Using the Tools in TRADE, II: Specification and Design of a Meeting Scheduler System

Roel Wieringa*

November 10, 1997

Contents

1 Introduction 2
2 System Purpose 2
3 System Environment 2
   3.1 Environment Purpose ........................................... 2
   3.2 Context model ................................................. 4
4 Requirements 4
   4.1 Behavioral requirements ........................................ 4
   4.2 Nonbehavioral requirements .................................... 6
5 Subject Domain 6
   5.1 Subject domain decomposition .................................. 6
   5.2 Subject domain properties ..................................... 7
   5.3 Subject domain dictionary ..................................... 7
6 Desired System Functions 8
   6.1 System function dictionary ..................................... 8
   6.2 System function specifications ................................. 8
   6.3 Other desired system properties ............................... 18
7 System Decomposition 18
   7.1 Conceptual decomposition .................................... 18
      7.1.1 Conceptual class model ................................ 18
      7.1.2 Function decomposition specifications .................. 19
      7.1.3 Traceability tables ...................................... 21
      7.1.4 Conceptual communication model ....................... 23
   7.2 Physical Decomposition ....................................... 24

*Faculty of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands. Email: roelw@cs.vu.nl. http://www.cs.vu.nl/~roelw. Tel: +31 20 4447771. Fax: +31 20 4447653.
8 Features
8.1 Locations ................................................. 25
8.2 Equipment .................................................. 25
8.3 Hypothetical subject domain changes ....................... 28
8.4 Changes to the scheduling objective ....................... 29
  8.4.1 Person status ........................................... 29
  8.4.2 Participant importance ................................. 31
  8.4.3 Function specifications ................................. 31
  8.4.4 Meeting significance .................................. 31
  8.4.5 Participant flexibility ................................. 32

9 Discussion and conclusion .................................. 32

1 Introduction

This report contains a solution of the meeting scheduler case set for the Ninth IEEE International Workshop on Software Specification and Design (IWSSD-9), Ise-Shima, Japan, April 16–18, 1998. The case description can be found in http://salab-www.cs.titech.ac.jp/iwssd9.html. The solution presented here uses tools from TRADE (Toolkit for Requirements and Design Engineering), a toolkit developed at the Vrije Universiteit, Amsterdam. I refer to the Meeting Scheduler System as MSS. This report makes no claim about the utility of meeting scheduler software.

The report is structured as follows. Sections 2 to 7 present a specification and design of a vanilla version of the MSS, that contains minimal functionality. The TRADE specification is contained in the figures; the body of the text gives some comments upon the specification. The reader can get an impression of the TRADE specification by skimming the figures whose caption starts with TRADE specification. Section 8 lists a number of additional features to the vanilla version and shows how they could be implemented. Section 9 winds up the report with a discussion and conclusions.

2 System Purpose

In the following, a meeting is any gathering of a group of people at a certain place and time, with a certain duration. The meetings may vary from faculty meetings to project or committee meetings. Figure 1 specifies the purpose of the MSS.

3 System Environment

3.1 Environment Purpose

The external environment of the MSS consist of a meeting initiator and potential participants. Figure 2 specifies the environment purpose of the system. The reason for wanting the system is that there is much overhead in organizing meetings where potential participants are geographically distributed. People don’t like this overhead, and this is what causes the need for the system. The system should help to reduce this overhead.
- **MSS purpose**: to support the selection of meeting dates and locations.

- **Responsibilities**:
  - Determine, for each meeting request, a meeting date and location given the available data about preferred and excluded dates of the participants, so that most of the participants will be able to participate.
  - Facilitate all communications among participants and the initiator required during the selection process.
  - Provide, during the selection process, information about the selection process to participants.

- **Exclusions**:
  - The MSS shall not support the organization of meetings (e.g., it shall not support the determination of meeting programs, maintain a database of participant registrations, not deal with payments, not support hotel bookings, is not able to store arrival dates, etc.).
  - The MSS shall not support the actual conduct of the meeting itself (e.g., it will not deal with last-minute payments, it will not provide on-line information about the meeting to the participants during the meeting, such as changes in program, map of conference dinner location, etc.).
  - The MSS shall not support post-meeting activities (e.g., it shall not maintain a database of actual meeting participation, it shall not print address labels of participants, etc.).

**Figure 1**: Specification of system purpose.

- **Environment purpose**: The environment purpose is to schedule meetings.

**Figure 2**: TRADE specification: Environment purpose.

**Figure 3**: TRADE specification: Context diagram.
3.2 Context model

The purpose of the MSS is to help the external environment achieve its purpose. To achieve the environment purpose, a number of interactions take place, in some of which with the MSS participates, as shown in the context diagram of figure 3. The environment consists of a meeting initiator, potential participants and a system administrator, all of whom interact with the MSS. The initiator may interact with one or more locations to find one that is available at the desired dates.

We include as much in the external environment of the system so that we can understand the purpose of the environment in the service of which the system is built. This purpose is to organize a meeting. So we include the other external entities that work towards this purpose, the initiator and the potential locations.

The context diagram is layered. In general, each layer exists in order to provide services to higher layers. Thus, purposes are found in higher layers, means in lower layers. The context diagram shows that the MSS provides communication services to potential participants. The initiator, MSS and potential locations jointly facilitate the meeting to be held by the participants. When we move to implementation of the MSS in section 7.2, a lower layer will be added.

4 Requirements

4.1 Behavioral requirements

Analysis of the statement of requirements as described in the problem statement leads to the list of system requirements shown in figure 4. These are things that external actors would like to do, stated in terms that have meaning with respect to their purpose. These terms do not refer to the MSS at all. The task of the MSS is to facilitate this desired behavior of the environment, and this desired behavior itself must not be defined in terms of the MSS. This is a first rough version of the desired functionality, which is sufficient to build a first version of the subject domain model and understand its boundary. Some functions listed above were uncovered when writing the more detailed specification in section 6.

This list must be verified and further elaborated with the users. For example, how does the system remind potential participants to respond? If it uses email and a potential participant is on holiday and has switched on an automatic answering service, then an email from the initiator receives answer but this is not yet a response. The deadline for a response then still holds. Or suppose the potential participant answers that he or she will send the requested date lists before the end of the week, how is the system to know this? The solution chosen above is that the initiator sets a deadline for responses, and then marks potential participants as having responded to a particular message. This gives the system sufficient information to send reminders to participants when the date is due. Other solutions are possible though, that should be discussed with the users because they lead to different ways of working.

Another issue is that of sending confirmations. One of the requirements is that the participants are made aware of the state of the process. (These are listed with the Other Properties in section 6.3.) This can be achieved by informing them of every significant change in the state of the process but this significantly increases the number of messages, which contradicts another requirement, which says that the number of messages should be
Figure 4: TRADE specification: Behavioral requirements specification.
The way of working of the participants and initiator should remain close to the way
meetings are typically organized.

