CS 6361 — Requirements Engineering, Fall 2004

Project 1: First Iteration - Requirements Elicitation: Initial Understanding

Due: 5:30pm Monday September 27 (in class)

> *The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later. [Brooks, 1987]*

I. Summary

A facility for scheduling meetings has many potential applications, such as scheduling courses and flights, room assignments at hospitals and hotels, scheduling national and international meetings, logistics, job scheduling in production systems, as well as command and control systems. Such a facility can also be used in allocating transmission lines, buffers and routers in computer networks. Etc.

The particular type of systems this project is intended for is supporting people to schedule their meetings. Many software vendors are eager to offer such a system, especially one with a powerful vantage point (cf., Microsoft, IBM-Lotus, etc.). In particular, SynergySoft, Inc. aims to provide such a facility which would outperform any such system that is currently available in the highly competitive market.

The company has gathered some initial requirements from potential customers. However, the company is well aware that they haven't yet clearly characterized what their customers really want, not to mention who their *real* customers might be. Consequently, the requirements definition is only preliminary, sketch, imprecise, incomplete and possibly inconsistent. Based on its past experience, however, the company is also well aware that getting the right requirements the first time will be the barometer to successfully completing the entire development effort, reducing production time, and to keeping up its well-established reputation and ultimately to satisfying their workforce and customers.

Due to this criticality, SynergySoft, Inc. is looking to a renowned consulting firm for help. As requirements engineers of the consulting firm, you are to deliver a detailed requirements description which captures *real* customers' *real* needs/wants as *precisely, concisely and conceptually* as possible.

II. SDMS - Synergy Distributed Meeting Scheduler — Preliminary Definition

Enterprise Requirements: Stakeholders, Functional and Non-Functional Objectives

In the application domain, meetings are typically arranged in the following manner. A *meeting initiator* will ask all *potential meeting attendees* for the following information based on their personal agenda:

- a set of dates on which they cannot attend the meeting (hereafter referred to as *exclusion set*; and
- a set of dates on which they would prefer the meeting to take place (hereafter referred to as *preference set*);

A meeting date shall be defined perhaps by a pair (calendar date, time period). The exclusion and preference sets should be contained in some time interval prescribed by the meeting initiator (hereafter referred to as *date range*).

The initiator could also ask, in a friendly manner, *active participants* to provide any special equipment requirements on the meeting location (e.g., overhead projector, workstation, network connection, telephone, etc.). She may also ask *important participants* to state preferences about the meeting location.

The proposed meeting date should belong to the stated date range and to none of the exclusion sets; furthermore it should ideally belong to as many preference sets as possible. The proposal should be made as early as possible. A *date conflict* occurs when no such date can be found. A conflict is strong when no date can be found within the date range and outside all exclusion sets; it is weak when dates can be found within the date range and outside all exclusion sets, but no date can be found at the intersection of all preference sets.

Conflicts can be resolved in several ways, including:

- the initiator extends the date range; and
- some participants remove some dates from their exclusion set.
- some participants withdraw from the meeting;
- some participants add some new dates to their preference set.

Each conflict resolution should be done as quickly as possible and with no more interactions than is really needed.

A meeting room must be available at the selected meeting date. It should meet the equipment requirements; furthermore it should ideally belong to one of the locations preferred by as many important participants as possible. **It is absolutely necessary, however, to allow each meeting to take place in a virtual place, e.g., through teleconferencing using laptop computers**. This flexibility is considered crucial in future. The number of negotiations should be kept minimal, but a new round of negotiation may be required when no such room can be found.

The meeting initiator can be one of the participants or some representative (e.g., a secretary).

System Requirements: Functional Requirements

The purpose of **SDMS** is to support the organization of meetings — that is, to determine, for each meeting request, a meeting date and location so that most of the intended participants will effectively participate.

SDMS shall assist users in the following activities:

- Monitor meetings, especially when they are held in a distributed manner;
- Plan meetings under the constraints expressed by participants (see domain theory);
- Replan a meeting to support the changing user constraints, for instance:
  - to modify the exclusion set, preference set and/or preferred location before a meeting date/location is proposed; and
  - to take some external constraints into account after a date and location have been proposed — e.g., due to the need to accommodate a more important meeting. — here, the original meeting date or location may then need to be changed; sometimes the meeting may even be cancelled.

In all cases some bound on replanning should be set up.

