STAKEHOLDERS OF SOFTWARE ARCHITECTURE

BY PANKAJ KAMTHAN

1. INTRODUCTION

In recent years, the recognition and role of stakeholders in software engineering has received attention [McManus, 2004].

It is known that **stakeholder theory** is a part of **stakeholder management** that, in turn, is a part of **strategic management** [Mitchell, Agle, Wood, 1997].

This document introduces elements of stakeholder theory relevant to software architecture.

2. DEFINITION OF STAKE

There are a number of definitions of a stake. The following definition is specific to a software project:

Definition [Stake] [McManus, 2004, Page 23]. A stake can be described as an interest or share in a project undertaking to achieve **business, technical, or social goals**.

3. DEFINITION OF STAKEHOLDER

There are a number of definitions of a stakeholder, including the following:

Definition [Stakeholder] [ISO/IEC/IEEE, 2011]. A [stakeholder is] individual, team, organization, or classes thereof, having an interest in a system.

The next definition refines the relationship between a stakeholder and a system.

Definition [Stakeholder] [Wikipedia]. A stakeholder is any person or group, who can be **positively or negatively** impacted by, or cause an impact on the actions of an organization.

OBSERVATIONS

- A stakeholder need **not** be a single individual. A stakeholder need not be a user (of the software system).

- It is evident that a user is a human actor (in the use case sense) and, in turn, a human actor is a stakeholder. The converse in both cases is **not** necessarily true.

4. MOTIVATION FOR STUDYING STAKEHOLDERS AS PER SOFTWARE ARCHITECTURE

There are a number of reasons for studying stakeholders as part of the coverage on software architecture.

RELEVANCE

The relevance (or purpose) of software architecture is determined by stakeholders of the software project. In an industrial setting, the purpose is often related to **business goals** [Sangwan, 2015, Chapter 2].

For example, a business goal could be big (**"Establish Position in International Markets in Eastern Europe and South Asia"**) or small (**"Save Maintenance Costs by Taking Advantage of Software-as-a-Service"**).

DECISIONS

The design decisions that compose a software system's architecture, individually and collectively, define what is and is not a relevant part of the problem [Booch, 2007]. These **decisions are made by stakeholders** of the software project.

VIEWS

A view of software architecture should be relevant to some stakeholder [IEEE, 2000]. Therefore, identifying relevant stakeholders is a **prerequisite** to **eliciting views**.

5. MODELS FOR STAKEHOLDERS OF SOFTWARE ARCHITECTURE

There are a number of different philosophies for relevant stakeholders of software architecture. They have resulted in different models that are presented here **chronologically**.

STAKEHOLDERS OF SOFTWARE ARCHITECTURE: MODEL 1

In [IEEE, 2000], the following stakeholder classes of software architecture have been identified:

- User
- Acquirer
- Developer
- **Maintainer**

STAKEHOLDERS OF SOFTWARE ARCHITECTURE: MODEL 2

In [Bass, Clements, Kazman, 2003, Table 9.2], the following stakeholder classes of software architecture have been identified:

- Customer
- Software Architect
- User

It has been pointed out that a software architect must **identify and engage** the stakeholders to solicit their needs and expectations [Bass, Clements, Kazman, 2003, Section 1.1].

STAKEHOLDERS OF SOFTWARE ARCHITECTURE: MODEL 3

In [Rozanski, Woods, 2005, Chapter 9], the following stakeholder classes of software architecture have been identified:

- Acquirer
- Assessor
- Communicator
- Developer
- **Maintainer**
- Supplier
- Support Staff
- System Administrator
- Tester
- User

STAKEHOLDERS OF SOFTWARE ARCHITECTURE: MODEL 4

In [Clements, Bachmann, Bass, Garlan, Ivers, Little, Merson, Nord, Stafford, 2010, Page 326], the following stakeholder classes of software architecture have been identified:

- Project Manager
- Member of Development Team
- Tester
- Integrator
- Designer of Other System
- **Maintainer**
- **Product-Line Application Builder**
- Customer
- End User
- Analyst
- Infrastructure Support Personnel
- New Stakeholder
- Current and **Future Architect**

For example, a **project manager** may use the software architecture description to **organize a project** and **distribute responsibilities** among the team members, or to **train new team members**, and a **maintainer** may use the software architecture description for understanding the system (as an initial step towards making changes to the system) [Sangwan, 2015, Section 6.1].

STAKEHOLDERS OF SOFTWARE ARCHITECTURE: MODEL 5

In the **ISO/IEC/IEEE 42010 Standard** [ISO/IEC/IEEE, 2011], the role of a stakeholder is critical to software architecture (and its description), as shown in a UML Class Diagram in Figure 1.