- The participants should be confident about the reliability of communications between
  them and the initiator.
- Information about the meetings should be made available as early as possible to all
  participants.
- The amount of interaction among participants should be as small as possible.
- During the scheduling process, participants may be distributed over many different
  places.

Figure 5: TRADE specification: Nonbehavioral requirements specification.

kept as small as possible. The solution chosen above is to make the information available
to participants without bombarding them with it. Another solution would be to allow each
participant to set a preference about receiving status updates and confirmations and let the
system behave accordingly. This would have to be discussed with the users, also taking the
complexity and cost of the system into account.

4.2 Nonbehavioral requirements

These are requirements that cannot usefully be stated in terms of desired behavior of the
actors. They are listed in figure 5.

5 Subject Domain

5.1 Subject domain decomposition

The list of requirements given above leads to the subject domain decomposition given in
figure 6. This is a class diagram that expresses the following facts: Each person may play
the role of participant or initiator with respect to a meeting. One person can be initiator
and participant of the same meeting, but this need not be the case. Whatever the case,
the participant and initiator roles are distinct. Each participant participates in exactly
one meeting and a meeting can have any number of participants. Participants may have
preferred and excluded dates. Each meeting has a first and last date at which it can be
held.

It is impossible to find a subject domain model without listing the requirements first,
for otherwise there would be no criterion that tells us where to draw the subject domain
boundary. The requirements are all about actual, possible and desirable behavior of the
subject domain. On the other hand, without at least a preliminary understanding of the
subject domain, it is impossible to begin to understand the requirements. To break out
of this hermeneutical circle, the analyst starts with the requirements, using his or her
human capability to use background knowledge in interpreting communication, and to turn
the conversation explicitly to this background if there is an unclarity. As a result, the
Figure 6: TRADE specification: Subject domain decomposition.

- The list of preferred and excluded dates of each participant must fall within the date range of a meeting (prescriptive).

Figure 7: TRADE specification: Subject domain property.

requirements and subject domain model are specified in an interleaved manner, taking the requirements as a clue for relevance.

5.2 Subject domain properties

The class diagram of a subject domain represents a decomposition of the subject domain and some cardinality properties. A subject domain model may represent other properties in addition to cardinality properties. These must be listed separately from the class diagram, because they cannot be represented by it. Also, each property must be classified as descriptive or prescriptive. A descriptive property is a property present in the subject domain regardless of the presence of the system under development. As far as the system under development is concerned, these are laws of nature. In fact, using the phrase of Parnas & Madey (1995), descriptive properties of the subject domain are properties of nature or previously installed equipment. A prescriptive property is a property that someone must ensure holds in the subject domain. It describes the desired state of the subject domain.

There is one property of the MSS subject domain not represented by the class diagram, listed in figure 7.

5.3 Subject domain dictionary

The object classes in a subject domain model must be defined in a specification dictionary, that defines all important terms in the specification. This includes the important terms in the specifications of the system purpose, environment purpose, context model, requirements, subject domain and system functions. It is useful to merge all dictionary entries in one list and identify subdictionaries as views on this overall dictionary. One such view
could list all subject domain class names in alphabetical order; another would additionally list all attributes for each subject domain class, etc.

For each object class name, the classification criterion and identification criterion must be given, and possibly the concept of existence must be defined. The identification criterion tells us how we count instances of the class. Since objects have contingent, and mostly mutable states, we cannot rely on the state to tell distinct objects apart and identify the same objects in different states as the same. For objects, the identification criterion usually lies in its creation moment, from which moment the identity is preserved by a kind of persistence until the object is destroyed. For relationships, the identity is determined by the identity of its components.

We also sometimes make the concept of existence for a class explicit. This is done for the Meeting class below.

The classification criterion tells us by virtue of which property an object is an instance of a class.

Values, that have no contingent state, have a classification criterion but no identification criterion other than their value. Each value is its identity. Also, it does not make sense to say that a value is created, exists, is destroyed or does not exist. The concept of existence is not applicable to values. Figure 8 lists the entries in the subject domain part of the dictionary. We italicize terms defined elsewhere in the dictionary.

6 Desired System Functions

We give specifications of the desired system functions that correspond to the behavioral requirements of section 4. In order to do this, we first introduce some extra terminology in the dictionary.

6.1 System function dictionary

The system function dictionary given in figure 9 extends the subject domain dictionary with a number of key terms that occur in the function specifications. The terms still define concepts of the subject domain, i.e. the definiens can always be expanded so that all its key terms are from the subject domain dictionary.

6.2 System function specifications

Note that the following specification presupposes agreement on the requirements of section 4. On the other hand, making the requirements explicit, as is done in the following list, may help to reach agreement on the requirements. There is thus a hermeneutical circle between requirements and system functions. This circle is broken by starting with the requirements and returning to them as frequently as needed until the time is up.

A system function is a useful piece of external system interactions. What is considered useful is a subjective judgement left to the perception of the beholder. The important thing is that an external function involves an interaction with one or more external actors. When we specify a function, we therefore treat the system as an actor, interacting with other actors in its external environment. This distinguishes the external function specifications from the

---

3 Jacobson et al. (1992) call external functions use cases.
- **Date**: A date and time at which a *meeting* can be held. Since dates are values without contingent state, they have no separate identity criterion.

- **Date range**: Each *meeting* has a date range determined by a **first** and **last** possible date of the *meeting*.

- **Excluded date**: Each *participant* may have a list of **dates** on which the participant cannot attend the *meeting*. Each *participant-date* pair identifies at most one excluded date for a particular *meeting*.

- **Exclusion set**: The set of excluded dates of a *participant*.

- **Initiator**: A role of a *person* who initiates a *meeting*. The initiator takes the initiative in the process of meeting scheduling. Each *meeting* has exactly one initiator. Initiators are identified by means of a negotiation process that is outside the scope of the current subject domain model. For example, a steering committee may appoint an initiator.

- **Meeting**: A gathering of *participants* at a **location**, where they use the requested **equipment**. Each meeting has a purpose (outside the scope of the subject domain model). A *meeting* is identified by the **initiator**. A *meeting exists* when the initiator, after agreement with the stakeholders, declares it to exist. A meeting is thus considered to exist even before a date for it is scheduled.

- **Meeting date**: A date at which a *meeting* can be held. A meeting date is identified by a *meeting* and a **date**. There is at most one meeting date for each meeting.

- **Person**: A biological person who can play the role of *participant*. The identity of a person is his/her biological identity, determined at birth.

