A Methodological Approach to Apply Security Tactics in Software Architecture Design

Gilberto Pedraza-Garcia [*][†]
[*]Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogotá, Colombia
g.pedraza56@uniandes.edu.co
[†]Universidad Piloto de Colombia
Bogotá, Colombia

Hernan Astudillo
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
hernan@inf.utfsm.cl

Dario Correal
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogotá, Colombia
dcorreal@uniandes.edu.co

***Abstract*—Architectural tactics are decisions to efficiently solve quality attributes in software architecture. Security is a complex quality property due to its strong dependence on the application domain. However, the selection of security tactics in the definition of software architecture is guided informally and depends on the experience of the architect. This study presents a methodological approach to address and specify the quality attribute of security in architecture design applying security tactics. The approach is illustrated with a case study about a Tsunami Early Warning System.**

***Index Terms*—Architectural tactics; security tactics application; secure architectures; secure software development; software architecture design.**

I. Introduction

The architecture of a software system is defined through a process of decision making. The result of the definition of a software architecture is a set of views describing several concerns expressed by their stakeholders.

Architectural tactics represent codified knowledge obtained from the experience gained by architects. Architectural tactics are high-level decisions made by architects to meet or improve a quality attribute [1]. Tactics are general guidelines on how to design a specific aspect of a software system without imposing a particular structure of software [2]. However, much of the literature describes the tactics as a catalog organized hierarchically but do not describe the activities to be incorporated into the architecture.

On the other hand, security is a quality attribute with particular characteristics that makes a complex property due to its strong dependence on the application domain and requires a sophisticated analysis. Security is an attribute that is not planned at the beginning of the software life cycle [2]. In terms of architecture design, security is a quality property that can not be resolved in a separate view because it is traversal to the concerns considered into architectural views. Security has dimensions such as: process security, information security, operational security and security deployment. Security tactics can be applied simultaneously in multiple models and use several forms of implementation such as: introduce a new hardware or software security technology, add operational procedures to support secure operation or modify existing structures, among others.

This paper describes a set of activities, tools and a notation to address and specify the quality attribute of security in the architecture design. These activities help to identify, specify and prioritize security needs expressed by stakeholders and guide the definition of security policies. Besides the activities allow to develop a threat model to identify and prioritize risks, establish the required countermeasures, select tactics architecture and annotate them in the design of architecture.

The document is organized as follows: Section 2 gives some background of architectural tactics, Section 3 presents the strategy for the application of security tactics, Section 4 describes the study case and validation, and Section 5 concludes.

II. Background

A. Architectural Tactics

Tactics are well proven design solutions to resolve issues related to quality attributes. Tactics are decisions that influences the control of a response to a quality attribute [1]. Also tactics are considered as measures taken to improve quality attributes [3]. Other authors suggest that a tactic is an architectural building block that provides a generic solution to guide issues related with quality attributes [4]. The common point is that they provide solutions for compliance of quality attributes in a software system.

B. Security Principles

Security is a property that exhibits the ability to a software system to protect data and functionality from unwanted access, ensuring access to authorized actors [5]. Security describes a set of processes and technologies that reliably control the access to resources of a system [2]. Security in software systems has the following characteristics [5]:

- **Confidentiality** Protect data or services from unauthorized access.
- **Integrity** Avoid unauthorized manipulation of data or services.
- **Availability** Ensure the system availability for legitimate use.
- **Authentication** Identify the actors involved in a transaction and verify that they are who claims to be.
- **Nonrepudiation** Ensure that the sender of a message does not deny having sent the message and the receiver does not deny receiving the message.

C. Security Tactics

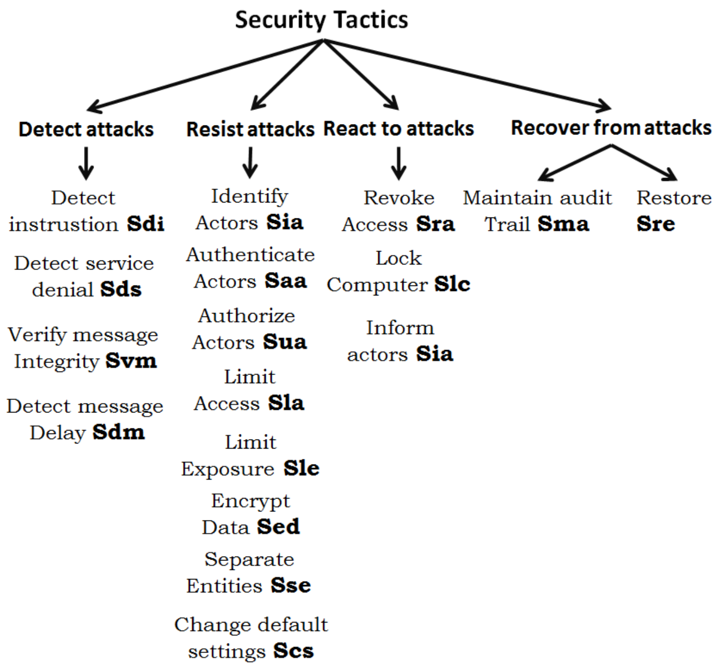

Fig. 1. Security tactics catalog [5]