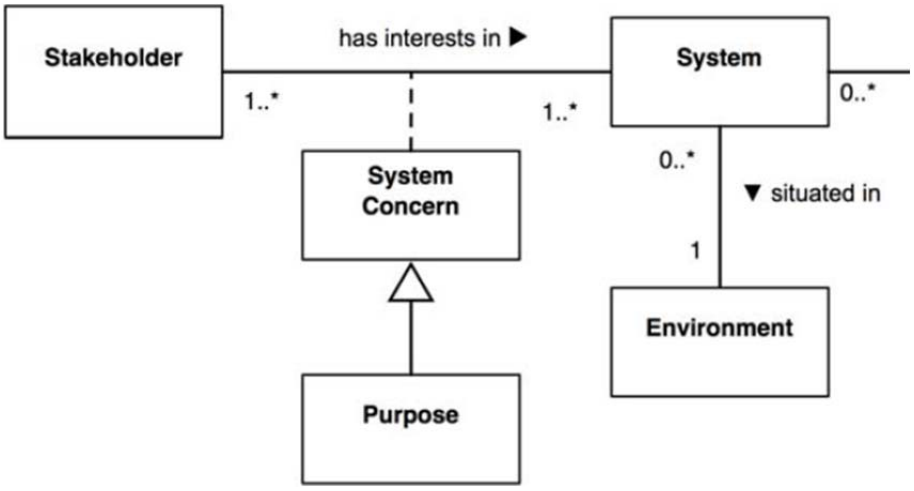

Figure 1. The placement of stakeholders in relation to other aspects of a software system. (Source: [ISO/IEC/IEEE, 2011].)

In [ISO/IEC/IEEE, 2011], the following stakeholder classes of software architecture have been identified:

- User
- Operator
- Acquirer
- Owner
- Supplier
- Developer
- Builder
- **Maintainer**

This list, evidently, **subsumes** that of Model 1.

6. EXAMPLE OF STAKEHOLDERS OF SOFTWARE ARCHITECTURE

In [McGee, Eklund, Lundin, 2010], a **Global Vehicle Control System** has been studied, and the following stakeholder classes of software architecture have been identified:

- Driver (User)
- Marketer (Acquirer)
- Engineer (Developer)
- Repair Technician (Maintainer)
- Producer

The list is inspired by the stakeholders given in [IEEE, 2000], as suggested by the entries in the parenthesis.

7. NEGATIVE STAKEHOLDERS

It is **not** always the case that a software system is used in the manner it is intended by software engineers [Rost, Glass, 2011].

This motivates the following definition:

Definition [Negative Stakeholder] [Alexander, Beus-Dukic, 2009, Page 33]. A negative stakeholder is, intentionally or incidentally, a threat to the software project.

For every software project, there exists a negative stakeholder.

7.1. EXAMPLES OF NEGATIVE STAKEHOLDERS FOR INTERACTIVE SYSTEMS

The opportunity of use of a software system raises the possibility of both a positive use and the potential for a **negative use** (or misuse). A negative use usually manifests itself as a **security, privacy, or safety** concern.

For example, the **'negative' uses of a software system** can be modeled by a 'negative' use case (specifically, **abuse case** [McDermott, Fox, 1999] or **misuse case** [Sindre, Opdahl, 2001; Sindre, Opdahl, 2005]).

In such a case, a **human actor of a misuse case** [Sindre, Opdahl, 2001; Sindre, Opdahl, 2005] or, equivalently, a **negative user** [Courage, Baxter, 2005], is a negative stakeholder.

A **black hat hacker** is a kind of **hacker** as well as a negative user. Therefore, a black hat hacker is a kind of negative stakeholder. Figure 2 presents an example.

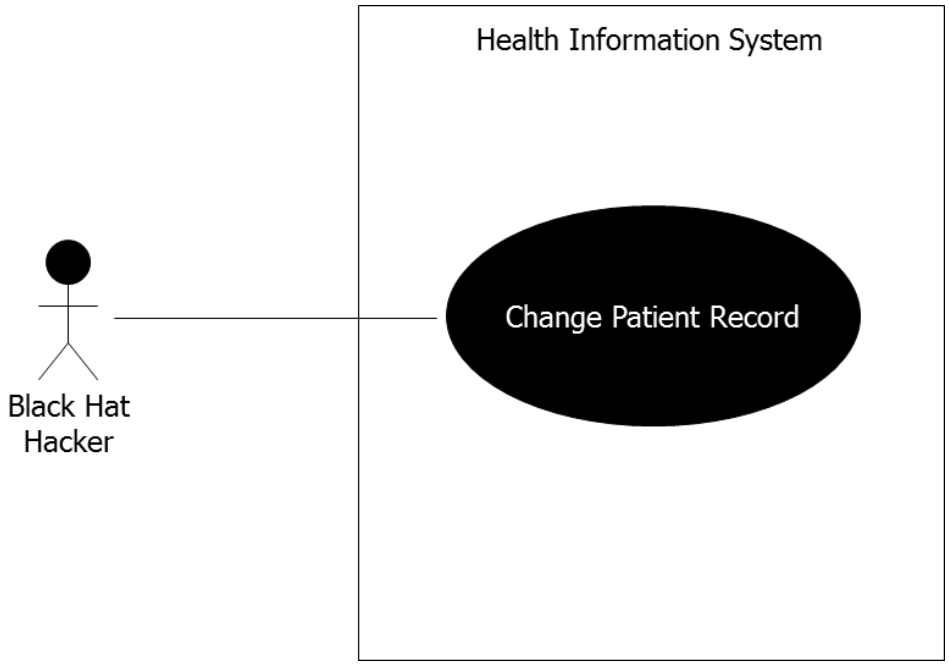

Figure 2. An example of a negative stakeholder for a health information system.