- **Participant**: a role of a *person* who indicated that he or she wants to participate in a meeting. Each person can play any number (including 0) participant roles. Actual participation is not relevant for the MSS. The participant role just represents the intention to participate and a person may never turn up at a *meeting* in which he/she participates. The participant role is identified by the **initiator** by criteria unknown to the MSS. For example, the **initiator** determines a list of participants on her own or by receiving a list from elsewhere.

- **Preferred date**: Each *participant* may have a list of **dates** on which the participant can attend the *meeting*. Each *participant-date* pair identifies at most one preferred **date** for a particular *meeting*.

- **Preference set**: The set of preferred **dates** of a *participant*.

- **Preferred location**: Each *participant* may have a list of **locations** which the participant would like to conduct the *meeting*. Each *participant-location* pair identifies at most one preferred location for a particular *meeting*.

**Figure 8**: TRADE specification: Subject domain dictionary.
• **Conflict**: There exists no *meeting date*, given the current *subject domain state*, that satisfies the *scheduling objective*. There are many kinds of conflict, among which are *Weak* and *Strong conflict*, defined elsewhere in this dictionary.

• **Conflict resolution strategy**: An action to eliminate a *conflict*. Conflicts can be eliminated by changing the state of the subject domain or changing the *scheduling objective*. Change of *subject domain state*:
  - A *participant* withdraws from the *meeting*.
  - A *participant* changes his or her *exclusion* or *preference* set.
  - The *date range* is changed.

Change of *scheduling objective*:
  - Ignore the *exclusions* or *preferences* of certain *participants* (to be indicated by the *initiator*).

• **Performance attribute**: All attributes of a *meeting date* are called performance attributes. They represent an aspect of how well the *meeting date* is doing with respect to the *scheduling objective*. The performance attributes of *meeting dates* are:
  - Set of *exclusion sets* violated. This is the violation relationship between *Meeting date* and *Exclusion date*. There is a *strong conflict* if this set is non-empty.
  - Set of *preference sets* violated. This is the violation relationship between *Meeting date* and *Preference date*. There is a *weak conflict* when this set is non-empty but the set of violated *exclusion dates* is empty.

• **Provisional meeting date**: A *solution* chosen by the *initiator* for further consideration. The *initiator* can turn this into a definitive *meeting date*.

• **Scheduling objective**: The set of criteria that the *meeting date* must satisfy. For example, the *meeting date* must lie inside the *date range*, outside all *exclusion set* and the number of *preference sets* to which it belongs must be maximal. For every *scheduling objective* and *subject domain state*, there may be 0, 1 or more possible *meeting dates* that satisfy the *scheduling objective*.

• **Solution**: An element of a *solution set*.

• **Solution set**: The set of *meeting dates* that meets a *scheduling objective* and a given *subject domain state*.

• **Strong conflict**: There is no *date* inside the *date range* of the *meeting* and outside the *exclusion sets* of participants of the *meeting*.

• **Subject domain state**: The set of objects and links that exist in the *subject domain*, and their attribute values.

• **Weak conflict**: There is a *date* inside the *date range* of a *meeting*, and outside all *exclusion sets* for the *meeting*, but there is no *date* inside the *date range*, outside all *exclusion sets*, and inside all *preference sets* of the *participants* of the *meeting*.

Figure 9: TRADE specification: Function dictionary.
requirements specification, where we describe the things that the external actors would like to do without referring at all to the system.

Each function is specified according to the following template:

- **Event**: The event that triggers the function.

- **Meaning**: When initiated by an external actor, we view the event as a speech act with the actor as speaker and the system as hearer. This agrees with our view of the system as an actor among other actors in its environment and also capitalizes on the fact that all interactions between a software system and its environment are symbolic. Therefore, a description of the meaning of the external function should include the following.
  
  - The propositional content of the event as interpreted in the subject domain.
  - The purpose of the event viewed as speech act. The purpose can be to *describe, prescribe, declare* or *express* something to the hearer, or to *commit* the speaker to do something.
  - The description of the meaning may also include a specification of the preparatory conditions of the speech act. For example, a minimal preparatory condition of any prescription to do action \( \alpha \) is that the hearer is able to carry out \( \alpha \) and that the hearer will not perform \( \alpha \) of its own accord.

- **Desired response**: Desired system response to the event. This must be stated in terms of future externally observable behavior of the system. This response may consist itself of speech acts with the system as speaker and one or more actors as hearers, or it may consist of changing the disposition of the system to behave in certain ways in the future.

- **Guard**: If this condition is false, the event/response pair cannot take place. This may have to be made explicit.

- **Pre/postcondition pairs**: If there is significant data exchange with the external environment, the input and output data can be related by means of pre/postcondition pairs. Since the data refer to the subject domain, the conditions have meaning in the subject domain. There may be any number of pre/postcondition pairs for a function. If there is more than one, they must be mutually consistent. We do not give pre/postcondition pairs here because they are trivial.

- **Context diagram**: This can show which external actors are involved in the interaction and what data flows between the actors and the system. We do not show context diagrams here because they are trivial in the example.

The system administrator functions below do not correspond with any requirement but are a consequence of the fact that the MSS exists and must perform certain functions for its environment.

---

4 This is also called *illocutionary point* by Searle (1969).
**System administration.**

**Event 1.1:** System administrator informs the MSS of the existence of a Person or Initiator subject domain objects.\(^5\)
- **Meaning:** (Obvious.)
- **Desired response:** The MSS remembers the existence of the entity for later use.
- **Guard:**

**Event 1.2:** System administrator disinforms the MSS of the existence of a subject domain entity. (This is a deletion of the representation of the subject domain entity.)
- **Meaning:** (Obvious.)
- **Desired response:** The MSS does not remember the existence of the entity.
- **Guard:**

**Event 1.3:** System administrator informs the MSS of the nonexistence of a subject domain entity. (This may be called “deletion of the representation of a subject domain entity”.)
- **Meaning:** (Obvious.)
- **Desired response:** The MSS remembers the nonexistence of the entity.
- **Guard:**

**Event 1.4:** System administrator informs the MSS of the state of a subject domain entity. (This is usually called an *update.*)
- **Meaning:** (Obvious.)
- **Desired response:** The MSS remembers the state of the entity.
- **Guard:**

**Determine a meeting date.**

**Event 2.1:** Initiator selects a date range.
- **Meaning:** The initiator makes a commitment to schedule the meeting in this date range by informing the MSS of this date range.\(^6\)
- **Desired response:** When requested to suggest possible dates, the MSS shall only suggest dates within the date range. Information about the date range is public for all participants.
- **Guard:** A meeting must exist before a date range can be given. (See definition of meeting existence.)

**Event 2.2:** Initiator changes a date range.
- **Meaning:** (Not specified.)
- **Desired response:**
- **Guard:** A meeting with a selected date range must exist.
Table 1: TRADE specification: Desired external functions.

