veken [10]. In this paper we have applied some of its basic ideas to organisational communication related to the use of databases. This kind of communication has certain characteristics that make this enterprise different from a more traditional use of the speech act theory. How is it different? A typical example of a speech act for Austin and Searle is an act that involves two people in a synchronous communication, where both persons are located in the same place. When people use databases, they often communicate asynchronously with people at a far distance. Moreover, a software system transforms information. It is not necessarily a passive communication media. We are also concerned with strictly institutionalised behaviour, much more than in our everyday use of natural language in general. We may hence expect that it is easier to create a taxonomy of speech acts in this context. Finally, Searle,s taxonomy is designed to deal with communication between individuals, while we are typically concerned with people that act on the behalf of social organisations.

The speech act theory has been applied in the context of information systems development by Lyytinen et al., [2-5] and Flores et al. [7,8,12], also described in [11]. The 'speech act, concept has also been used in the ORDIT approach, see e.g. [6]. The contribution of the COMMODIOUS method, in relation to Lyytinen et al., is to consider how we can use a speech act model more directly when designing software systems. The SAMPO method is, however, broader in scope and content. The COMMODIOUS method focuses only on contracting discourses and has not used the full expressiveness and formalism of the SAMPO method. The contribution in relation to Flores et al., is that we have extended the generic schema, we have allowed the user to edit the speech act sequence, and we have allowed the user to specify an information model for a discourse.

7. Summary

In this paper we have proposed a method for requirements capture, called the COMMODIOUS method. We have shown how it is possible to use a model of the organisational communication as a basis when formulating software requirements. We have also described how the user of this method can be supported by a tool. We have seen how this tool can check a discourse (communication structure) for completeness. It can suggest specific rules for how speech acts may follow upon each other. We have also shown how the tool can generate suggestions for the information model. Finally we have sketched how the tool can suggest what input is needed for some software support functions, and how it can suggest specific rules for these functions.

There are several problems that have not been addressed. We have not allowed the information model to include generalisations. We have not focused on how speech acts may trigger and be triggered by other activities, e.g. a confirmation of an order may be preceded by a complex procedure of order verification performed by different persons in the organisation. There is also a need for further empirical studies in order to extend the set of standard suggestions, and to better understand how contracting discourses are concretely implemented in business firms.

8. Literature

[1] Action Technologies (1993) "ActionWorkFlow Analyst User's Guide,,

[2] Auramäki E, Hirschheim R, and Lyytinen K (1992a) "Modelling Offices Through Discourse Analysis: The SAMPO Approach,,, In: *The Computer Journal*, 35, No 4 1992, (pp. 342 - 352)

[3] Auramäki E, Hirschheim R, and Lyytinen K (1992b) "Modelling Offices Through Discourse Analysis: A Comparison and Evaluation of SAMPO and OSSAD and ICN,,, In: *The Computer Journal*, 35, No 5 1992, (pp. 492 -500)

[4] Auramäki E, Lehtinen E, and Lyytinen K (1988) "A speech-Act-Based Office Modeling Approach,,, In: *ACM Transactions on Office Information Systems*, 6(2), (pp. 126-152)

[5] Auramäki E (1988) *A Speech Act Based Model for Analysing Cooperative Work in Office Information Systems* In: Conference Proceedings: 6:th EFISS Symposium, Atlanta, Georgia, USA, Oct 19-21 1988

[6] Blyth A J C, chudge J, Dobson j E, and strens M R (1992) *The ORDIT Approach to Requirements Identification* In: Compsac '92, (pp. 356-361), (1992)

[7] Flores F, Graves M, Hartfield B, and Winograd T (1988) "Computer Systems and the Design of Organizational Interaction,,, In: *ACM Transactions on Office Information Systems*, 6 (2)

[8] Medina-Mora R, Winograd T, Flores R, and Flores F (1993) *The Action Workflow Approach to Workflow Management Technology* In: Conference Proceedings: Third European Conference on Computer Supported Cooperative Work, (pp. 281-288), Milano, Italy,

[9] Searle J R (1979) *Expression and Meaning*, Cambridge, 1979, Cambridge U P

[10] Searle J R and Vanderveken D (1985) *Foundations of Illocutionary Logic*, Cambridge, 1985, Cambridge University Press

[11] Schäl T and Zeller B (1993) *Supporting Cooperative Processes with Workflow Management Technology* Tutorial preceding: Third European Conference on Computer Supported Cooperative Work, Milano, Italy, 1993

[12] Winograd T and Flores F (1986) *Understanding Computers and Cognition*, Massachusetts, 1986, Addison-Wesley Publishing Company Inc.

- 1,-,-,- always.[5]
- -,m,-,- if spA2 can be performed several times after spA1. [6,7]
- -,1,-,- Otherwise.

The advice mechanism can also suggest and generate inter speech act information links between core entity types of speech acts of group 1 and core entity types of speech acts of group 2. There will also be indirect cross references between the ordering part and the contracting part. A core entity type of "create commitment„ in an ordering part typically refers to the core entity type of the information about customer, and the information about the service.

5.2.2. Generation of Core Entity Types

The above algorithm assumes that a core entity type is specified for each registration event. If this is not the case, but it is specified what entity types and attribute types that are updated by a registration event, and if the mapping constraint is specified for the attribute types between these entity types, then the COMMODIOUS tool can make suggestions as regards the core entity types. Unfortunately there is not enough room to describe this algorithm in its full length. I will simply give a short example. Figure 5-4 illustrates how the tool can choose two core entity types among a set of entity types updated by two different speech acts. In this way it can derive the source and destination of a back reference link.