8. IDENTIFICATION OF STAKEHOLDERS

There are a number of sources for the identification of stakeholders [Pacheco, Garcia, 2008; Pacheco, Garcia, 2009; Pacheco, Garcia, 2012]. However, a 'standard' is yet to emerge.

It is likely that **multiple approaches** for the identification are needed to elicit a complete list of stakeholders.

The commonly-used approaches are:

- **Organizational Chart.** An organization (say, a company) may have a chart that provides the organization (say, hierarchy) of its employees.

- **Similar Projects.** A **record** of past, related experience on an old (software) project can be useful in identifying stakeholders for a new (software) project.

- **Context Analysis.** A documentation of the context (and/or context of use) of a software system can be useful in identifying stakeholders. (This is based on the assumption that every software system is **developed, delivered, and deployed** not in isolation, but in some **context**.)

REMARKS

The approaches that are not common, but may become so, include **crowdsourcing**.

ACKNOWLEDGEMENT

The inclusion of images from external sources is only for non-commercial educational purposes, and their use is hereby acknowledged.

REFERENCES

[Alexander, Beus-Dukic, 2009] Discovering Requirements: How to Specify Products and Services. By I. Alexander, L. Beus-Dukic. John Wiley and Sons. 2009.

[Bass, Clements, Kazman, 2003] Software Architecture in Practice. By L. Bass, P. Clements, R. Kazman. Second Edition. Addison-Wesley. 2003.

[Clements, Bachmann, Bass, Garlan, Ivers, Little, Merson, Nord, Stafford, 2010] Documenting Software Architectures: Views and Beyond. By P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Second Edition. Addison-Wesley. 2010.

[Courage, Baxter, 2005] Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques. By C. Courage, K. Baxter. Elsevier. 2005.

[IEEE, 2000] IEEE Standard 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society. 2000.

[ISO/IEC/IEEE, 2011] ISO/IEC/IEEE 42010:2011. Systems and Software Engineering -- Architecture Description. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2011.

[McDermott, Fox, 1999] Using Abuse Case Models for Security Requirements Analysis. By J. McDermott, C. Fox. The Fifteenth Annual Computer Security Applications Conference (ACSAC 1999). Scottsdale, U.S.A. December 6-10, 1999.

[McGee, Eklund, Lundin, 2010] Stakeholder Identification and Quality Attribute Prioritization for a Global Vehicle Control System. By R. McGee, U. Eklund, M. Lundin. The Fourth European Conference on Software Architecture (ECSA 2010). Copenhagen, Denmark. August 23-26, 2010.

[McManus, 2004] Managing Stakeholders in Software Development Projects. By J. McManus. Butterworth-Heinemann. 2004.

[Mitchell, Agle, Wood, 1997] Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts. By R. K. Mitchell, B. R. Agle, D. J. Wood. The Academy of Management Review. Volume 22. Number 4. 1997. Pages 853-886.

[Pacheco, Garcia, 2008] Stakeholder Identification Methods in Software Requirements: Empirical Findings Derived from a Systematic Review. By C. Pacheco, I. Garcia. The Third International Conference on Software Engineering Advances (ICSEA 2008). Sliema, Malta. October 26-31, 2008.

[Pacheco, Garcia, 2009] Effectiveness of Stakeholder Identification Methods in Requirements Elicitation: Experimental Results Derived from a Methodical Review. By C. Pacheco, I. Garcia. The Eighth IEEE/ACIS International Conference on Computer and Information Science (ICIS 2009). Shanghai, China. June 1-3, 2009.

[Pacheco, Garcia, 2012] A Systematic Literature Review of Stakeholder Identification Methods in Requirements. By C. Pacheco, I. Garcia. Journal of Systems and Software. Volume 85. Issue 9. 2012. Pages 2171-2181.

[Rost, Glass, 2011] The Dark Side of Software Engineering: Evil on Computing Projects. By J. Rost, R. L. Glass. John Wiley and Sons. 2011.

[Rozanski, Woods, 2005] Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. By N. Rozanski, E. Woods. Addison-Wesley. 2005.

[Sangwan, 2015] Software and Systems Architecture in Action. By R. S. Sangwan. CRC Press. 2015.

[Sindre, Opdahl, 2001] Capturing Security Requirements through Misuse Cases. By G. Sindre, A. L. Opdahl. The Fourteenth Norwegian Informatics Conference (NIK 2001). Tromsø, Norway. November 26-28, 2001.

[Sindre, Opdahl, 2005] Eliciting Security Requirements by Misuse Cases. By G. Sindre, A. L. Opdahl. Requirements Engineering. Volume 10. Number 1. 2005. Pages 34-44.