<table>
<thead>
<tr>
<th>Event 2.3: Initiator collects a list of participants.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> The initiator informs the MSS of the participants. The initiator collects this list from some source. The right to participate is often granted to people at an earlier time by some other person or institution than the initiator. Or possibly, everyone interested can participate. Whatever the case, the initiator must have a list of names and addresses with whom to communicate about the meeting scheduling process. As far as the MSS is concerned, it is the initiator who says that certain persons are participants.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> The MSS will facilitate communication between the initiator and participants. It will send messages from the participants to the initiator and vice versa, and it will make certain information public for all participants.</td>
</tr>
<tr>
<td><strong>Guard:</strong> A meeting must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.4: Initiator changes list of potential participants.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> (Not specified.)</td>
</tr>
<tr>
<td><strong>Desired response:</strong></td>
</tr>
<tr>
<td><strong>Guard:</strong> A meeting with a list of participants must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.5: Initiator requests the MSS to inform each participant about the existence of a meeting and the current date range.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> The information is sent to the participants through the MSS, which delivers it to the participants. These may or may not read the delivered information.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> The MSS delivers the information to the participants.</td>
</tr>
<tr>
<td><strong>Guard:</strong> The meeting must exist and a date range and list of participants must have been selected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.6: Initiator requests the MSS to ask each participant to give its excluded and preferred dates before a certain deadline.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong></td>
</tr>
<tr>
<td><strong>Desired response:</strong> The MSS passes the question on to the participants, together with the deadline.</td>
</tr>
<tr>
<td><strong>Guard:</strong> The meeting, initiator, participants and date range must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.7: Participant informs the MSS of its excluded and preferred dates.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> A participant informs the MSS of his or her excluded and preferred dates.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> If the dates fall within the date range and the information is given before the deadline, the MSS will use these dates in searching for an optimal meeting date. Otherwise, the MSS will ignore the dates and will inform the participants of this.</td>
</tr>
<tr>
<td><strong>Guard:</strong> The meeting, initiator, participants and date range must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.8: Participant changes his/her list of excluded and preferred dates.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> (Obvious.)</td>
</tr>
<tr>
<td><strong>Desired response:</strong></td>
</tr>
<tr>
<td><strong>Guard:</strong> The meeting, initiator, participants and date range must exist.</td>
</tr>
</tbody>
</table>
Table 1: TRADE specification: Desired external functions.

<table>
<thead>
<tr>
<th>Event 2.9: Initiator selects scheduling objective.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Meaning</strong>: The initiator informs the MSS of the criteria that a meeting date must satisfy. The current version of the system only knows one possible criterion: The meeting date must lie inside the date range and outside the exclusion sets of the participants of this meeting, and lie inside as large a number of preference sets for this meeting as possible. More sophisticated objectives are discussed in section 8.</td>
</tr>
<tr>
<td>- <strong>Desired response</strong>: When requested to suggest meeting dates, the MSS will search for meeting dates according to the selected objective.</td>
</tr>
<tr>
<td>- <strong>Guard</strong>: The meeting, initiator, date range and objective must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.10: Initiator requests meeting dates from the system that satisfy the selected objective given the subject domain state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Meaning</strong>: (Obvious.)</td>
</tr>
<tr>
<td>- <strong>Desired response</strong>: The MSS suggests all possible meeting dates that satisfy the scheduling objective, given the subject domain state.</td>
</tr>
<tr>
<td>- If there is a strong conflict, the solution set is empty.</td>
</tr>
<tr>
<td>- If there is a weak conflict, the MSS indicates that there is a weak conflict.</td>
</tr>
<tr>
<td>- If there is no conflict, the MSS indicates this.</td>
</tr>
<tr>
<td>The MSS remembers the solution set until further notice from the initiator, so that it can be used for analysis.³</td>
</tr>
<tr>
<td>- <strong>Guard</strong>: The meeting, initiator, date range and objective must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.11: Initiator requests the list of participants whose preference sets are violated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Meaning</strong>: If there is a weak conflict, there is a non-zero natural number n such that each solution violates n preference sets. If there are different solutions, then each solution may violate different preference sets, but all solutions violate the same number of preference sets.</td>
</tr>
<tr>
<td>- <strong>Desired response</strong>: For each solution, the MSS informs the initiator whose preference sets are violated.</td>
</tr>
<tr>
<td>- <strong>Guard</strong>: The meeting, initiator, date range, scheduling objective, subject domain state, and solution set with a weak conflict must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 2.12: Initiator selects meeting date.</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Meaning</strong>: Given a solution set, the initiator informs the MSS which of these he or she chooses.</td>
</tr>
<tr>
<td>- <strong>Desired response</strong>: The MSS will remember the meeting date to answer informative questions by the initiator or participants. The MSS also immediately informs the participants of the selected date.</td>
</tr>
<tr>
<td>- <strong>Guard</strong>: The meeting, initiator, date range, scheduling objective, subject domain state and solution set must exist.</td>
</tr>
</tbody>
</table>

*Provide information about the state of the meeting scheduling process.*
**Table 1: TRADE specification: Desired external functions.**

**Event 3.1:** Participant requests information about the subject domain state.

- **Meaning:** The participant can query the following parts of the subject domain state:
  - The set of participants.
  - The date range.
  - The selected date.

- **Desired response:** The MSS answers the query if the requested datum is present.

- **Guard:** The participant must exist.

**Event 3.2:** Participant requests information about the state of the selection process.

- **Meaning:** Once a meeting is created, the MSS distinguishes the following states of the selection process.
  - Date range selected.
  - Participants collected.
  - Requesting preferred and excluded dates.
  - Preferred and excluded dates collected.
  - Searching for solution.
  - Solution selected.

There are some obvious temporal dependencies among these states, that may be represented by for example a UML activity diagram (figure 10).

- **Desired response:** The MSS provides the requested information.

- **Guard:** The participant must exist.

**Facilitate interactions.**

**Event 4.1:** Initiator sends a message to a set of participants, to which no response is requested.

- **Meaning:** The initiator requests the MSS to send an arbitrary text message to a set of one or more participants, called the addressees.

- **Desired response:** The MSS sends the message to the addressees.

- **Guard:** A meeting must exist with this initiator and the participants of this meeting must exist.

**Event 4.2:** Initiator sends a message with response obligation to a set of participants.

- **Meaning:** The initiator requests the MSS to send a message to a set of participants, together with a response deadline. The request for preferred and excluded dates is one example of such a message.

- **Desired response:** The MSS sends the message to the addressees and informs each addressee of the response deadline.

- **Guard:** A meeting must exist with this initiator and these participants.
Table 1: TRADe specification: Desired external functions.