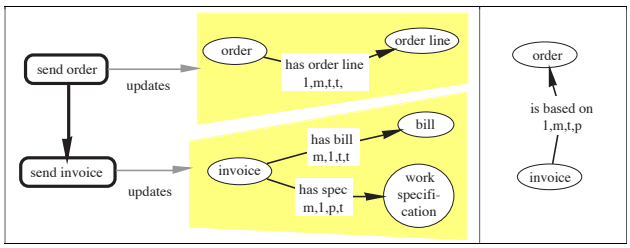


Figure 5–4: Detection of core entity types

The basic idea is to identify the entity type that is „obligatory„ for the group. (An instance of this entity type will be created at each registration event occurrence.) We must also consider uniqueness in the group. In some situations the tool will suggest a creation of a new entity type and new attribute types, with mapping constraints. It may also suggest deletion of old attribute types that have become superfluous. This is illustrated in figure 5-5.

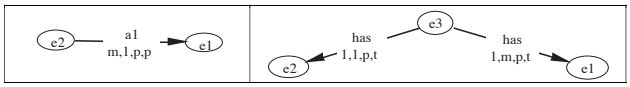


Figure 5–5: Illustration of a creation of a core entity type

5.3. The System Behaviour Model

Unfortunately there is not much room left in this paper to cover the last part of the advice mechanism. However, here is a short summary: When producing traditional process models, such as information flow diagrams, one of the major problems is to identify, and motivate different types of information needs for different activities. The COMMODIOUS tool can derive several such information needs from the sequence diagram. These derived information needs will be presented to the user as suggestion that a support function for an activity should use certain predicates in the rules (see figure 4-7). In some cases the tool can help the user with an even harder problem, i.e. to derive, understand, and motivate the rules for certain software processes. Here are some examples of suggestions made by the tool:

- Each action guidance function for speech acts in a contracting discourse needs to use the (system identifier of) the core entity type of all speech acts that must precede the speech act. It also needs a user input to be matched with this system identifier. There should also be a rule that the user is warned not to perform the speech act if the user input does not match the core entity type in the schema. Let us take an example. Consider an active action guidance function for the speech act "register loan„ in a library. The tool will suggest:
  user input: aCardNr
  Op1: action guidance: "Advice the user no to perform the speech act 'register loan„
  Predicate to use in the rules: borrowingCard[1], cardNr[2]
  Formal rule: -∃x,y(borrowingCard(x) & cardNr(x,y) & y = aCardNr)) → Op1
  Explanation of the suggested event rule: "The speech act 'register loan, must be preceded by the speech act 'give borrowing card,„. (Notice that also this precedence condition can be generated by the tool in the first place.)
- An analogous suggestion holds for each speech act that should not be precede by another speech act.
- A support function for a speech act of the type "complaint against customer„ will typically need to use the following predicates in its rules: core predicate (with the system identifier) of the information about customer,s fulfilment of obligation, the information about report of fulfilment of customer obligations, the information about the fulfilment of service, and the regulations about customer obligations.
- et cetera...

6. Related Research

What is often referred to as the "classical speech act theory„ was developed by Austin, Searle [9], and Vander-

---

[5] The intra speech act information link indirectly denotes the discourse, to which e2 belongs. This reference should always be unique.

[6] This can happen in two situations: If spA2 is included in a cycle without spA1, e.g. see figure 5-2, and if there is an and-node after spA1, that reaches spA2 in two ways.
Every time spA2 is performed and a statement e2(x) is inserted in the database instance, the statement a1(x,y), that connects the statement e2(x) with earlier statements produced in the same dicourse performance, will always refer to the same y .

[7] In rare situations it is undecidable whether the inverse of a1 is multivalued, i.e. if there are several inter speech act link „backwards„. The advice mechanism suggests an 'm?, since 'm, is never too restrictive. At the same time the user should be allerted that this constraint *may* be a '1,.

The advice mechanism is based on the following three assumptions: All entity and attribute types are updated by only one registration event. All entity and attribute types that are updated by the same registration event should be connected to each other. Each registration event (and hence each speech act) is associated with one and only one core predicate, which is supposed to be an entity type. It is called the „core entity type„.

5.2.1. Inter Speech Act Information Links

Inter speech act information links express rules for how the speech acts in a discourse, performed at different occasions, perhaps by different agents at different locations, are related to each other. Basically they represent rules concerning the dynamic progress in a discourse. Can an order be succeeded by several invoices? Must it be succeeded by one? Intra speech act information, on the other hand, expresses rules for the internal coherence and consistency of the speech acts updating these entity and attribute types. Can an order contain several order lines? Can a person have two telephone numbers?

Notice that the sequence diagram in figure 5-2 contains more rules than the information model. The information model does not, for instance, capture the fact that you can not register a complaint after you have registered a payment. It is reasonable to assume that many people would regard a sequence diagram easier and more intuitive to work with, compared to the mapping constraints in the information model. The user can thus express more rules in a simpler way.[3]

But, if we can express these rules in the sequence diagram, why should we bother to repeat them in the information model? The information model need only to contain intra speech act information. We could introduce a unique identifier for each discourse. Each core predicate could then have an attribute type with this identifier and another attribute type describing the time when the speech act was performed. Then all intra speech act information links would be superfluous. If all contracting discourses were specified like this, they would all look the same and it would be easy to compare them. However, things are not that easy. In requirements engineering it is satisfactory if we can express requirements in some way or another. It does not matter *per se*, if rules are expressed in a sequence diagram or in an information model. But, an information model may also be used for other purposes, e.g. to create an overview of a database schema, as a reference when defining search conditions. In these situations it may be important to include the sequence conditions also in the information model, in order to give the user an idea of how the different parts of the schema are related to each other, e.g. that there is only one payment per invoice. We may thus expect that people will continue to create traditional information models. Hence the COMMODIOUS tool is designed to support the generation of such information models. The advice mechanism is based on the following assumed requirements for the generation of inter speech act information links:

r_a: For each core entity type it should be possible to identify all entity types that belong to the same discourse.

r_b: The inter speech act information links should capture as many sequence conditions as possible.

r_c: If possible, there should be only one inter speech act information link „backwards„. (We assume that the user wants an information model with as few attribute types as possible.)

