Andre Lutfiansyah – 211511005

2A – D3 Teknik Informatika

Restablishing the Groundwork and Eliciting Requirement

Sommerville and Sawyer mendefinisikan stakeholder sebagai seseorang yang diuntungkan secara langsung ataupun secara tidak langsung.

- Business operations manage
- Product managers
- Marketing people
- Ineternal and external customers
- End users
- Consultants
- Product engineers
- Software engineers
- Maintenance engineers

Next Page 😊, rangkuman saya yang ini hanya menghighlight yang menurut saya penting.

SOFTWARE TOOLS

Requirements Engineering

Objective: Requirements engineering tools assist in requirements gathering, requirements modeling, requirements management, and requirements validation.

Mechanics: Tool mechanics vary. In general, requirements engineering tools build a variety of graphical (e.g., UML) models that depict the informational, functional, and behavioral aspects of a system. These models form the basis for all other activities in the software process.

Representative Tools:[7]
A reasonably comprehensive (and up-to-date) listing of requirements engineering tools can be found at the Volvere Requirements resources site at **www.volere.co.uk/tools.htm**. Requirements modeling tools are discussed in Chapters 6 and 7. Tools noted below focus on requirement management.

EasyRM, developed by Cybernetic Intelligence GmbH (**www.easy-rm.com**), builds a project-specific dictionary/glossary that contains detailed requirements descriptions and attributes.

Rational RequisitePro, developed by Rational Software (**www-306.ibm.com/software/awdtools/reqpro/**), allows users to build a requirements database; represent relationships among requirements; and organize, prioritize, and trace requirements.

Many additional requirements management tools can be found at the Volvere site noted earlier and at **www.jiludwig.com/Requirements_Management_Tools.html**.

5.2 ESTABLISHING THE GROUNDWORK

In an ideal setting, stakeholders and software engineers work together on the same team.[8] In such cases, requirements engineering is simply a matter of conducting meaningful conversations with colleagues who are well-known members of the team. But reality is often quite different.

Customer(s) or end users may be located in a different city or country, may have only a vague idea of what is required, may have conflicting opinions about the system to be built, may have limited technical knowledge, and may have limited time to interact with the requirements engineer. None of these things are desirable, but all are fairly common, and you are often forced to work within the constraints imposed by this situation.

A *stakeholder* is anyone who has a direct interest in or benefits from the system that is to be developed.

In the sections that follow, I discuss the steps required to establish the groundwork for an understanding of software requirements—to get the project started in a way that will keep it moving forward toward a successful solution.

5.2.1 Identifying Stakeholders

Sommerville and Sawyer [Som97] define a stakeholder as "anyone who benefits in a direct or indirect way from the system which is being developed." I have already

7 Tools noted here do not represent an endorsement, but rather a sampling of tools in this category. In most cases, tool names are trademarked by their respective developers.

8 This approach is strongly recommended for projects that adopt an agile software development philosophy.

setiap stakeholder pasti puuya pandangan dan pendapat masing masing yang mana bisa saja berhubungan ataupun tidak berhubungan, naha maka dengan itu adanya nanti kolaborasi dari tiap stakeholder

identified the usual suspects: business operations managers, product managers, marketing people, internal and external customers, end users, consultants, product engineers, software engineers, support and maintenance engineers, and others. Each stakeholder has a different view of the system, achieves different benefits when the system is successfully developed, and is open to different risks if the development effort should fail.

At inception, you should create a list of people who will contribute input as requirements are elicited (Section 5.3). The initial list will grow as stakeholders are contacted because every stakeholder will be asked: "Whom else do you think I should talk to?"

5.2.2 Recognizing Multiple Viewpoints

banyak berbagai pandangan dari beberapa stakeholder yang mana bisa disatukan ataupun tidak

Quote:

"Put three stakeholders in a room and ask them what kind of system they want. You're likely to get four or more different opinions."

Author unknown

Because many different stakeholders exist, the requirements of the system will be explored from many different points of view. For example, the marketing group is interested in functions and features that will excite the potential market, making the new system easy to sell. Business managers are interested in a feature set that can be built within budget and that will be ready to meet defined market windows. End users may want features that are familiar to them and that are easy to learn and use. Software engineers may be concerned with functions that are invisible to nontechnical stakeholders but that enable an infrastructure that supports more marketable functions and features. Support engineers may focus on the maintainability of the software.

contoh pandangan dari beberapa stakeholder

Each of these constituencies (and others) will contribute information to the requirements engineering process. As information from multiple viewpoints is collected, emerging requirements may be inconsistent or may conflict with one another. You should categorize all stakeholder information (including inconsistent and conflicting requirements) in a way that will allow decision makers to choose an internally consistent set of requirements for the system.

5.2.3 Working toward Collaboration

If five stakeholders are involved in a software project, you may have five (or more) different opinions about the proper set of requirements. Throughout earlier chapters, I have noted that customers (and other stakeholders) must collaborate among themselves (avoiding petty turf battles) and with software engineering practitioners if a successful system is to result. But how is this collaboration accomplished?