<table>
<thead>
<tr>
<th>Event 4.3: Participant sends a message to the initiator, to which no response is obligated.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> The participant requests the MSS to send an arbitrary text message to the initiator. The message about preferred and excluded dates is one example of such a message.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> The MSS sends the message to the initiator. If there are outstanding obligations of this participant to respond to earlier messages by the initiator, the MSS asks the initiator whether this is such a response. If the initiator confirms this, the MSS marks that response obligation as fulfilled.</td>
</tr>
<tr>
<td><strong>Guard:</strong> A meeting must exist with this initiator and participant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 4.4: Participant sends a message to the initiator, to which response is obligated.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> The participant requests the MSS to send an arbitrary text message to the initiator, together with a response deadline.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> The MSS sends the message to the initiator and informs the initiator of the response deadline. If there are outstanding obligations of this participant to respond to earlier messages by the initiator, the MSS asks the initiator whether this is such a response. If the initiator confirms this, the MSS marks that response obligation as fulfilled.</td>
</tr>
<tr>
<td><strong>Guard:</strong> A meeting must exist with this initiator and participant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 4.5: The MSS reminds initiator to respond to participant message(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> This is a temporal event and not a speech act by an actor. The event consists of the passing of a response deadline.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> When the deadline for response to a participant message passes, the MSS informs the initiator of this in a way that attracts the attention of the initiator, until the initiator has responded.</td>
</tr>
<tr>
<td><strong>Guard:</strong> The initiator, participant and message with response obligation must exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 4.6: The MSS reminds a participant to respond to initiator message(s).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning:</strong> This is a temporal event and not a speech act by an actor. The event consists of the passing of a response deadline.</td>
</tr>
<tr>
<td><strong>Desired response:</strong> When the deadline for response to an initiator message passes, the MSS informs the participant of this in a way that attracts the attention of the participant, until the participant has responded.</td>
</tr>
<tr>
<td><strong>Guard:</strong> The initiator, participant and message with response obligation must exist.</td>
</tr>
</tbody>
</table>

This list must be discussed with users in order to reach agreement about the relevant events, their meaning, desired responses and guards. Also, the overhead incurred by this functionality must be compared to the current overhead in scheduling a meeting date. For example, to let the MSS manage the response obligations, the participants and initiator are harassed by reminders of the MSS to do something, and in order to enjoy this privilege, they have to perform additional actions such as thinking about response deadlines, entering them, and marking messages as responses to earlier messages! The above list of functions is to be viewed as input of a discussion with users, not as a result of such a discussion.
Figure 10: Activity diagram of the selection process.
6.3 Other desired system properties

These are all the “ilities” that cannot specified in event-response form. They follow from the nonbehavioral requirements given earlier. Each of these can be translated into a desired system property. Because this gives little information above the information already contained in the requirements specifications, we omit this here.

Following Tom Gilb, the desired system properties should either be dropped or be translated in measurable form. We take a less harsh standpoint and view these statements as good intentions, that can guide development. However, if no measurement procedure is formulated, they cannot affect acceptance tests.

7 System Decomposition

7.1 Conceptual decomposition

The conceptual decomposition of a software system is a decomposition in terms of conceptual objects whose meaning and behavior is defined entirely in terms of the external user environment. In terms of the layered context diagram of figure 3, the user environment is the collection of actors at higher layers, that use the services of the software system. The conceptual decomposition of the software system may contain software objects such as the following:

- Software objects that correspond to subject domain objects.
- Software objects that correspond to external functions.
- Software objects that correspond to user interface objects.

This list is not exhaustive but gives three frequently used decomposition criteria for the conceptual decomposition. It also shows that TRADE uses the principles of subject-domain-oriented decomposition, known from JSD (Jackson 1983) and object-oriented methods, as well as of functional decomposition, known from structured analysis (DeMarco 1978).

The conceptual decomposition contains no traces of any implementation platform. It assumes an unbounded number of processors, one for each conceptual object, each with infinite processing speed and unlimited memory and with the ability to broadcast information instantaneously and without communication overhead. This corresponds roughly with McMenamin and Palmer’s essential model (McMenamin & Palmer 1984) and with Syntropy’s specification model (Cook & Daniels 1994).

When we add components that deal with peer entities or lower-level entities, we get the physical decomposition discussed in section 7.2.

7.1.1 Conceptual class model

Figure 11 shows a class diagram of the conceptual decomposition of the MSS. In order to avoid any confusion, we follow Syntropy in suffixing every class name in this model with
Figure 11: TRADE specification: Conceptual decomposition.

This makes abundantly clear that we are here talking about software objects and not about the subject domain objects that they represent.\(^{10}\) The conceptual decomposition takes into account that Date is really a value type whose instances have no state but are just values. Consequently, the relationships to Date are now represented by attributes of Participant.\(^{S}\). All other many-many and many-one relationships are still represented as relationships. No decision has been made at this level of abstraction as to representing them as single- or multi-values attributes in either or both of the related classes.

7.1.2 Function decomposition specifications

To check whether the conceptual decomposition suffices as conceptual implementation of the external functions, we add a conceptual implementation specification to every external function specification. This gives each external function specification the appearance of a minispec as in structured analysis (DeMarco 1978, Yourdon 1989, Yourdon Inc. 1993).

A conceptual function decomposition specification is a specification of the activities performed by one or more conceptual objects as part of an external function. With a good conceptual system decomposition, the conceptual function decompositions are trivial.

Table 2 gives the conceptual decomposition of most functions. If a software object represents a subject domain object and corresponds one-one with this subject domain object, we call the software object a surrogate for the subject domain object.

Table 2: TRADE specification: Conceptual decomposition specifications.

\(^{10}\) These are often called real-world objects. I find this a confusing practice, for software objects are just as real as the subject-domain objects that they represent. In many cases, they are even more real. A software object that represents a bank account is more tangible than the represented bank account.
**Table 2: TRADE specification: Conceptual decomposition specifications.**