The left part of figure 5–3 illustrates the basic idea in the advice mechanism, i.e. to create one inter speech act information link from the core entity type of one speech act to the core entity type of the speech act that directly precedes the speech act under consideration. We will call these links for „back reference links„. The right part of the figure illustrates a deviation from this principle (since „payment„ does not have an attribute type to „complaint„). This deviation can be explained with reference to requirement r_c, above. If there are several possible back references, the advice mechanism tries to find a reference to an entity type that *must* have been produced earlier in the discourse. Among these, we choose the core entity type of the speech act that precedes the speech act under consideration most immediately.

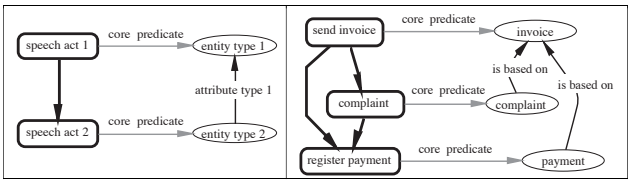


Figure 5–3: Illustration of inter speech act information links

The advice mechanism can also generate mapping constraints for back reference links. The input to this algorithm is two speech acts, spA1 and spA2, with two core entity types, e1 and e2, and an attribute type (the back reference link), a1, from e2 to e1. The different parts of the mapping constraints for a1 can be generated accordingly:

- -,-,t,- if spA1 must have been performed before spA2.
- -,-,-,p always.[4]

[3] This result can be generalised to all situations where there exists a well defined sequence of events that causes the database to be updated and where it is desirable to store information about this sequence (e.g. by means of inter speech act information links). It is not limited to contracting discourses.

[4] The speech act that updates e1 happens before the speech act that updates e2, in the discourse. Hence, by the time a statement e1(x) is inserted in the database instance, there will not exist a statement e2(y) for which we can say a1(x,y).

a “creation of commitment„ or a “request„. Likewise they typically contain a cancellation of commitment. Discourses that contain a request typically also contain a declination of requests, and vice versa. If there is a complaint against customer or an acceptance of customer obligation, there is often also a description of the customer obligation, or a report of completion of customer obligation in the supplier,s organisation. The same holds for the corresponding speech acts about service fulfilment, i.e. complaint against supplier, et cetera. Whenever there is a speech act of the type “send bill„, there is typically also a description of the fulfilment of service, a registration of claims, and a speech act of the type “report of completion of customer obligation„, i.e. to register the payment.

5.1.2. Advice Regarding Speech Act Sequence

When the user has specified what speech acts that belong to a discourse, the tool can suggest sequence conditions. The advice mechanism contains a long list of possible diagrams. However, the basic idea is easy to understand. Figure 5-1 exemplifies the suggestions for some of the core speech act types. (The speech acts are named according to their types. „cancel commitment„ is not included in the diagram, in order to keep it simple.)

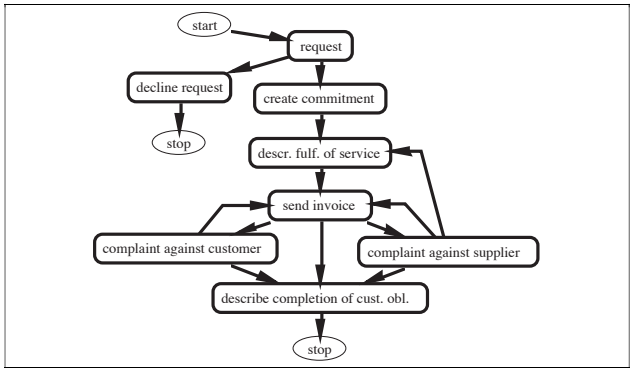


Figure 5-1: Illustration of suggestions of sequence links

Some of these suggestions are more probable than others. Consider, for instance, the speech act by which a customer complaints against the supplier. It is not obvious what should happen after this speech act. Should it be possible to send a new invoice (e.g. with reduced price)? Should it be possible to redo the work and report the completion again? These sequence links need to be checked and edited by the user. However, the basic structure in the sequence diagram is rather predictable. You do not send an invoice before the customer has issued an order. Neither do you start a discourse by a registration of payment, or end it with a request.

5.1.3. Standard Discourses

If the user has specified nothing at all about a discourse, the tool can suggest a standard solution, i.e. a set of speech acts with partial speech act specifications. There are different suggestions for inter and intra organisational communication. Only the former is supposed to have a contracting part. The tool can then go on and generate a sequence diagram, core predicates for the registration events, and inter speech act information links in the information model. Finally it can generate a (minimal) suggestion of intra speech act information. In this way the tool can be used similar to how Action Technologies, WorkflowManager works [1,8,11]. All the user needs to do is to identify that a contracting discourse exists. This can be useful in situations were the software system should only contain general information about the discourse. (Notice, though, that we have not included any mechanism for connecting one discourse with another, like in the WorkFlowManager, where one work flow may trigger another. How this could be done lies outside the scope of this paper.)

The advice mechanism for sequence diagrams and the standard discourses will be modified in the future based on empirical studies of how contracting discourses are typically implemented in business organisations. The major contribution in this paper is to sketch the general framework for the advice mechanism and to present a core advice mechanism based on common sense knowledge about contracting discourses. When doing this we have considered examples of information systems found in the literature, since these typically concern ordering procedures in business companies. We have also considered old existing case studies.

5.2. The Information Model

The information model is indirectly linked to the communication model, through the registration events (since a speech act can be described by or made through a registration event, which in turn updates a set of entity types). In this section we will focus on inter speech act information links. (See figure 4-6.) Figure 5–2 illustrate some typical inter speech act information links. These links and their mapping constraints can be generated by the COMMODIOUS tool. Notice how this is achieved by using the sequence diagram as an intermediate step in the advice mechanism. First the tool suggests a sequence diagram. The user confirms this diagram, or edits it. *When this is done*, the tool can generate the inter speech act information links. It would be a much harder problem to suggest standard solutions in the information model directly.

