, pp. 51–64 (2006)
15. Hamlen, K.W., Morrisett, G., Schneider, F.B.: Computability classes for enforcement mechanisms. ACM Trans. Program. Lang. Syst. 28, 175–205 (2006), doi:10.1145/1111596.1111601
16. Jacobs, B., Piessens, F., Smans, J., Leino, K.R.M., Schulte, W.: A programming model for concurrent object-oriented programs. ACM Trans. Program. Lang. Syst. 31, 1–1 (2008), doi:10.1145/1452044.1452045
17. Kubo, A., Washizaki, H., Fukazawa, Y.: Extracting relations among security patterns. In: SPAQu'08 (Int. Workshop on Software Patterns and Quality) (2008)
18. Lazouski, A., Martinelli, F., Mori, P.: Usage control in computer security: A survey. Computer Science Review 4(2), 81–99 (2010)
19. Le Guernic, G., Banerjee, A., Jensen, T., Schmidt, D.A.: Automata-based confidentiality monitoring. In: Okada, M., Satoh, I. (eds.) ASIAN 2006. LNCS, vol. 4435, pp. 75–89. Springer, Heidelberg (2008)
20. Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.-M., Solberg, A., Dehlen, V., Blair, G.S.: An aspect-oriented and model-driven approach for managing dynamic variability. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 782–796. Springer, Heidelberg (2008)
21. Park, J., Sandhu, R.S.: The ucon_{abc} usage control model. ACM Trans. Inf. Syst. Secur. 7(1), 128–174 (2004)
22. Pretschner, A., Hilty, M., Basin, D.A.: Distributed usage control. Commun. ACM 49(9), 39–44 (2006)
23. Rosado, D.G., Fernandez-Medina, E., Lopez, J.: Security services architecture for secure mobile grid systems. Journal of Systems Architecture. In Press (2010)
24. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications 21(1), 2003 (2003)
25. Whittle, J., Moreira, A., Araújo, J., Jayaraman, P., Elkhodary, A.M., Rabbi, R.: An expressive aspect composition language for UML state diagrams. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 514–528. Springer, Heidelberg (2007)
26. Yoshioka, N., Washizaki, H., Maruyama, K.: A survey on security patterns. Progress in Informatics 5, 35–47 (2008)