Security tactics are organized in a catalog described as a tree and grouped into four hierarchies: detect attacks, resist attacks, react to attacks, and recover from attacks [5]. Each hierarchy is organized such that the branches contain categories and the leaves represent the tactics. The security tactics proposed by Bass [5] are presented in Figure 1.

III. Strategy to Apply Security Tactics

In order to define a methodological approach to apply security tactics into software architecture were taken as reference several studies. First, Rozanski [2] defines a set of activities to address the quality attribute of security using perspectives. Second, Fernandez [6] establishes a method for applying security patterns considering the life software cycle. Third, Mayer [7] proposes a process based on the following activities: context analysis and assets identification, security goal determination, security requirements elicitation, and countermeasures selection. Finally, we use the annotations scheme proposed by Harrison et al. [3] to specify the security tactics.

A. Previous activities of architectural design

The following activities of architectural design are proposed before application of security tactics:

1) **Specify functional requirements.** Before applying security tactics is necessary to identify the security requirements of the system. One of the key elements to carry out this task is to analyze the functional requirements. These requirements can be expressed in a Use Case Model where their participating actors, functional expectations and interactions with the software system are described.
2) **Definition of architectural views.** The proposed strategy assumes that the functional requirements have already been defined and the existence of a set of architectural views containing progress in the resolution of functional requirements. A view represents one structural approach where the solution to one or several concerns is described [2]. Rozanski et al. [2] describes in detail how to define the architectural views. A brief description of these views is made below.
   - **Context view.** Describe the interactions, relationships, responsibilities and dependencies between the system and external entities, people or systems.
   - **Functional view.** This view allows to specify the functional elements at runtime. It describes functional capabilities, external interfaces, internal structure, and define functional design approach.
   - **Information view.** Describe how the software system stores, processes, and distributes data. Some design issues are resolved such as the data structure and content and the purpose, usage and ownership of data among others.
   - **Deployment view.** The deployment view allows to represent the physical runtime environment, the technical requirements of the nodes and the mapping of the software elements to the runtime environment established.

B. Activities to apply security architectural tactics

The following activities of architectural design are proposed for the application of the security tactics:

1) **Identify the security requirements.** Security requirements define the security needs in terms of the problem to be solved [8]. The functional requirements, the relevant external regulations, and the organizational security

policies are the inputs that allows to identify security requirements. In this activity, the functional requirements related with security that were expressed by stakeholders are analyzed. Generally, the functional requirements are specified by means of a use case model. Several studies propose to use misuse cases [9] [10], activities [11], or actions [12] to specify security requirements. Misuse cases are independent use cases initiated by external attackers to the system. For example, in the case of use *"Send seismic parameters"* security requirements could be *"The only seismic information is received from the CSN"*, *"The seismic information received is the same as sending the CSN"*.

Other studies propose to describe the security requirements as restrictions [13]. Each security requirement should meet the criteria to make known: what is explicitly (Definition), implicit and explicit assumptions (Assumptions), determine whether it meets the security goals (Satisfaction) [13].