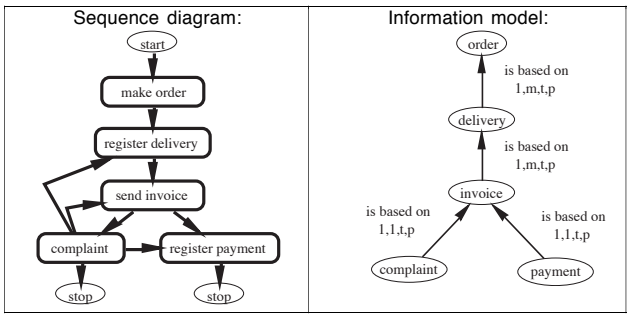


Figure 5–2: Illustration of inter speech act information links

Sometimes speech acts in a discourse contain references to information produced by other speech acts in the same discourse, e.g. an invoice may indicate what order it is based on. This is specified in the middle part of the form, labelled „References to other information types„. These attribute types are called “inter speech act information links„. All other attribute types and all entity types are called “intra speech act information„. (See figure 4-7.)

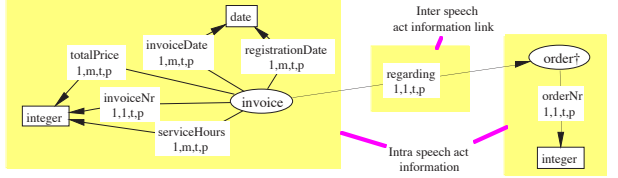
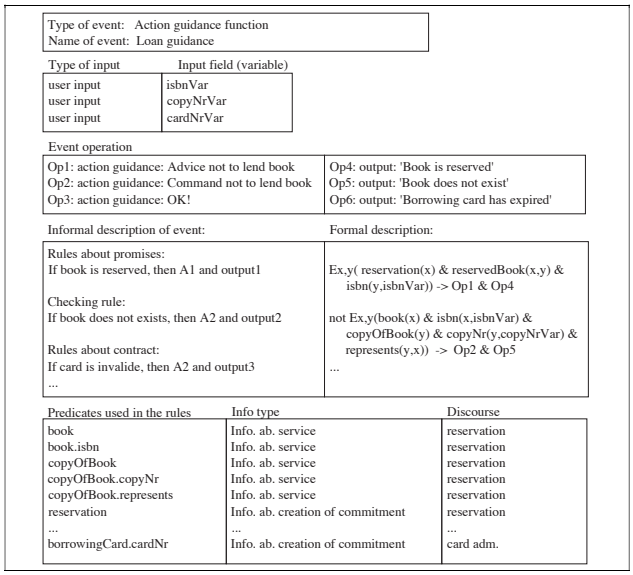


Figure 4-7: Intra and inter speech act information

The inter speech act information links and their mappings can be suggested by the advice mechanism in the COMMODIOUS tool. These links will be deduced from specifications in the communication model. Finally, the form for specifying a registration event contains an informal and a formal description of the update rule.

4.3.2. Support Functions

Figure 4-8 describes how the user will specify a support function. It consists of typed input fields, (system input or user input), a specification of even operations, an informal and a formal description of the event rules. Performance support functions must have triggering rules. All other support functions are supposed to be triggered by the user. Action guidance functions will have a set of guidance actions as event operations, e.g. to advice, warn, or instruct the users. To list guidance actions like this, is a way to characterise the output, before it is designed in detail.



Type of event: Action guidance function
Name of event: Loan guidance

Type of input	Input field (variable)
user input	isbnVar
user input	copyNrVar
user input	cardNrVar

Event operation

Op1: action guidance: Advice not to lend book	Op4: output: 'Book is reserved'
Op2: action guidance: Command not to lend book	Op5: output: 'Book does not exist'
Op3: action guidance: OK!	Op6: output: 'Borrowing card has expired'

Informal description of event:	Formal description:
Rules about promises: If book is reserved, then A1 and output1 Checking rule: If book does not exists, then A2 and output2 Rules about contract: If card is invalide, then A2 and output3 ...	Ex,y(reservation(x) & reservedBook(x,y) & isbn(y,isbnVar)) -> Op1 & Op4 not Ex,y(book(x) & isbn(x,isbnVar) & copyOfBook(y) & copyNr(y,copyNrVar) & represents(y,x)) -> Op2 & Op5 ...

Predicates used in the rules	Info type	Discourse
book	Info. ab. service	reservation
book.isbn	Info. ab. service	reservation
copyOfBook	Info. ab. service	reservation
copyOfBook.copyNr	Info. ab. service	reservation
copyOfBook.represents	Info. ab. service	reservation
reservation	Info. ab. creation of commitment	reservation
...	...	...
borrowingCard.cardNr	Info. ab. creation of commitment	card adm.

Figure 4-8: Form for specifying a support function (library example)

In figure 4-8, the rules are already formalised. However, before the user has reached a thorough understanding of the rules, he or she may be supported by getting a list of relevant predicates in the database schema. There is also a list of how these predicates are classified. By means of this list, the user can formulate informal rules, structured according to the classification of predicates. In a contracting discourse, the COMMODIOUS tool can suggest certain predicates as being relevant to a specific support function. In some situations, the tool can also generate typical informal and formal rules as a suggestion to the user.

Different types of events can trigger each other and give input to each other. Support functions that support the same activity will typically trigger each other, e.g. a support function for sending reminders to customers may trigger a performance support function that writes the letters. How this is specified lies outside the scope of this paper.