<table>
<thead>
<tr>
<th>Event 1.1: System administrator informs the MSS of the existence of Person or Initiator subject domain objects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> An instance of the corresponding software object class is created. In the case of an Initiator(_S) software object, a player link to a Person(_S) software object must be created as well.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 1.2: System administrator disinforms the MSS of the existence of a subject domain entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> The surrogate is deleted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 1.3: System administrator informs the MSS of the nonexistence of a subject domain entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> The surrogate is updated to represent the nonexistence.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 1.4: System administrator informs the MSS of the state of a subject domain entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> The corresponding attributes of the surrogate are updated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Determine a meeting date.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event 2.1:</strong> Initiator selects a date range.</td>
</tr>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Attribute update of a Meeting(_S) object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.2:</strong> Initiator changes a date range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Attribute update of a Meeting(_S) object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.3:</strong> Initiator collects a list of participants.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Creation of Participant(_S) objects, and of links with the Meeting(_S) linked to this initiator's surrogate.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.4:</strong> Initiator changes list of potential participants.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Creation and deletion of Participant(_S) objects, and of links with the Meeting(_S) linked to this initiator's surrogate.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.5:</strong> Initiator requests the MSS to inform each participant about the existence of a meeting, the current date range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> A message with this information is created for each participant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.6:</strong> Initiator requests the MSS to ask each participants to give its excluded and preferred dates before a certain deadline.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> A message is created with this request.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.7:</strong> Participant informs the MSS of its excluded and preferred dates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Update of the corresponding participant attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.8:</strong> Participant changes his/her list of excluded and preferred dates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Update of the corresponding participant attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Event 2.9:</strong> Initiator selects scheduling objective.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Conceptual decomposition:</strong> Update the appropriate attributes of the scheduling objective surrogate for this meeting.</td>
</tr>
</tbody>
</table>
Table 2: TRADE specification: Conceptual decomposition specifications.

<table>
<thead>
<tr>
<th>Event 2.10:</th>
<th>Initiator requests meeting dates from the system that satisfy the selected objective given the subject domain state.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conceptual decomposition:</strong></td>
<td></td>
</tr>
<tr>
<td>Event 2.11:</td>
<td>Initiator requests the list of participants whose preference sets are violated.</td>
</tr>
<tr>
<td><strong>Conceptual decomposition:</strong></td>
<td>Trivial.</td>
</tr>
<tr>
<td>Event 2.12:</td>
<td>Initiator selects meeting date.</td>
</tr>
<tr>
<td><strong>Conceptual decomposition:</strong></td>
<td></td>
</tr>
</tbody>
</table>

Provide information about the state of the meeting scheduling process.

| Event 3.1: | Participant requests information about the subject domain state. |
| **Conceptual decomposition:** | |
| Event 3.2: | Participant requests information about the state of the selection process. |
| **Conceptual decomposition:** | |

Facilitate interactions.

| Event 4.1: | Initiator sends a message to a set of participants, to which no response is requested. |
| **Conceptual decomposition:** | |
| Event 4.2: | Initiator sends a message with response obligation to a set of participants. |
| **Conceptual decomposition:** | |
| Event 4.3: | Participant sends a message to the initiator, to which no response is obligated. |
| **Conceptual decomposition:** | |
| Event 4.4: | Participant sends a message to the initiator, to which response is obligated. |
| **Conceptual decomposition:** | |
| Event 4.5: | The MSS reminds initiator to respond to participant message(s). |
| **Conceptual decomposition:** | |
| Event 4.6: | The MSS reminds a participant to respond to initiator message(s). |
| **Conceptual decomposition:** | |

7.1.3 Traceability tables

The function decomposition specifications can be summarized in traceability tables: These relate conceptual components with external functions. Each traceability table actually shows a finite set of links between external functions and conceptual components, which are called traceability links. Figure 12 shows a traceability table that links the functions in the group “Determine a meeting date” to the conceptual system components. We use the standard convention to abbreviate Create, Read, Update and Delete with their first letter.

There are two kinds of traceability tables: A function allocation table marks an entry in the table if the component corresponding to that entry does something to realize the external function corresponding to that entry. This table can be done to analyze the
impact of changing a component or external function. A function decomposition table marks the same entries of the table as a function allocation table, but indicates in the entry which action the component must perform as part of this external function. For example, it can be indicated whether the component is created, updated or deleted, or whether its state is read. Figure 12 is a function decomposition table of this kind. If we want to be more precise, we can define the action with which this component participates in this function more precisely. This shows us the flowdown of external functions to component functions. The entries of the table corresponding to one component (columns in figure 12) give us the interface of the components to its environment. The entries corresponding to one external function (rows in figure 12) show us the mechanism by which a function is conceptually implemented. In the UML, this would be illustrated by a collaboration or sequence diagram.

If there is a finite number of components that is known in advance, the table can represent all individual components of the system. In conceptual software decomposition, there are usually a potentially unbounded and unknown number of software objects, and a traceability table shows software object classes rather than individual objects. Figure 12 is of this type. An entry in the table then merely shows that one or more instances of this class participates in the external function.

The traceability table of figure 12 is not complete because we omitted the relationships from the top row, so that we cannot represent the creation and deletion of links. Nevertheless, the table shows the feasibility of the design, helps in making modularization decisions, and facilitates impact analysis. We can also analyse the use that is made of the different

<table>
<thead>
<tr>
<th>Person</th>
<th>Participant</th>
<th>Initiator</th>
<th>Meeting</th>
<th>Message</th>
<th>Scheduling</th>
<th>Meeting date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 select date range</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2 change date range</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 collect participants</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4 change participants</td>
<td>CD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5 inform participants</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6 ask dates</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.7 give dates</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.8 change dates</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9 select scheduling objective</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.10 request meeting dates</td>
<td>R</td>
<td>R</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.11 Request preference violations</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.12 select meeting date</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12:** TRADE specification: Conceptual decomposition of the functions in the group “Determine a meeting date and location”.


component object classes.

- Is each object created? We see that instance sof Person_S, Initiator_S, Meeting_S and Scheduling_objective_S are not created by these functions. There must therefore be other functions that create them.

- Is each object ever deleted? For example, we see that messages are only created by the functions of the table and never deleted.

- Are objects used, i.e. read or updated? This allows us to find out what the purpose of each object is.

7.1.4 Conceptual communication model

This is a summary of the communications that take place during execution of the external functions. If a function decomposition table shows the decomposition of atomic external functions, i.e. of external transactions, then each column in the table represents a number of communications. Figure 13 shows a communication diagram of the MSS conceptual components and external actors. Each rectangle represents an object class and each line a potential communication between instances of the linked classes. In the conceptual decomposition, all communications are synchronous.

Each communication link can be labelled by the actions by means of which the components communicate over this link and by the number of the external function during which this link is activated. These extra labels do not improve the readability of the diagram and they are omitted here.

There seem to be several styles of conceptual decomposition with respect to communication structures:

- **Database systems.** For example, the subject domain database consisting of person, initiator, participant and meeting surrogates. These do not communicate with each other but with external actors, who create, read, update or delete surrogates in this database.

- **Communication systems.** For example, the message software objects implement a communication system. These communicate with each other and with external actors to pass on messages.
• Control systems. This is not present in the case study but examples abound where we have a control loop connecting external hardware to controlling software objects. The software objects communicate with external hardware actors to measure their state and influence their behavior. They communicate with each other to pass close the loop from measurement to control.

These styles can occur mixed in one system, as the case study illustrates.