2) **Identification of sensitive resources.** This activity identifies key security needs where resources can be sensitive data or system functions. To identify sensitive resources should take into account the diverse situations where it may affect the security principles. For example: identifying information in hands of others to produce damage, determine whether the disclosure of information violates some law or principle, determine the impact of the loss or modification of information, establish if there is concern about the availability of the information or functional operations, and identify some time-critical information. The input for this activity are the use case model, the information view and the functional view.
3) **Definition of security policies.** The security policy is the basis for designing of a security system because it identifies who is entrusted access to the system resources and the restrictions to the system resources, the rules of system integrity, and the accountability of the resources when they are accessed [2]. A security policy assigns responsibility to a role on a set of resources, define types of authorized access, sensitive operations and associated integrity rules. To specify security policies Lampson [14] has been proposed to consider four aspects: Who has access to data (Secrecy), how data changes or resources are used (Integrity), providing prompt access to information and resources (Availability) and prove who had access to data or resources (Accountability).
4) **Description of threat Model.** To identify system threats is important to known potential attackers the system, its motivation, how can circumvent the security policy, its characteristics in terms of resources, commitment, level of sophistication and the consequences of that policy has been violated. Attackers can be inside and outside the organization. In the literature there are several methods to analyze threats and risks, including the following: HAZard and OPerability study (HAZOP) [15],Fault Tree Analysis (FTA) [16], Failure Mode and Effect Criticality Analysis (FMECA) [17], Markov analysis methods (Markov) [18], Goals Means Task Analysis (GMTA) [19], CCTA Risk Analysis and Management Methodology (CRAMM) [20]. To analyze and describe the threat model we select Attack Tree, a variant of Fault Tree Analysis. The figure 6 shows an attack tree. The activities to build a tree of attack are:
   a) Identify root events. To identify the root events are taken into account the security policies. For example the policy: *"only legitimate users can read profile alert"*, thus the root node can be the undesired event: *"an unauthorized user reads a profile alert"* [21].
   b) Elaborate attack tree. Tree branches are constructed by setting intermediate events that describe possible attacks. The attacks may be directed to a particular component or the whole system. The Attack Tree is expanding with possible triggered actions by the attacker [21].. To know the elements and syntax of the Attacks Tree we suggest reviewing [21], [22], [23].
   c) Analyze attack tree. The attack tree provides information for the risk analysis of the system because it determines how the system will be attacked taking into account whether the event is basic, undevelopment or external [21]. Also, it reveals the interactions for which the system fails.

   Furthermore, this analysis should identify compromised resources, describe the attacks results and determine the attack probability in order to prioritize threats.
5) **Identification of security architectural tactics.** Before selecting tactics, it is necessary to define countermeasures to be applied to mitigate threats. A countermeasure is an action, procedure or technique that avoids or minimizes the impact caused by a threat, attack, or vulnerability on a software system [24]. For example, to the threat *"spoofing user identity"*, a countermeasure can be *"do not store secrets in plain text"*. After establishing the countermeasures it proceeds to describe them on the software architecture using security tactics. For instance, to the countermeasure *"do not store secrets in plain text"* the category of security tactic can be *"resist attacks"* and the particular tactic *"encrypt data (passwords)"* [5].
6) **Specification of security tactics into the software architecture.** In general, the application of security tactics is expressed in the software architecture as [2]:
   a) Add, modify, or delete architectural elements with specific responsibilities such as managing the authorization by means of a server component.
   b) Introduce security technologies such as: user authentication, single-sign-on systems, virtual private networks, database access control systems, and SSL/TLS encryption, among others.
   c) New operational procedures to support secure operation such as certificate management.

These actions affect the structure of the architecture that is organized in the several architectural views. For example the first set of actions can be described in the functional and information view, the second set of actions can be expressed in the deployment and development view, and the third set of actions can be applied in operational view. The next activity is to apply the annotation [3] to describe the tactics in the models presented by several architecture views. Changes due to tactics occur in architectural components or relationships between components. The changes in the structure of components can be:

- **AC.** Add a component to the architecture.
- **IC.** Implement changes to a single component causing low impact.
- **RC.** Duplicate one component causing low impact on the structure.
- **MC.** Modify one component causing high impact and other components must also change.
- **DC.** Delete a component.
- **AT.** Add a security technology.
- **DT.** Delete a security technology.
- **MT.** Update a security technology.
- **AO.** Add an operational procedure.
- **MO.** Modify an operational procedure.

In addition to the above structural changes may occur in the relationships between architectural elements. Those are the following:

- **AR.** Add a relation or connector.
- **MR.** Modify a relation.
- **DR.** Delete a relation.

The tactics proposed by Bass [5] are described in figure 1 along with the abbreviation used in the annotations. The annotation consists of a circle within which is described at the top of the tactic identification and at the bottom the corresponding change identification in the structure. See figure 7.

IV. Validation

A. Study Case: Tsunami Early Warning System

Tsunamis are phenomena generated by underwater earthquakes or landslides and their fast propagation can generate very large waves that potentially produce inundations with tragic consequences [25] in human lives, infrastructure and capacity among others. Tsunami Early Warnings Systems (TEWS) are an effective tool to mitigate the impact on coastal border and avoid human victims caused by the waves. The time that elapses before the tsunami effects are observed on the coast, is just a few minutes; therefore, it is necessary to have data to make the right decisions. In this way, these systems require to be accurate and fast, have a proper understanding of the hydrodynamic processes and make right decisions related with determining the level of danger, evacuation zones and the allocation of emergency / rescue resources.

In particular, Chile is a country with a high probability of tsunamis occurrence because it is located in a very active area wherein an intraplate deformation caused by continental movement and collision with the Nazca slab is presented [26]. In Chile, the National Tsunami Warning System (SNAM) is responsible of fast prediction, reliable response and efficient alarm in response to tsunami occurrence. Currently the TEWS has a set of tools, some manuals and other automatics, for reception of seismic parameters using emails, alert decision-making support, and generation of alert bulletins. Figure 2 shows a diagram of context with these elements.