5. The Advice Mechanism

At last we have arrived at the essential part of this paper: how the COMMODIOUS tool can assist its users by making suggestions and checkings of the three models. Table 5–1 summaries the advice mechanism. All advice given by the tool can be edited by the user. The advice mechanism does not contain any “absolute„ knowledge about contracting discourses. It presents mere suggestions based on knowledge about how these discourses are typically implemented in organisations and in their software systems. The first two rows in table 5-1 are described in section 5–1, the next two rows are described in section 5–2, and the last row is described in section 5–3.

If this is specified for a contracting discourse,	the tool can do this
nothing	suggest standard discourses, including speech acts, sequence diagram, registration events, and parts of the information model
speech acts	check the discourse for completeness; suggest a sequence diagram and additional sequence conditions
speech acts, speech act specifications, and registration events	suggest core predicates for each registration event
all the above, sequence conditions, and core predicates for all registration events	suggest “inter speech act information links„ with mappings in the information model
all the above, and all relevant parts of the information model	suggest parts of the schema as input for action guidance functions; suggest concrete rules for some active action guidance functions

Table 5-1: An overview of suggestions made by the COMMODIOUS tool

5.1. The Communication Model

5.1.1. Advice Regarding Discourse Completeness

When the user has specified what speech acts that exist in a discourse the COMMODIOUS tool can check whether this is a reasonable and complete set of speech acts. Here are some examples of rules for checking discourse completeness in this sense: All discourses typically contain either

have software support, just like actions and speech acts. Even though no action is carried out, there is a decision which speech act that should be performed.)

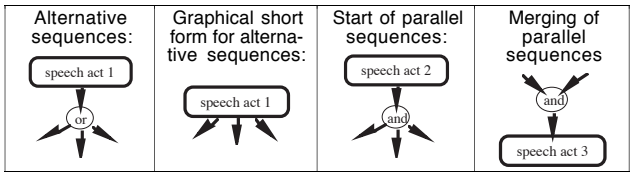


Table 4–2: Parallel and alternative speech act sequences

The user must also specify how the speech acts in group 1 (see figure 4-3) are sequentially related to the speech acts in group 2, and how speech acts in the two parts of a contracting discourse are related to each other. How this is done is illustrated in table 4–3. If the graphical diagram becomes too complex, the user can also choose to specify any part of it by means of such tables. Both representations generate rules in an underlying representation.

Additional sequence conditions for "Make reservation„	
1 What speech acts must precede this speech act?	"Give borrowing card„ and "Declare reservation period„
2 What speech acts may not precede this speech act?	"Invalidate card„

Table 4–3: Sequence conditions for the speech act "Make reservation„

4.2. The Information Model

So far the user of the COMMODIOUS method has only modelled the organisation from an abstract perspective. He or she has identified what tasks, actions, and discourses that exist. Now we shall go on and describe how the user can model the software system. It will consist of two parts, the information model and the system behaviour model. The information model constitutes the *database schema*. It defines what information items that can be stored in the *database instance*. The database schema contains definitions of predicates and a set of constraints. A predicate is either a *lexical type*, an *entity type* or an *attribute type*. A constraint can be either a mapping constraint or a typing constraint. Table 4-4 illustrates how this will look in practice for a small example. The schema also contains information about the system identifier for each entity type.

Database schema: P={string[1], person[1], socSecNr[2]} ©xy(socSecNr(x,y) → person(x) & string(y)) ∃x(person(x) → ∃y(string(y) & socSecNr(x,y))) identifier, person: socSecNr Database instance: Person(peter) socSecNr(peter,,630618-1234,)	Graphical illustration of the schema: string socSecNr m,m,t,p person

Table 4-4: An example of a very small database

The first item of the mapping in the graphical representation indicates whether the attribute type is single valued ('1') or multi valued ('m'). The second item indicates the same for the inverse of the attribute. The third item indicates whether the attribute type is total ('t') or partial ('p'). The forth item indicates the same for the inverse of the attribute. Since the COMMODIOUS tool needs to reason about the schema and its relations to the other models, there will be an underlying formalism that is different from the one presented in table 4-4. It will contain statements like: lexicalType(string), entityType(person), attrType(person, name,string), and mapping(name,1,m,t,p). Table 4-4 simply illustrates how the database is viewed by the user.

4.3. The System Behaviour Model

The system behaviour model contains no graphical parts, only forms for specifying events. There are two types of events, *registration events* and *support functions.*

4.3.1. Registration Events

A registration event updates the database by adding statements to the database instance. This update event contains information produced by a speech act, described in the communication model. It can also be information *about* the speech act. The underlying assumption is that the software system is used either as the *media* for the speech act or as a media for the act of *describing* the speech act. There can hence be two different links between a speech act and a registration event. These links are specified in the speech act specification where a registration event is said to "describe the speech act„ or to "mediate the speech act„. Figure 4-6 illustrates how the user specifies a registration event.

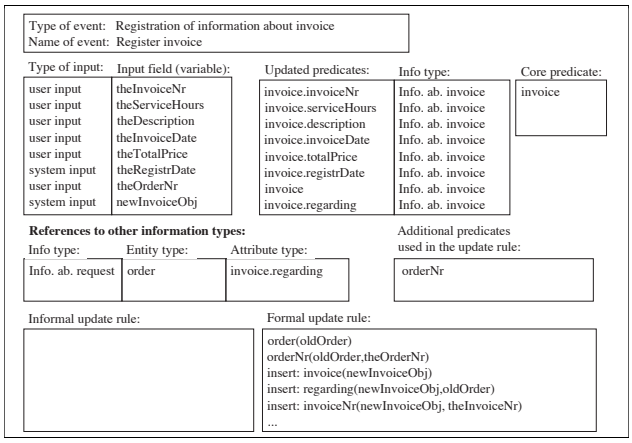


Figure 4-6: Form for specifying a registration event

There is a list of input fields (which are treated as variables) and another list of what entity and attribute types in the information model are updated by this registration event. (The notation invoice.invoiceNr refers to the attribute type invoiceNr of the entity type invoice.) The system automatically generates a classification of the event "registration of information about invoice„ and a classification of the information that is updated (Info. ab. invoice) from the classification of the speech act that is linked to this registration event, i.e. „send invoice„. Each registration event has a "core predicate„. (Section 5 explains how it will be used.)