The job of a requirements engineer is to identify areas of commonality (i.e., requirements on which all stakeholders agree) and areas of conflict or inconsistency (i.e., requirements that are desired by one stakeholder but conflict with the needs of another stakeholder). It is, of course, the latter category that presents a challenge.

INFO

Using "Priority Points"

One way of resolving conflicting each (from his or her viewpoint) by spending one or

Exciting requirements. These features go beyond the customer's expectations and prove to be very satisfying when present. For example, software for a new mobile phone comes with standard features, but is coupled with a set of unexpected capabilities (e.g., multitouch screen, visual voice mail) that delight every user of the product. fitur fitur

WebRef

Useful information on QFD can be obtained at **www.qfdi.org**.

Although QFD concepts can be applied across the entire software process [Par96a], specific QFD techniques are applicable to the requirements elicitation activity. QFD uses customer interviews and observation, surveys, and examination of historical data (e.g., problem reports) as raw data for the requirements gathering activity. These data are then translated into a table of requirements—called the *customer voice table*—that is reviewed with the customer and other stakeholders. A variety of diagrams, matrices, and evaluation methods are then used to extract expected requirements and to attempt to derive exciting requirements [Aka04].

5.3.3 Usage Scenarios

As requirements are gathered, an overall vision of system functions and features begins to materialize. However, it is difficult to move into more technical software engineering activities until you understand how these functions and features will be used by different classes of end users. To accomplish this, developers and users can create a set of scenarios that identify a thread of usage for the system to be constructed. The scenarios, often called *use cases* [Jac92], provide a description of how the system will be used. Use cases are discussed in greater detail in Section 5.4.

James Thurber

software to be built. In addition, the questions identify the measurable benefit of a successful implementation and possible alternatives to custom software development.

The next set of questions enables you to gain a better understanding of the problem and allows the customer to voice his or her perceptions about a solution:

? What questions will help you gain a preliminary understanding of the problem?

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

5.3.2 Quality Function Deployment

KEY POINT

QFD defines requirements in a way that

Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software. QFD "concentrates on maximizing customer satisfaction from the software engineering process" [Zul92]. To accomplish this, QFD emphasizes an understanding of what is valuable

The final set of questions focuses on the effectiveness of the communication activity itself. Gause and Weinberg [Gau89] call these "meta-questions" and propose the following (abbreviated) list:

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

Quote:

"He who asks a question is a fool for five minutes; he who does not ask a question is a fool forever."

Chinese proverb

These questions (and others) will help to "break the ice" and initiate the communication that is essential to successful elicitation. But a question-and-answer meeting format is not an approach that has been overwhelmingly successful. In fact, the Q&A session should be used for the first encounter only and then replaced by a requirements elicitation format that combines elements of problem solving, negotiation, and specification. An approach of this type is presented in Section 5.3.

5.3 Eliciting Requirements

Requirements elicitation (also called *requirements gathering*) combines elements of problem solving, elaboration, negotiation, and specification. In order to encourage a collaborative, team-oriented approach to requirements gathering, stakeholders work together to identify the problem, propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirements [Zah90].[9]

5.3.1 Collaborative Requirements Gathering

Many different approaches to collaborative requirements gathering have been proposed. Each makes use of a slightly different scenario, but all apply some variation on the following basic guidelines:

? What are the basic guidelines for conducting a collaborative requirements gathering meeting?

- Meetings are conducted and attended by both software engineers and other stakeholders.
- Rules for preparation and participation are established.
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
- A "facilitator" (can be a customer, a developer, or an outsider) controls the meeting.
- A "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room, or virtual forum) is used.

9 This approach is sometimes called a *facilitated application specification technique* (FAST).

Quote:

"We spend a lot of time—the majority of project effort—not implementing or testing, but trying to decide what to build."

Brian Lawrence

The goal is to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements in an atmosphere that is conducive to the accomplishment of the goal. To better understand the flow of events as they occur, I present a brief scenario that outlines the sequence of events that lead up to the requirements gathering meeting, occur during the meeting, and follow the meeting.

During inception (Section 5.2) basic questions and answers establish the scope of the problem and the overall perception of a solution. Out of these initial meetings, the developer and customers write a one- or two-page "product request."

A meeting place, time, and date are selected; a facilitator is chosen; and attendees from the software team and other stakeholder organizations are invited to participate. The product request is distributed to all attendees before the meeting date.

WebRef

Joint Application Development (JAD) is a popular technique for requirements gathering. A good description can be found at **www.carolla.com/wp-jad.htm**.