The proposed architecture is organized into two major subsystems: The first is the Premodeled Scenarios Subsystem, responsible for the preparation of a database with a large number of simulated scenarios using bathymetry to achieve broad coverage location with forecast points, geographical blocks impacted on the coastal line and wave height. The second is the Alert Decision-Making Subsystem to support the tsunami alert generation and monitoring. In this study we focused on the second subsystem.

1) Alert Decision-Making Subsystem: The purpose of this subsystem is to support activities that develop from the occurrence of a seismic event with the possibility to produce tsunamis.

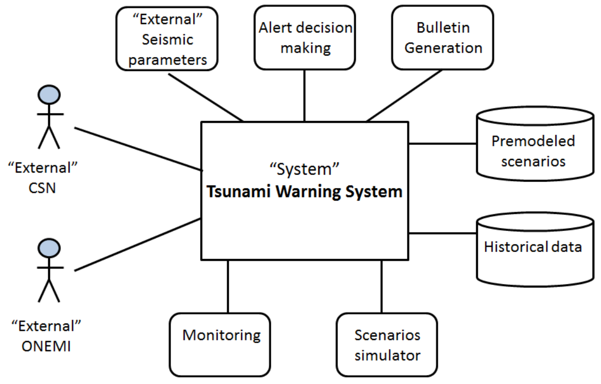

Fig. 2. Context model

The actors involved in this subsystem are:

- **National Seismological Center (CSN).** External actor representing the CSN, whose responsibility is to provide information related with the occurrence of a seismic event.
- **Seismic expert (SNAM).** Seismic expert evaluates the seismic data and decides whether or not to initiate the process of alert decision making.
- **Responsible for making decisions (SNAM).** When an alert profile occur, this actor is responsible for approval, emission and delivery of the Bulletin to the ONEMI.
- **National Emergency Office of the Ministry of Interior and Public Security (ONEMI).** This actor receives the Alert Bulletin and coordinates the logistic for dissemination and notification of the alert to general public and

other government entities.

B. *Previous activities of architectural design*

1) Use Case Model: This subsystem solves the following use cases:

- **Send seismic parameters.** When a seismic event occurs, an official from CSN sends the corresponding data to SNAM for assessment. Besides, these data are communicated to the seismic expert at SNAM.
- **Evaluate seismic data.** When the notification of the occurrence of a seismic event arrives, SNAM seismic expert evaluates the occurrence of a possible tsunami event and starts the generating of an alert profile.
- **Approve and send the bulletin.** Responsible at SNAM for making decisions approves the alert profile, authorizes the generation of the corresponding bulletin and sends it to ONEMI.
- **Change the status of the alert.** The seismic data are frequently updated, this situation requires re-determine the level of alert. SNAM seismic expert back to recalculate the alert status. If the alert status changes then the responsible for making decisions approves and sends the bulletin.

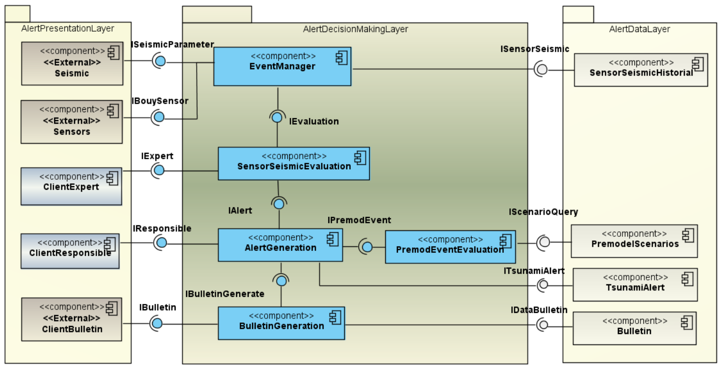
AlertPresentationLayer
AlertDecisionMakingLayer
AlertDataLayer
<<component>> <<External>> Seismic
<<component>> <<External>> Sensors
<<component>> ClientExpert
<<component>> ClientResponsible
<<component>> <<External>> ClientBulletin
ISeismicParameter
IBouySensor
IExpert
IResponsible
IBulletin
<<component>> EventManager
IEvaluation
<<component>> SensorSeismicEvaluation
IAlert
IPremodEvent
<<component>> AlertGeneration
<<component>> PremodEventEvaluation
IBulletinGenerate
<<component>> BulletinGeneration
ISensorSeismic
IScenarioQuery
ITsunamiAlert
IDataBulletin
<<component>> SensorSeismicHistorial
<<component>> PremodelScenarios
<<component>> TsunamiAlert
<<component>> Bulletin