- Support conflict resolution according to resolution policies stated by the client;
- Manage all the interactions among participants required during the organization of the meeting, for instance:
  - to communicate requests;
  - to get replies even from participants not reacting promptly;
  - to support the negotiation and conflict resolution processes;
  - to make participants aware of what's going on during the planning process;
  - to keep participants informed about schedules and their changes; and
  - to make them confident about the *reliability* of the communications.

The meeting scheduler system must in general handle several meeting requests in parallel. Meeting requests can be competing when they overlap in time or space. Concurrency must thus be managed.

System Non-Functional Requirements

In meeting the functional requirements, *non-functional requirements* should also be taken account. They include:

- A meeting should be *accurately* monitored, especially when it is held in a virtual place. Here, *nomadicity* will then be important to consider;
- Replanning of a meeting should be done as *dynamically* and with as much *flexibility* as possible;
- The amount of interaction among participants (e.g., number and length of messages, amount of negotiation required) should be kept *minimal*;
- The intended system should *considerably reduce the amount of overhead* usually incurred in organizing meetings where potential attendees are distributed over many different places and communicate with each other, for example, via Internet;
- The system should reflect as closely as possible the way meetings are typically managed (see the domain theory above);
- The meeting date and location should be as *convenient* as possible, and available as *early* as possible, to all (potential) participants;
- The system should accommodate as much decentralized requests as possible; any authorized user should be able to request a meeting independently of her whereabouts;
- Physical constraints should not be broken — e.g., a person may not be at two different places at the same time; a meeting room may not be allocated to more than one meeting at the same time; etc.;
- The system should provide an appropriate level of *performance*:
  - the elapsed time between the submission of a meeting request and the determination of the corresponding date/location should me *minimal*; or

  - the elapsed time between the determination of a meeting date/location and the communication of this information to all participants concerned should be *minimal*; or
  - a lower bound should be fixed between the time at which the meeting date is determined and the time at which the meeting is actually taking place;
- *Privacy* rules should be enforced; a non-privileged participant should not be aware of constraints stated by other participants;
- The system should be *usable* by non-experts;
- The system should be *customizable* to professional as well as private meetings — these two modes of use are characterized by different restrictions on the time periods that may be allocated (e.g., meetings during office hours, private activities during leisure time);
- The system should be *flexible* enough to accommodate evolving data — e.g., the sets of concerned participants may be varying, the address at which a participant can be reached may be varying, etc.;
- The system should be easily *extensible* to accommodate the following typical variations:
  - handling of explicit priorities among dates in preference sets;
  - handling of explicit dependencies between meeting date and meeting location;
  - participation through delegation — a participant may ask another person to represent her/him at the meeting;
  - variations in date formats, address formats, interface language, etc.; and
  - partial *reuse* in other contexts — e.g., to help establish course schedule.

III. The Deliverable

Your description should be elegant and comprehensible. Your product should include the following sections.

1. **The Process** At least two aspects of the process:
   - There will typically be multiple stakeholders involved in the development. Identify the stakeholders and describe how your team was divided up in handling this multiplicity and what was the role of each team member.
   - There will typically be multiple sources of requirements for just about any kind of RE project. Identify and describe what sources were considered and to what extent.
2. **Semi-formal Requirements Description: Preliminary definition** Present two dependency graphs which reflect the informal preliminary definition as faithfully as possible. One dependency graph should reflect the enterprise requirements, and the other the system requirements (possibly consisting of two dependency graphs, one for software system FRs and the other for software system NFRs).
3. **Issues:** Describe any issues (e.g., incompleteness, inconsistency, ambiguity, redundancy) that you encountered while transforming the informal preliminary definition into the dependency graphs.

   Also describe how you resolved such issues. According to the preliminary definition, for example, system performance can be improved by option 1 *or* option 2 *or* option 3. What

does this mean? Describe what your choice is and why you have made that particular choice (i.e., because that particular choice is good with respect to some *reasons* — *design rationale*).

For another example, system extensibility can be enhanced additionally by allowing a participant to attend only part of the meeting. This is an example of requirements incompleteness.

In order to resolve the issues, you might need to use your own "creative imagination" but based on your teamwork.

4. **Improved Understanding** Now eliminate as many defects as possible from the two dependency graphs in **2**, according to your discussion in **3**. The two resulting dependency graphs then should be as error-free as possible. The differences between the two sets of dependency graphs should be clearly visible. It is recommended that you start using Rational Rose during this phase.

5. **A Prototype** Build a web-based prototype of SDMS.