This is what may happen in a contracting discourse: A speech act typed “make invitation„ is performed when the supplier informs the customer what kinds of service are available. A speech act typed “make request„ is performed when the customer order the service. A request may be inhibit by the customer, declined or accepted by the supplier. It is accepted by a speech act typed “create commitment„. The commitment may concern a long time agreement (contract) or it may be a commitment concerning a specific service occasion. A commitment may contain obligations also for the customer, e.g. in a library where the customers must return the borrowed books. Both the supplier and the customer may need to record the fulfilment of service and the fulfilment of customer obligations (describe fulfilment of service, describe fulfilment of customer obligations). This is done internally within each partners organisation. The supplier may need to register the claims. The supplier may report the completion of service and the customer may report the fulfilment of customer obligations. These reports may be accepted or declined by the other partner (complaint against supplier, complaint against customer, accept completion of service, accept completion of customer obligation). The supplier may also debit the customer. If this is done together with a report of the completion of service, we call it an invoice. The partners may also cancel the commitment before it is fulfilled.

Taxonomy for speech acts in contracting discourses:
Group 1: (fully sequenced): make invitation, make request, decline request, create commitment, describe fulfilment of service, register claim, describe fulfilment of customer obligations, report completion of service, debit customer, send invoice, report completion of customer obligations, complaint against supplier, complaint against customer, accept completion of service, accept completion of customer obligation, cancel commitment
Group 2: (partially sequenced): declare regulations, describe customer, describe supplier, describe the services
Group 3: User defined type of speech acts (specific to the company or application)

Figure 4–3: Taxonomy for speech acts in a contracting discourse

It may also be important to have regulations of various sorts, regulating parts of the behaviour in the discourse, like the loan duration in a library, invoice frequency for different types of customers, credit limits, et cetera. These regulations may be changed dynamically. Both the customer and the supplier typically need to have information about each other and about the services. Hence we need speech acts of the type describe customer, describe supplier, and describe the services. Finally there may exist speech acts that are specific to the company or the application, that does not fit any of the predefined speech act types.

The speech act taxonomy is divided into three groups. Each group will be treated differently when the user shall specify sequence conditions for the discourse (see below).

For each speech act in a contracting discourse, the user should make a speech act specification. This is illustrated in figure 4–4. The most important is to specify *in what way the speech act is going to be supported by a software system*. A description of the propositional content may be included. However, since we will describe the propositional content indirectly through the information model and the registration events, this part can be ignored. There are similar specifications for actions. An action specification contains information about the name of the action, the performer, the customer, instruments that are used, resources that are consumed, the product or output, and a specification how the action should be supported by a software system.

Speech act specification	
Name of speech act:	**Send reminder**
Type:	Complaint against customer
Discourse:	Reservations
Speaker	Librarian
Informal description:	-
Listener(s):	Library customer
Media:	Letter
Propositional content	loan(x), loanDate(x,d1), returnDate(x,d2), book(y) ...
Software support:	
Exist registration event?	Yes, describing the speech act:
Name:	**“Registration of reminder„**
Speaker:	Librarian
Listener(s):	-
Support functions:	
name:	type:
“List potential reminders„	action guidance function
“Batch for writing letters„	instrumental function

Figure 4–4: An example of a speech act specification

Speech act sequence:

For each contracting discourse (both the ordering and the contracting part), the user should specify how the speech acts may precede each other. This is done graphically for all speech acts in group 1. An example is given in figure 4–5. Each link means that the latter speech act may follow upon the former speech act. We also mark where the discourse starts and stops. By making a sequence diagram like this, the user can check the specification for completeness, by checking that each arch in the diagram ends in an appropriate way. It is, for instance, often not allowed to end a discourse with an invoice. The user can also check that all possible speech acts that may follow upon each node are included in the discourse.

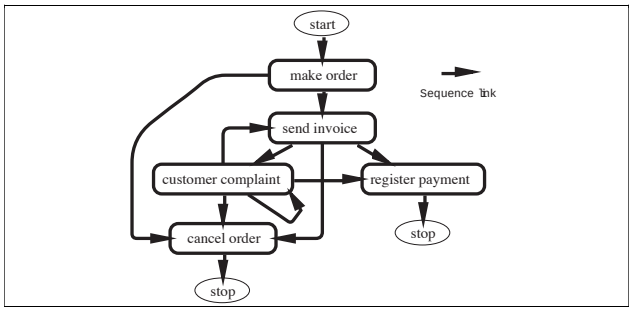


Figure 4–5: An example of a sequence diagram for the ordering part of a contracting discourse

Table 4–2 describes the graphical model constructs used in these diagrams. (The or-nodes in a sequence diagram can

example: In a library there is a task of lending books. If the development team wants to design a software support function for this task, they first characterise the type of support the system shall give. Should it passively monitor the information the user needs, e.g. information about existing borrowing cards and reservations, or should it process the information in the database and give more active advice to the user? Should it, for instance, advice the user not to lend a book that is reserved? Should the system prevent the user to register a new loan if a book is reserved? A design of an active action guidance function is directly or indirectly also a design of an organisational control mechanism. It must hence be evaluated with respect to the social institutions it creates or confirms. There is hence a need to discuss thoroughly, e.g.: What is the policy for this task? Can a book be lend to a customer, even if it is reserved for another customer? Who is allowed to overrule this policy?