Fig. 3. Functional view

2) Functional view specification: The architectural pattern selected for the solution architecture is layered pattern [27]. The proposed components are divided into three layers as follows (Figure 3):

- **Alert presentation layer.** This layer contains components that allow interoperability with external entities. The *Seismic* component provides the interface with the CSN, permanently monitors and publishes the arrival of seismic data. The *Sensors* component monitors and publishes information related with measures take by buoys and sea level stations. Also, *ClientBulletin* interacts with ONEMI to delivery of the bulletin. The internal presentation components are: *ClientExpert* allows the Seismic Expert to interact with the system, *ClientDecisionResponsible* allows the responsible of decision-making to approve the alert profile and send the bulletin.
- **Alert decision-making layer.** This layer contains the components: *EventManager* which subscribes to events generated by the Seismic and Sensors components and decides who to notify that data. This layer provides the *ISeismicParameter* interface to manage the seismic events and the interface *IBouySensor* manages sensor buoys and sea-level stations measurements. The *SensorSeismicEvaluation* component processes and allows the expert to assess the seismic and monitored data. In addition, this component allows to start the generation process of a alert profile. This component provides the *IEvaluation* interface to accept the seismic and monitored data and the *IExpert* interface to interact with the seismic expert.
  The *AlertGeneration* component applies a set of decision rules to determine whether an alert is or not recommended. If so the alert level is determined according to the geographical area. This component is supported by the *ScenarioQuery* component to obtain the premodeled scenario that most closely matches with the seismic event. Besides this component provides the *IResponsible* interface that allows the responsible to approve the alert profile and start generating the corresponding bulletin.
  The *PremodEventEvaluation* component is responsible for interpolate and select the scenario that best meets the seismic parameters. The selected scenario predicts the wave height and the arrival time to the forecast points. The component provides the *IPremodEvent* interface to manage the corresponding functionality.
  The *BulletinGeneration* component is responsible for the generation of the alert official bulletin and sends it to ONEMI through the *IBulletin* interface.
- **Alert data layer.** This layer contains several components. First, the *SensorSeismicHistorical* component is responsible for the persistence of the seismic data received, the measurements of sea level and the log of actions undertaken in the process, through the *ISensorSeismic* interface.
  Second, the *PremodeledScenarios* component is an instance of the database created in the *PremodeledScenarios* subsystem and contains all the scenarios simulated so far, through the *IScenarioQuery* interface.
  The *TsunamiAlert* component maintains information that supports the decision-making process, through the *ITsunamiAlert* interface. The *Bulletin* component persists alert bulletin data through the *IDataBulletin* interface.

3) Deployment View: Figure 4 shows how these components are deployed. The *SeismicClient* and *BulletinClient* external components are accommodated in the CSN and ONEMI machine respectively and communicate with the *Information Capture Server*. The *ReaderSensor* node represents the execution component responsible for reading measurements from buoys and sea level stations. The *ExpertClient* and *ResponsibleClient* execution components are terminals that allow the interaction of the corresponding actors.

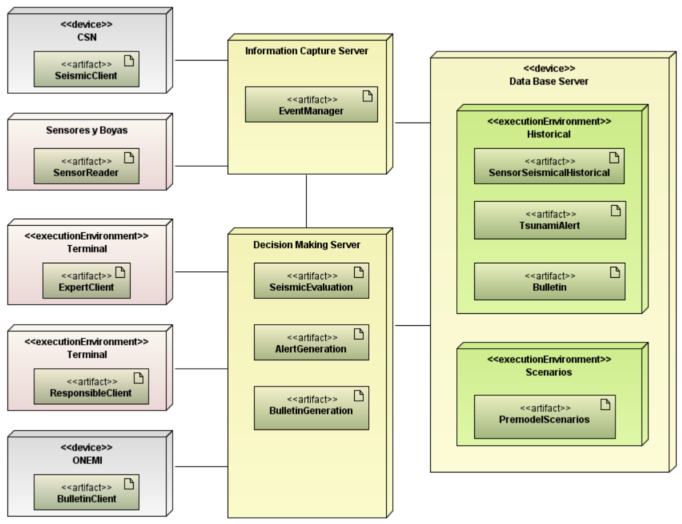

Fig. 4. Deployment view

4) Data Model: The data model proposed for the warning system includes the following entities (Figure 5): *Quake* represents data from a real earthquake. *QuakeReport* describes seismic parameters sent in a report from CSN. *Employee* is a CSN member that sends the report of the earthquake. *SeismicEvaluation* contains a preliminary assessment of the seismic parameters and the justification to start decision-making process. *Expert* represents the responsible for initiating the process of decision-making alert. *TsunamiAlert* contains the data about alert profile generated. *ForecastPoint* represents the coordinates and the forecast of wave height to a place located in coastal border. *Seaside* are data of the coastal zone that represents the point forecast. *Responsible* contains data of the staff member that guarantees the tsunami alert profile. *PremodelScenario* represents a early simulated scenario that is the reference to calculate a forecast point.