Another point to observe is the identification of subsystems. The following subsystems occur in the case study:

• Subject domain subsystem
• User interface subsystem
• Decision support subsystem
• Communication subsystem

7.2 Physical Decomposition

A number of methods give heuristics for mapping a conceptual design to a physical one. Cook & Daniels (1994) give heuristics for transforming a design that runs on an infinitely fast reliable machine to one that can run on a machine that contains finitely many sequential objects that run concurrently, must deal with errors, contains only message passing as communication primitive. Shumate & Keller (1992) give guidelines for transforming a functional decomposition into a design for a distributed system and Gomaa (1993) does the same in a mixed structured and OO notation. Awad et al. (1996) gives some guidelines for object-oriented models based upon Fusion and OMT. These different approaches are hard to compare, let alone evaluate with respect to each other. Transformation of a conceptual into a distributed physical design is subject of current research.

At least two layers seem relevant for a physical decomposition:

• The distribution layer, at which physical resources are located at nodes in a communication network. Often, this layer is already there and cannot be changed by the designer. However, if there is design freedom here, considerations such as the following lead to a definition of a network topology (Gomaa 1993):
  – Proximity to source of data
  – Local autonomy
  – Performance requirements
  – Need for specialized hardware
  – Need for specialized user interfaces
  – Need for database servers

• The concurrency layer, at which sequential processes run concurrently on one or more physically distributed resources. The heuristics given by different methods for identifying concurrent tasks in a group of conceptual objects are mutually inconsistent. We defer discussion of this to a future report.
8 Features

The basic MSS can be extended with many features that make it more useful — but that also make the user interface more complex. The features listed in this section are suggested by Feather et al. (1997). The interesting thing about the following specifications is that each feature specification may cut across requirements specification, functional specification, subject domain model and conceptual decomposition. This illustrates a tight coupling of features with the rest of the system.

8.1 Locations

Requirements. There are several features that we can add related to locations: administration of possible locations, administration of dates that possible locations are still available, etc. We consider the following list:

- Initiator collects possible locations.
- Initiator changes possible locations.
- Initiator sets available dates for a location.
- Participant suggests preferred locations.
- Initiator selects meeting location.

Subject domain. The subject domain model must be extended as shown in figure 14. The subject domain dictionary must be extended with the following entry:

- Location: An organization that owns facilities at which to conduct a meeting. Locations are identified by the chamber of commerce as companies.\footnote{Alternatively, geographical locations may be used. It is important however to identify a correspondence address.}

Desired functions.

- Event: Participant informs the MSS of locations preferred by him/her.
  - Meaning: (Obvious.)
  - Desired response: The MSS remembers the preferred locations.
  - Guard: The meeting and participant must exist.
  - Conceptual decomposition: Possible creation of Location$S$ object and creation of a link between this participant’s surrogate to this Location$S$ object.

- Event: Initiator selects possible location(s).
  - Meaning: The initiator makes a commitment to organize the meeting at this location/these locations by informing the MSS of these locations.
  - Desired response: When searching for possible meeting locations, the MSS shall first search for possible locations in this set. Information about the possible locations is public for all participants.
- **Guard**: A meeting must exist before a possible location can be given.
- **Conceptual decomposition**: Creation of a link between a `Meeting` object and a `Location` object.

- **Event**: Initiator changes possible location(s).
  - **Meaning**: (Not specified.)
  - **Desired response**:
    - **Guard**: A meeting must exist.
    - **Conceptual decomposition**: Creation and deletion of a link between a `Meeting` object and a `Location` object.

- **Event**: Initiator asks MSS to rank locations according to participant preference.
  - **Meaning**:
  - **Desired response**:
    - **Guard**:
    - **Conceptual decomposition**:

- **Event**: Initiator requests MSS to rank locations according to availability.
  - **Meaning**:
  - **Desired response**:
– Guard:
– Conceptual decomposition:

• **Event 2.5:** Initiator requests the MSS to inform each participant about the existence of a meeting and the current date range and possible locations.

• **Event 2.9:** Initiator selects scheduling objective. This includes the objective that the availability of preferred locations is matched against the preferred dates.

• **Event 3.1:** Participant requests information about the subject domain state. This now includes a query of the possible locations or the selected location.

### 8.2 Equipment

**Requirement.**

- Participant requests special equipment.

**Subject domain.** The subject domain model must be extended with an equipment object type, linked by a request association to the Participant object type. The dictionary must be extended with the following entries:

- **Equipment:** Tools desired by a participant for effective participation, such as an overhead sheet projector or a slide projector. In addition to equipment needed for meeting presentations, equipment may also be requested for the disabled, such as hearing aids etc. They are identified by the organization owning, selling or renting the equipment. For example, a microphone may or may not be treated as a different piece of equipment than the amplifier connected to it.

- **Equipment need:** Each participant may have a need for certain kinds of equipment. Each participant-equipment pair identifies at most one need.

**Desired system functions.**

- The system administrator can now also inform the MSS of the existence of a certain type of equipment.

• **Event:** Participant requests special equipment.
  – **Meaning:** (Obvious.)
  – **Desired response:** The MSS remembers the equipment request.
  – **Guard:** The meeting and participants must exist.
  – **Conceptual decomposition:** Update of appropriate attribute of the participant surrogate.

• **Event:** Initiator prints a list of desired special equipment.
  – **Meaning:** (Obvious.)
  – **Desired response:** A list of desired special equipment is printed.
  – **Guard:** The meeting and participants must exist.
  – **Conceptual decomposition:** Trivial.
8.3 Hypothetical subject domain changes

**Requirement.** The initiator is able to see if the scheduling objective can be satisfied in a hypothetical subject domain state.

**Function specifications.**

- **Event:** The initiator requests the MSS to change the subject domain state hypothetically.
  
  - **Meaning:** The following hypothetical changes are possible:
    
    * A participant withdraws from the meeting.
    * A participants drops a date from his/her exclusion set.
    * A participant drops a date from his/her preference set.

  The hypothetical state is invisible to any participant.

  - **Desired response:** The MSS remembers the hypothetical state of the subject domain for later analysis by the initiator.

  - **Guard:**

    - **Conceptual decomposition:** Subject domain surrogates are now sets of objects, identified by the surrogate and a version identifier. Exactly one version represents the actual subject domain state.

- **Event:** The initiator requests the MSS to roll back to a hypothetical subject domain state.

  - **Meaning:** Given a sequence of hypothetical domain states, the initiator selects one.

  - **Desired response:**

  - **Guard:**

    - **Conceptual decomposition:** Surrogate versions are delted.

- **Event:** The initiator requests the MSS to suggest possible meeting dates with respect to a hypothetical domain state.

  - **Meaning:** (Obvious.)