Type of support function in the system	Type of activity supported
Resource supplying function	Action or task
Product storage function	Action or task
Performance function	Action or task / speech act
Instrumental function	Action or task / speech act
Action guidance function (active or passive)	Action or task / speech act

Table 3-1: A classification of software functions

4. Modelling with Commodious

4.1. The Communication Model

The communication model is meant to describe how people communicate in an organisation. As mentioned above, we will focus primarily on communication in so called contracting discourses. Before modelling these discourses in detail, we need to describe how they are related to other activities in the organisation. The basic idea is that whenever there is a customer-supplier relation, i.e. if a task or an external agents provides service to another task or external agent, there may exist a contracting discourse administrating the contacts between the customer and the supplier. Table 4–1 illustrates what graphical model constructs we use in the communication model.

Tasks:	Contracting discourses:	
Loaning	<contracting part> <ordering part>	
Actions:	Speech acts:	External agents:
Lend a book	**Make order**	*Library customer*

Table 4–1: Graphical model constructs

A contracting discourse can have two sub discourses. One sub discourse that contains speech acts dealing with long time agreements or contracts (the contracting part) and one sub discourse that contains speech acts dealing with the ordering procedure for each service occasion (the ordering part). The two parts will be decomposed separately, but they contain cross references, e.g. the acceptance of an order in the ordering part often requires that a contract has been established with the customer in the contracting part.

When we specify who is performing the activities, we need to refer to organisational agents and organisational units. They are not model graphically. The external agents play the role of being customers or suppliers to different tasks in the organisation. Figure 4–1 illustrate how we can model a library with this method. Each arrow in the figure indicates that one task or external agent is providing service to another partner. The core business task in the library is to lend books to library customers. This task, „Loaning„, is supported by another task, "Book adm.„, that provides the former task with books. This task, in turn, is supported by publishers.

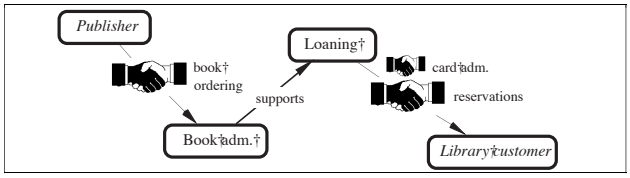


Figure 4–1: The library example

In the above example there are two contracting discourses, when the library orders new books from publishers and when they lend books to library customers. In the first case, there is no contracting part. In the second case there is. The library has long time agreements with its customers, i.e. the borrowing cards. The different parts of the contracting discourses are named by the users. When it comes to the two tasks "Book adm.„ and "Loaning„, there is no formal ordering procedure at all.

When we decompose a task, we may identify new potential customer-supplier relations and new contracting discourses. A contracting discourse, on the other hand, contains only speech acts. Figure 4–2 illustrates how we may decompose the two parts of the contracting discourse for loaning. The user has to specify the type of each speech act. The taxonomy for speech acts that may occur in a contracting discourse is described in figure 4–3. As mentioned above, it is based on the generic schema for "communication for action„ presented by Winograd and Flores. The speech act "give borrowing card„, in figure 4-2, is, for instance, classified as an instance of the speech act type "create commitment„. The classification of speech acts will be used later on by the advice mechanism in the COMMODIOUS tool.

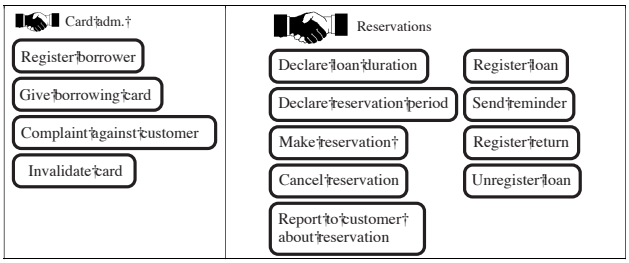


Figure 4–2: Decomposition of the contracting discourse for loaning

behaviour model. When we talk about their executions we will use the terms „registration event occurrences„ and „support function executions„. In the communication model we use terms like "tasks„, "actions„, "discourses„, and "speech acts„. We may also talk about "speech act performance„, "action performance„, and "task performance„.

In this paper we will focus on the more concrete benefits with the COMMODIOUS method, i.e. how the COMMODIOUS tool can assist its users by making suggestions and checkings of the models. Table 5–1 summarises the advice mechanism in the tool. However, before we can elaborate on this we must explain the framework. Section two and three deals with the basic terminology in the communication model and the classification of the organisational use of software. Section four describes the three models that are part of the method. The communication model is supposed to be created first. The system behaviour model and the information model can be created simultaneously, afterwards. Finally, section five describes how the user will be supported by the advice mechanism.

2. Basic Terminology

Table 2–1 clarifies some basic terminology used in the communication model. An *action* is a non-communicative activity that is performed by an agent in the organisation. It is atomic in the sense that it is not decomposed into sub components. A *speech act* is an atomic communicative action performed with symbolic information as its instrument. Sometimes people also use information when performing a non-communicative activity, like a sales man that sells books or journals, or an author that writes books. In such cases we will say that the information is a resource or the product of an activity. It is often stated that both actions and speech acts cause state transitions. However, in this framework we will not be concerned with these state transitions *per se*. The purpose of introducing the terminology is rather to illustrate how internal software events are related to certain types of user activities. The *agents* that perform activities are typically humans. However, sometimes we will also view the software system as an agent performing speech acts.

A *discourse* is a set of speech acts that are treated as a logical whole. A *contracting discourse* is a discourse that administrates the providing of service to customers. These speech acts are, e.g. "make order„, "send invoice„, "send delivery note„, "make agreement„, et cetera. The customers can be external or internal to the organisation. A contracting discourse includes both the inter partner communication and parts of the intra partner communication. There are general social conventions in our society for how this type of communication should be performed. The advice facilities in the COMMODIOUS tool are based on common sense knowledge about these conventions. They are also mirrored in the generic schema for "communication for action„ presented by Winograd and Flores, et al. [7,8,12]. In this paper we will only focus on contracting discourses. How other types of discourses should be described is not approached.