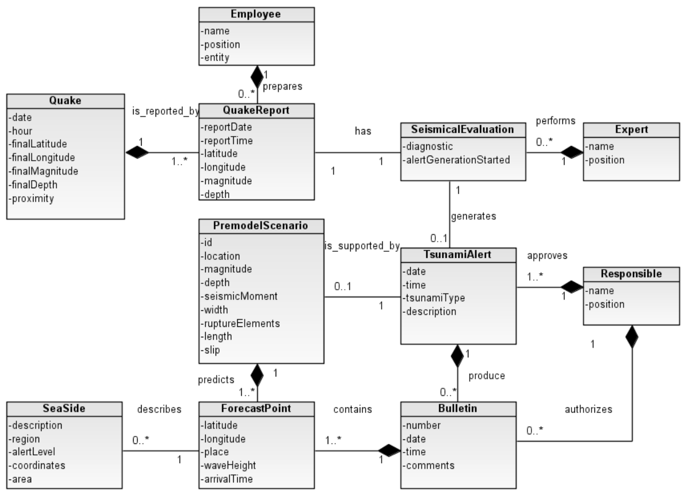

Fig. 5. Information view

C. Activities to apply security architectural tactics

1) Identify the security requirements: Security requirements identified are shown in table I.

TABLE I
SECURITY REQUIREMENTS

Use Case	Security requirement
Send seismic parameters	The seismic data is only received by the CSN
Evaluate seismic data	The seismic evaluation is performed only by the seismic expert
Approve and send the bulletin	Ensure that the bulletin is received by the ONEMI

2) Identification of sensitive resources: Sensitive resources are defined by analyzing the use case model and the functional and information views.

- **Sensitive data.** Sensitive data are identified from the use case model and the information view. For example, the success of defining a correct alert profile depends on the accuracy of the seismic parameters, which makes seismic parameters as sensitive data. Others sensitive data are premodeled scenarios and the alert bulletin.
- **Sensitive operations.** Sensitive operations are extracted from the use case model and the functional view. Since seismic data are sensitive parameters then the delivery of the seismic parameters from the CSN (external entity) becomes sensitive. Also deliver of seismic data, evaluate seismic data (initiation of the alert), make alert decisions, and send the alert bulletin are sensitive operations

3) Definition of security policies: Security policies are defined on previously identified sensitive resources. Below are some of the identified policies:

- The Seismic Expert should be able to initiate the chain of decision-making 7 days a week, 24 hours without the service is affected by denial of service.
- No person may obtain information from the bulletin or alert profile except authorized recipients.
- Seismic data do not change as soon as it is sent by the CSN.
- Seismic data can not be changed from the time it is shipped by CSN.
- The premodeled scenarios can not be modified within the system.
- The event manager is sure that the seismic data were sent by CSN.
- Seismic Expert is the only one who can evaluate the seismic information and he is uniquely able to produce an alert from it.
- Responsible for making decisions of SNAM is the only one who can authorize the creation of the bulletin.
- At the arrival of a tsunami alert bulletin ONEMI must guarantee that the bulletin comes from SNAM.

- Upon arrival of seismic data to Tsunami Warning System should be certainty that comes from CSN.
- Responsible for making decisions SNAM is responsible for bulletins produced.

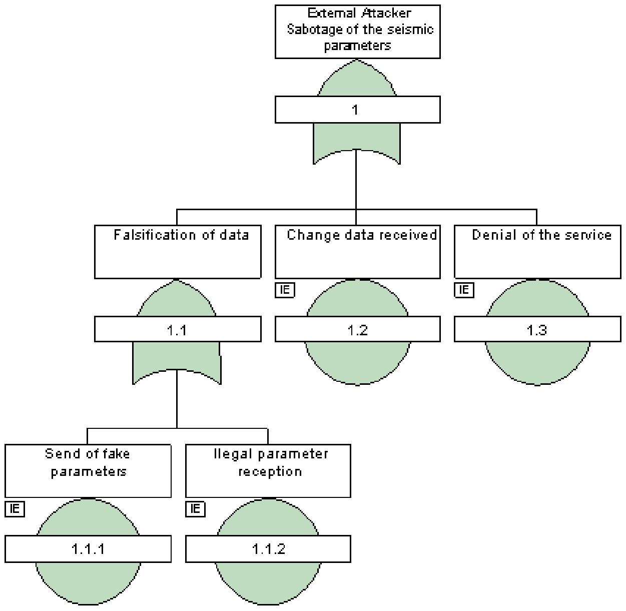

Fig. 6. Attack tree

4) *Description of threat Model:* An attack tree for each attacker is developed and thus the potential threats are identified. The figure 6 shows the tree for an external attacker. The table II presents the main threats identified by the analysis of security policies.