  - **Desired response:** The same as for the event *Initiator requests meeting dates from the system that satisfy the selected objective given the subject domain state.*

  - **Guard:**

    - **Conceptual decomposition:** As before, but with respect to a subject domain version.

- **Event:** Initiator requests the list of participants whose preference sets are violated.

  - **Meaning:** This function is now extended to include hypothetical solutions, i.e. solutions with respect to a hypothetical subject domain state.

  - **Desired response:**
- Guard:
- **Conceptual decomposition:** As before.

- **Event:** The initiator requests the MSS to present a list of exclusion sets.
  - **Meaning:** (Obvious.) The reason for wanting this function is that it may suggest hypothetical updates.
  - **Desired response:** The MSS presents the exclusion sets in a way that allows the initiator to infer something about exclusion sets whose elimination would eliminate strong conflict.
  - Guard:
  - **Conceptual decomposition:** Trivial.

- **Event:** Initiator requests the list of participants whose exclusion sets are violated in the actual subject domain state.
  - **Meaning:** Given a solution of a scheduling objective with respect to a hypothetical subject domain state, there may be participants whose excluded dates are violated in the actual domain state. The MSS is asked to give a list of these participants.
  - **Desired response:** The MSS gives the requested list plus the actual exclusion set.
  - **Guard:** The initiator has selected a meeting date with respect to a hypothetical subject domain state.
  - **Conceptual decomposition:** Search through participant instances.

### 8.4 Changes to the scheduling objective

#### 8.4.1 Person status

**Requirement.** The initiator is able to select a scheduling objective that, in case of conflict, ignores the constraints set by persons of lower status.

**Subject domain.** Participant status is a representation of the location of a person in an institutional hierarchy, independent from any meeting. The set of possible statuses is defined, and a status is assigned to persons, by institutional processes outside the scope of the MSS.

To incorporate this in the model, we extend the subject domain model with an abstract data type **Status** with enumeration values such as low, medium, high and extend the subject domain dictionary with the following entries:

- **High status:** (To be defined according to the rules of the organization).
- **Low status:** (To be defined according to the rules of the organization).
- **Medium status:** (To be defined according to the rules of the organization).
- **Participant status:** The status of the person who plays the role of this participant.
- **Status**: attribute of a person that indicates a position in an organizational hierarchy.

Modeling status as attribute of person is not quite right. More accurate is a distinction between persons and organizational roles and the assignment of status as attribute to a role (figure 15). This immediately makes clear that status may be ambiguous: a person may play several roles in an organization, with different statuses. The problem is eliminated if we require every person to play exactly one role, but this is not very realistic. It can also be eliminated by modeling participant as role of organizational roles (figure 16). Now one person can play several organizational roles, each with its own status, and each of these organizational roles may play several participant roles. This should be sorted out with the user.

**Function specifications.** The following list gives a number of possible functions for this added feature. Specification of the meaning, desired response, guard and conceptual
decomposition of these function sis trivial.

- **Event:** The initiator requests the MSS to select the following scheduling objective:
  - Satisfy the exclusion and preference sets of all persons above a certain status.
- **Event:** List the persons below a certain status whose exclusion set blocks a solution.
- **Event:** List the participants below a certain status whose preference set blocks a solution.

### 8.4.2 Participant importance

**Requirement.** The initiator is able to select a scheduling objective in which the constraints set by participants of less importance need not be satisfied.

**Subject domain.** The subject domain model must be adapted to include a *importance* attribute of *Participant*. The subject domain dictionary must be extended to include the following entries:

- **Importance:** A representation of the desirability that a participant is able to attend a meeting. It is represented by a natural number, in which 0 represents the highest importance and higher numbers represent a lower importance.

### 8.4.3 Function specifications.

- **Event:** The initiator requests the MSS to select the following scheduling objective:
  - Satisfy the exclusion and preference sets all participants above a certain importance.
- **Event:** The initiator requests the MSS to give a list of participants below a certain importance whose exclusion set blocks a solution.
- **Event:** The initiator requests the MSS to give a list of participants below a certain importance whose preference set blocks a solution.

### 8.4.4 Meeting significance

**Requirement.** The initiator is able to select as scheduling objective that the constraints of participants for whom this meeting is of low significance, are not satisfied.

**Subject domain.** In the subject domain model, each participant gets an attribute *significance*. The subject domain dictionary is extended with the following entry.

- **Significance:** Attribute of a participant that represents the relative importance of this meeting for this person (player of participant) with respect to other meetings that this person wants to participate in. The default is that each meeting has the highest possible significance.
Function specifications.

- **Event**: A participant sets his or her significance attribute.
- **Event**: The initiator requests the MSS to select the following scheduling objective:
  - Satisfy the exclusion and preference sets all participants for whom this meeting has at least a certain significance.
- **Event**: The initiator requests the MSS to give a list of participants for whom this meeting has a significance below a certain level.

8.4.5 Participant flexibility

**Requirement.** The participants are able to weigh their excluded and preferred dates, and the scheduling objective can be to satisfy only excluded and preferred dates with higher weights.

**Subject domain.** The excluded and preferred relationships between Participant and Date have an attributes weight. The subject domain dictionary is extended with the following entry:

- **Weight**: Attribute of excluded and preferred links between Participant and Date that indicates the flexibility of the participant with respect to these links. Higher weights indicate less flexibility.

Function specifications

- Participants must now attach a weight to each date in each exclusion and preference set they send to the initiator. This datum is confidential.
- **Event**: The initiator requests the MSS to select the following scheduling objective:
  - Satisfy the exclusion and preference dates whose weight exceeds a certain threshold.

9 Discussion and conclusion

This report shows that the TRADE framework gives a simple software specification that can be applied to a groupware system. A central point in TRADE is the distinction between different specification levels: requirements, external functions, conceptual decomposition, physical decompositions. In addition, a subject domain model and a system dictionary support the understandability of the specifications. TRADE incorporates features from structured and object-oriented development methods.

To apply TRADE in practice, one only uses as much as is useful and omits the rest. This is no different from the application of any other notation in engineering design. The use of design methods is to guide the designer in his or her decisions about the system under development. Each design decision reduces uncertainty. The role of a framework like
TRADE is to guide the decisions and document the result. This role stops when there is no need for guidance or documentation.

The only part of a TRADE specification that is obligatory for all systems is the system purpose specification. Of the other parts, a rough sketch of desired functions and their subject domain are often useful. These give a good first version of a conceptual decomposition. For the rest, the amount of detail to be added is up to the designer. This report shows what happens if we single-mindedly apply all parts of TRADE that could be applied to the meeting scheduler problem.

Acknowledgements: Klaas Sikkel gave some useful comments upon an earlier version of this report.

References