A *task* may consist of discourses, speech acts, actions, and other tasks. We use the term "activity„ to refer to either an action, a speech act, or a task.

Action: Ordinary action, which is treated as an atom and not further analysed. Speech acts are not included.
Speech act: An atomic communicative action, that is performed with symbolic information as its instrument.
Task: A set of actions and speech acts. It is synonymous to "business task„. (A task can also be decomposed into other tasks and discourses.)
Activity: Action, speech act, or task.
Discourse: A set of speech acts that are treated as a logical whole.
Contracting discourse: A discourse that contains speech acts that are meant to administrate the providing of service to a customer.

Table 2–1: Terminology in the communication model

3. A Classification of the Organisational Use of Software

One essential part of the COMMODIOUS method is to categorise the organisational use of software and information. Table 3–1 describes how we will classify the organisational use of software. We view the software system as containing different types of support functions for various activities in the organisation. A *resource supplying function* is a software function that provides the user with information that is viewed as a resource for the activity at hand. Consider a task of selling electronic books. A software system may store the books and make them available to a sales man, when he wants to deliver them to a customer. A *product storage function* is a software function that stores the product of the user activity. Consider a research organisation that sells reports. Whenever the task of writing a report is finished, people store the report in a database. A *performance function* is a software function that performs an activity in its whole, without being triggered or controlled by a human user. All robots can be considered to be agents performing tasks. Another example is when a system automatically orders new material, when the company's store is registered to be below a certain level. An *instrumental function* for actions or tasks is a function that is used as an instrument when performing the activity at hand, e.g. a word processor or a painting tool. An instrumental function for speech acts is a function that mediates the content of the speech act, e.g. an e-mail system. (An alternative term could be "media function„.) We will also talk about speech acts being mediated by a database system. In this case the registration event can be viewed as an instrumental function. An *action guidance function* is a software function that instructs the user when performing an activity. A passive action guidance function simply presents relevant information. An active action guidance function instructs the user actively.

This type of classification of software functions is meant to enhance the understanding and evaluation of a system specification. It can also guide the development team in how they shall go on in their analysis. Consider this

THE COMMODIOUS METHOD: COMMUNICATION MODELLING AS AN AID TO ILLUSTRATE THE ORGANISATIONAL USE OF SOFTWARE

by Peter Holm

SISU - Swedish Institute for Systems Development, Electrum 212, 164 40 Kista, Sweden, e-mail: pholm@sisu.se; and
Department of Computer and Systems Science, Stockholm University, Tel: +46 8 16 16 84; Fax: +46 8 703 90 25

Abstract[1]: This paper proposes a method for requirements capture, called the COMMODIOUS method. The basic idea is to produce an abstract model of organisational communication as a basis when formulating requirements on a software system. The user of the method should also specify how the software system is going to be used in this communication. Moreover the paper describes a support tool for this method, called the COMMODIOUS tool[2]. The tool supports its users by using an extended version of the generic schema for "communication for action,,, developed by Winograd and Flores, et al. The support is limited to situations where the software is meant to support communication between a customer and a supplier. The tool can check a model of communication sessions (discourses) for completeness and suggest standard solutions, i.e. what communicative actions (speech acts) that exist and how they may precede each other. It is also shown how parts of the information model are derivable from an abstract model of the discourse.

1. Introduction

In requirements engineering it is often assumed that an information system is supposed to capture some excerpt of the real world history. Hence, many methods for requirements engineering concentrate on conceptual modelling. Such a model can be used as a starting point when designing a database schema. This way of working prevents us from model the future database (or the information) *per se* as a part of the object system. In this paper we will take another route, where we view a database schema as a partial model of the organisational language. We will describe how the language in the database system is used in the organisation and for what purpose. Figure 1–1, illustrate the basic framework of the COMMODIOUS method. There are three inter related models: a communication model, an information model, and a system behaviour model. In the communication model we describe what communicative actions (speech acts) that are performed in the organisation. We also describe other types of business tasks and how they are related to the speech acts. When a speech act is performed, information may be inserted into the database. The information model, i.e. the future database schema, may thus be viewed as containing a partial schema for the propositional content of speech acts (confer: Searle [9,10]). By specifying registration events in the system behaviour model, that are linked to the speech acts in the communication model, we can keep track of what part of the information model that is updated by what speech act. Moreover, we describe software support functions and their relations to the activities that they support.

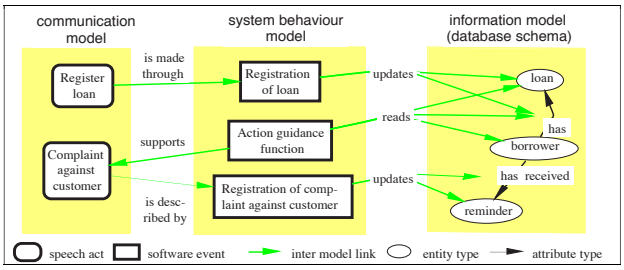


Figure 1–1: Illustration of the three models in the COMMODIOUS method

The system behaviour model and the information model constitute a software specification on a conceptual level, capturing the dynamic and static aspects respectively.

In database research it is common to talk about database *schemas* and database *instances*. It is also common to talk about process models and process executions. The model constructs used in the three models are all meant to describe the software system and its usage context on an abstract "schema,, level. The information model corresponds to a database schema. When we describe it, we will use terms like "predicates,, (= entity types, lexical types, and attribute types) and "constraints,,. If we refer to the database instance, we will use terms like "propositions,, and "constants,,. In the same manner we use the terms "registration events,, and "support functions,, in the system

[1] This work was supported by ESPRIT Basic Research Action 6353 (NATURE) which is concerned with Novel Approaches to Theories Underlying Requirements Engineering.
[2] This tool (a research prototype) is still being developed.