5) *Selection of security tactics:* The countermeasures to mitigate threats are described in Table II. Table III shows some of the tactics selected to mitigate the identified threats.

6) *Specification of security tactics into the software architecture:* Selected tactics are described in the architecture using the annotations described in paragraph 6 of section III-B. Figure 7 shows the annotation to the authentication tactic (Saa) to protect the sensitive resource *"seismic parameters"*. The tactic is implemented by adding the authenticationServer component (AC).

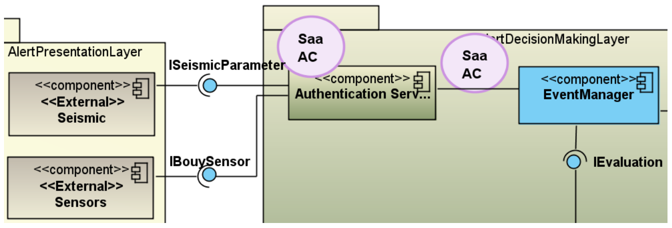

Fig. 7. Addition of authenticate tactic

TABLE II
THREATS AND COUNTERMEASURES [28]

Sensitive Resource	Threat	Countermeasure
Seismic parameters	Send of fake parameters. Illegal parameter reception. Modification of received information. Denegation of service (DoS).	Determine the originator. Verify the identity of the sender. Verify the identity of the sender, protect resources. Increase availability of services.
Evaluate seismic data	Modification of received information. False alert generation. Decision is modified when sent. Warning is ignored by the responsible.	Verify the identity of the expert, protect resources. If insider: Trace back each action. If responsible:Trace back each action, determine the originator. If expert:Trace back each action, determine the originator. Information cannot be changed undetectably, determine the originator. Trace back each action.
Alert bulletin	Bulletin is not send. Bulletin is modified before being sent. Bulletin is sent to a fake destination. Bulletin is modified in transit. DoS	Trace back each action. Verify the identity of the responsible, protect resources, trace back each action. Verify the identity of the sender. Information cannot be changed undetectably, determine the originator. Increase availability of services.

TABLE III
SECURITY TACTICS IDENTIFICATION [28]

Countermeasure	Security tactic
Determine the originator	Digital sign
Verify the identity of the sender	Sender authentication
Protect resources.	Authorization
Increase availability of services	Redundancy
Trace back each action	Logger
Information cannot be changed undetectably	Encryption

V. CONCLUSIONS

The tactics are proven solutions to recurring software design problems; however, the success of their application depends on the experience of the architect and there are little research studies that describe in detail the application of tactics in software architecture. In this study we describe and apply a set of activities and tools to implement a set of security tactics in the design of a software system. The applicability of the tactics of security was achieved with the case study: tsunami warning-prevention system. The proposed approach is developed between the requirements and design phases of the software cycle and is guided by a backlog of unresolved risks. Besides, it was possible to characterize the actions required in the software architecture to implement security tactics and one notation scheme was adapted to facilitate the specification of these into the architecture.

Acknowledgments

This work was partially supported by UTFSM (grant DGIP 24.12.50, CCTVal Basal FB0821), AGCI Chile Student Mobility Platform of the Pacific Alliance (PhD research internship 2013).

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[2] N. Rozanski and E. Woods, *Software systems architecture: working with stalkeholders using viewpoints and perspectives.*, 2nd ed. Addison-Wesley Educational Publishers., 2012.

[3] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? a model and annotation," *J. Syst. Softw.*, vol. 83, no. 10, pp. 1735–1758, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.04.067

[4] S. Kim, D.-K. Kim, L. Lu, and S.-Y. Park, "A tactic-based approach to embodying non-functional requirements into software architectures," in *Enterprise Distributed Object Computing Conference, 2008. EDOC '08. 12th International IEEE*, Sept 2008, pp. 139–148.

[5] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[6] E. B. Fernandez, M. M. Larrondo-Petrie, T. Sorgente, and M. Vanhilst, *Integrating Security and Software Engineering: Advances and Future Visions*, H. Mouratidis and P. Giorgini, Eds. Idea Group Inc. Global, Aug. 2007. [Online]. Available: http://www.igi-global.com/chapter/methodology-develop-secure-systems-using/24052/

[7] N. Mayer, A. Rifaut, and E. Dubois, "Towards a risk-based security requirements engineering framework," in *Workshop on Requirements Engineering for Software Quality REFSQ*, 2005.