As an example,[10] consider an excerpt from a product request written by a marketing person involved in the *SafeHome* project. This person writes the following narrative about the home security function that is to be part of *SafeHome:*

> Our research indicates that the market for home management systems is growing at a rate of 40 percent per year. The first *SafeHome* function we bring to market should be the home security function. Most people are familiar with "alarm systems" so this would be an easy sell.
>
> The home security function would protect against and/or recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, carbon monoxide levels, and others. It'll use our wireless sensors to detect each situation. It can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

narative pembukaan untuk rapat

ADVICE

If a system or product will serve many users, be absolutely certain that requirements are elicited from a representative cross section of users. If only one user defines all requirements, acceptance risk is high.

In reality, others would contribute to this narrative during the requirements gathering meeting and considerably more information would be available. But even with additional information, ambiguity would be present, omissions would likely exist, and errors might occur. For now, the preceding "functional description" will suffice.

While reviewing the product request in the days before the meeting, each attendee is asked to make a list of objects that are part of the environment that surrounds the system, other objects that are to be produced by the system, and objects that are used by the system to perform its functions. In addition, each attendee is asked to make another list of services (processes or functions) that manipulate or interact with the objects. Finally, lists of constraints (e.g., cost, size, business rules) and performance criteria (e.g., speed, accuracy) are also developed. The attendees are informed that the lists are not expected to be exhaustive but are expected to reflect each person's perception of the system.

10 This example (with extensions and variations) is used to illustrate important software engineering methods in many of the chapters that follow. As an exercise, it would be worthwhile to conduct your own requirements gathering meeting and develop a set of lists for it.

5.3.2 Quality Function Deployment

KEY POINT
QFD defines requirements in a way that maximizes customer satisfaction.

Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software. QFD "concentrates on maximizing customer satisfaction from the software engineering process" [Zul92]. To accomplish this, QFD emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process. QFD identifies three types of requirements [Zul92]:

Normal requirements. The objectives and goals that are stated for a product or system during meetings with the customer. If these requirements are present, the customer is satisfied. Examples of normal requirements might be requested types of graphical displays, specific system functions, and defined levels of performance. sesuai dengan requirement customer

ADVICE
Everyone wants to implement lots of exciting requirements, but be careful. That's how "requirements creep" sets in. On the other hand, exciting requirements lead to a breakthrough product!

Expected requirements. These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause for significant dissatisfaction. Examples of expected requirements are: ease of human/machine interaction, overall operational correctness and reliability, and ease of software installation. persyaratan yang tersirat pada product yang tidak secara eksplisit dinyatakan oleh customer, maka harus bisa menganalisis dengan baik hingga dapat mengambil requirement yang tidak dijelaskan secara gamblang

Exciting requirements. These features go beyond the customer's expectations and prove to be very satisfying when present. For example, software for a new mobile phone comes with standard features, but is coupled with a set of unexpected capabilities (e.g., multitouch screen, visual voice mail) that delight every user of the product. fitur fitur

WebRef
Useful information on QFD can be obtained at **www.qfdi.org**.

Although QFD concepts can be applied across the entire software process [Par96a], specific QFD techniques are applicable to the requirements elicitation activity. QFD uses customer interviews and observation, surveys, and examination of historical data (e.g., problem reports) as raw data for the requirements gathering activity. These data are then translated into a table of requirements—called the *customer voice table*—that is reviewed with the customer and other stakeholders. A variety of diagrams, matrices, and evaluation methods are then used to extract expected requirements and to attempt to derive exciting requirements [Aka04].

5.3.3 Usage Scenarios

As requirements are gathered, an overall vision of system functions and features begins to materialize. However, it is difficult to move into more technical software engineering activities until you understand how these functions and features will be used by different classes of end users. To accomplish this, developers and users can create a set of scenarios that identify a thread of usage for the system to be constructed. The scenarios, often called *use cases* [Jac92], provide a description of how the system will be used. Use cases are discussed in greater detail in Section 5.4.

5.3.4 Elicitation Work Products

The work products produced as a consequence of requirements elicitation will vary depending on the size of the system or product to be built. For most systems, the work products include

What information is produced as a consequence of requirements gathering?

- A statement of need and feasibility. • Pernyataan kebutuhan dan kelayakan
- A bounded statement of scope for the system or product. • Pernyataan ruang lingkup terbatas untuk sistem atau produk.
- A list of customers, users, and other stakeholders who participated in requirements elicitation. list customer, users dan stakeholder yang ikut dalam requrement elicitation
- A description of the system's technical environment. deskripsi dari sistem technical environment
- A list of requirements (preferably organized by function) and the domain constraints that apply to each. daftar requirement dan domain yang berlaku pada masing masing
- A set of usage scenarios that provide insight into the use of the system or product under different operating conditions. • Satu set skenario penggunaan yang memberikan wawasan tentang penggunaan sistem atau Produk dalam kondisi operasi yang berbeda.
- Any prototypes developed to better define requirements. • Prototipe apa pun yang dikembangkan untuk menentukan persyaratan yang lebih baik

Each of these work products is reviewed by all people who have participated in requirements elicitation.