[8] C. B. Haley, R. C. Laney, and B. Nuseibeh, "Deriving security requirements from crosscutting threat descriptions," in *Proceedings of the 3rd International Conference on Aspect-oriented Software Development*, ser. AOSD '04. New York, NY, USA: ACM, 2004, pp. 112–121. [Online]. Available: http://doi.acm.org.ezproxy.uniandes.edu.co:8080/10.1145/976270.976285

[9] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software*, vol. 20, no. 1, pp. 58–66, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1109/MS.2003.1159030

[10] G. Sindre and A. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005. [Online]. Available: http://dx.doi.org/10.1007/s00766-004-0194-4

[11] F. Braz, E. Fernandez, and M. VanHilst, "Eliciting security requirements through misuse activities," in *Database and Expert Systems Application, 2008. DEXA '08. 19th International Workshop on*, Sept 2008, pp. 328–333.

[12] E. Fernandez, M. VanHilst, M. Larrondo Petrie, and S. Huang, "Defining security requirements through misuse actions," in *Advanced Software Engineering: Expanding the Frontiers of Software Technology*, ser. IFIP International Federation for Information Processing, S. Ochoa and G.-C. Roman, Eds. Springer US, 2006, vol. 219, pp. 123–137. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-34831-5_10

[13] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–153, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1109/TSE.2007.70754

[14] B. W. Lampson, "Computer security in the real world," *Computer*, vol. 37, no. 6, pp. 37–46, June 2004.

[15] F. Redmill, M. Chudleigh, and J. Catmur, *System Safety : HAZOP and Software HAZOP*. John Wiley & Sons, Jul. 1999. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471982806

[16] J. D. Andrews and T. R. Moss, *Realibility and Risk Assessment*, 1st ed. Longamn Group UK, 1993.

[17] A. Bouti and D. A. Kadi, "A state-of-the-art review of fmea/fmeca," *International Journal of Reliability, Quality and Safety Engineering*, vol. 01, no. 04, pp. 515–543, 1994. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218539394000362

[18] B. Littlewood, "A reliability model for systems with markov structure," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 24, no. 2, pp. 172 – 177, 1975. [Online]. Available: http://www.jstor.org/stable/2346564

[19] E. Hollnagel, *Human Reliability Analysis: Context and Control*, 1st ed., ser. Computers and People, B. R. Gaines and A. F. Monk, Eds. Academic Press, 1994.

[20] B. Barber and J. Davey, *MED-INFO 92*, 1st ed., K. C. Lun, D. P., and P. T. Rienhoff, Eds. Amsterdam: North Holland Publishig Co., 1992.

[21] P. J. Brooke and R. F. Paige, "Fault trees for security system design and analysis," *Computers Security*, vol. 22, no. 3, pp. 256 – 264, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404803003134

[22] I. N. Fovino, M. Masera, and A. D. Cian, "Integrating cyber attacks within fault trees," *Reliability Engineering System Safety*, vol. 94, no. 9, pp. 1394 – 1402, 2009, {ESREL} 2007, the 18th European Safety and Reliability Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832009000337

[23] T. W. DeLong, "A fault tree manual," Texas AM University, Tech. Rep., 1970.

[24] R. Shirey, "Internet security glossary," RFC Editor, United States, 2000.

[25] C. Cecioni, G. Bellotti, A. Romano, A. Abdolali, P. Sammarco, and L. Franco, "Tsunami early warning system based on real-time measurements of hydro-acoustic waves," *Procedia Engineering*, vol. 70, no. 0, pp. 311 – 320, 2014, 12th International Conference on Computing and Control for the Water Industry, {CCWI2013}. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187770581400037X

[26] J. Klotz, D. Angermann, G. W. Michel, R. Porth, C. Reigber, J. Reinking, J. Viramonte, R. Perdomo, V. H. Rios, S. Barrientos, R. Barriga, and O. Cifuentes, "Gps-derived deformation of the central andes including the 1995 antofagasta mw = 8.0 earthquake," *pure and applied geophysics*, vol. 154, no. 3-4, pp. 709–730, 1999. [Online]. Available: http://dx.doi.org/10.1007/s000240050249

[27] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented Software Architecture: A System of Patterns*. New York, NY, USA: John Wiley & Sons, Inc., 1996.

[28] R. Noël, G. Pedraza-García, H. Astudillo, and E. B. Fernández, "An exploratory comparison of security patterns and tactics to harden systems," in *Proceedings of the 11th Workshop on Experimental Software Engineering (ESELAW 2014)*, ser. CibSE 2014, 2014.