Univ Access Inf Soc (2012) 11:125–150
DOI 10.1007/s10209-011-0236-x

LONG PAPER

Whom to talk to? A stakeholder perspective on business process development

Albert Fleischmann · Christian Stary

Published online: 12 June 2011

Abstract Although many organizations operate in a process-driven way, few members are skilled in specifying and developing business processes—a skill that has become crucial for organization development, in particular to establish agile enterprises. This paper shows, on the basis of natural language constructs (subject, predicate, object) and communication patterns between actors (subjects), how individual members of an organization could contribute to coherent and intelligible process specifications. A language and tool supporting Subject-oriented Business Process Management (S-BPM) are introduced, allowing organizations to cope with strategic and operational challenges dynamically. As many organizations already work with BPM concepts and technologies, existing approaches to process modelling are also revisited with respect to representing natural language constructs and standard sentence syntax. Since most of them refer either to subjects, predicates, objects or to a respective combination, a roadmap can be developed for enriching existing modelling approaches. In doing so, organizations can benefit from stakeholder inputs for effective business process engineering re-using existing specifications.

Keywords Business process modelling · Subject orientation · Representation · Process prototyping · Business process execution · Agile business development · Organization design

A. Fleischmann
Metasonic, Pfaffenhofen, Germany
e-mail: Albert.Fleischmann@metasonic.de

C. Stary (✉)
Johannes Kepler University Linz, Linz, Austria
e-mail: Christian.Stary@JKU.AT

1 Introduction

Contemporary enterprises have to compete in the business environment by implementing processes and information systems addressing quality, cost, partner/customer relationships and structural flexibility [10]. They need to adapt at the stakeholder level to changing needs albeit increasing their operational efficiency and effectiveness (cf. [6, 18]). In order for flexible operation to be addressed accurately, management and stakeholders have to work on the associated processes [10]. They depend on the particular business objectives set by the enterprise and affect strategic, tactical and operational issues [45].

Business process specifications have to be tailored by participating parties to the specific situation at hand. Of particular importance are the intuitiveness of the notation and coherence of the modelling language. The first addresses the semantic distance to human understanding. It should be minimal, in the sense that it requires minimal cognitive effort to understand and communicate the represented information—organizational design is a good deal negotiating (cf. [22]).

When reflecting and redesigning the focus is on business process models and their specification, according to their nature as 'boundary objects' [5]: 'Boundary objects not only help to clarify the attitudes of other communities, they can also make a community's own presuppositions apparent to itself, encouraging reflection, and second loop learning.' As such, processes should be not only intelligible to all stakeholders involved in work procedures, but also to organization and technology developers [1]. Coherence of specification addresses the consistent propagation of strategic objectives, e.g., to establish the enterprise as innovation leader, to tactic and operational structures, e.g., reducing the innovation cycle times within the enterprise

on the tactic layer and establishing idea loops in production-process definitions on the operation layer (cf. [29]). As coherence also addresses the adequacy of operational structures, it has to be considered to be relevant for executable representations, either for automating business processes or simulating them (e.g. to anticipate changes). It is a challenge still to be met. According to Strosnider et al. [46], traditional flow-driven assembly and orchestration of service components are 'too rigid and static to accommodate complex, real-world business processes' (p. 415).

Language structures might help to structure the perceived world more accurately than any other form of representation, as recent empirical work in knowledge management reveals, even when information is presented by diagrammatic means (cf. [51] working with hierarchical networks that contain expression anchors for associations on different levels of abstraction). The work presented in this paper takes on such a structural perspective. The introduced Subject-oriented Business Process Management (S-BPM) approach follows the syntax of natural language syntax for standard sentences. The information is presented in diagrammatic form, focussing on interaction among actors (denoted as subjects). S-BPM also follows the principles of decomposition and hierarchy-specific associations. Conceptually, the subject, predicate and object of concern are of equal importance when modelling the reality as perceived by humans. Using identical formal representations leads to concise models in terms of flow control. Once the interaction patterns among actors (subjects) have been refined in terms of exchange of messages, suitable program code can be generated automatically.

S-BPM triggers a shift of process modelling paradigms centred on functions or activities to business entities to role-specific entities (actors, subjects). Subject orientation incorporates them into the structures of business processes guiding their decomposition for implementation at a service level. Executing process models in this way provides immediate organizational user experience, as it abstracts from implementation details in terms of programming languages or software execution.

This work builds upon the endeavours of enterprise modelling as started by Vernadat [48], taken up by model-based workflow design [44], and leading to active knowledge modelling (cf. [28]). Although recognizing the context of work processes, S-BPM does not claim to map business strategies to operational procedures in a traceable way (cf. [29]). S-BPM rather re-invents the business in terms of communication and goal-directed work-interactions (cf. [16]).

S-BPPM closes gaps between describing and experiencing collaborative work processes. As such, it does not follow a layered approach, in contrast to active knowledge modelling [28] separating community and personal work spaces (p.29). The community space is defined through communication relations. The novelty of the approach can be summarized by two key benefits, resulting for stakeholders and organization developers:

1. Stakeholders need only to be familiar with natural language (in particular, sentence syntax and semantics) to express their work behaviour *in terms of (e-)mail communication*
2. Stakeholder specifications can be processed directly *without further transformations*, and thus, experienced as described.

In this way, *non-disruptive and non-distracting round-trip engineering* can be established, reducing development effort through stakeholder specifications and automated execution without any transformation.

The paper is structured as follows. Section 2 reflects on the situation context when process specifications are utilized in organizations and the user requirements to that respect. As the current focus of BPM on functions and function-oriented flows of control does only partially help to meet user requirements, natural language and communication structures are closer to human behaviour and facilitate stakeholder-oriented BPM [33]. In this way, the acceptance of process models and process-based organization development can be increased. Section 3 introduces S-BPM, based on standard sentence semantics—subject–predicate–object, such as 'I prepare an invoice', and message exchanges between subjects (systems or stakeholders), such as 'Sending the invoice to the customer'. This section also demonstrates the coherent refinement procedure and shows behaviour specifications, including their execution.

In order to explore the development potential of existing modelling approaches towards stakeholder-oriented BPM, major modelling concepts need to be reviewed in terms of fundamental natural language constructs and standard sentence semantics. Section 4 starts out discussing predicates (activities), proceeds with objects addressed by activities and focuses on the subject as a starting point of work descriptions, and thus, of modelling and specification. Section 5 re-captures a study set up to observe stakeholders when modelling business processes with minimal structural inputs, confirming the proposed direction of stakeholder-oriented process development. Section 6 concludes the paper summarizing the achievements and sketching upcoming research.

2 Business process models as transformation enablers

This section provides the rationale for the conducted research in terms of current BPM activities. From a

hermeneutic perspective, system development is motivated as a natural language-driven transformation process of model descriptions to executable ones.

Organizations act and react on operational efficiencies through process design/redesign and execution (cf. [36]). There is empirical evidence for modelling languages being critical success factors for process-oriented organizations and Business Process Re-engineering projects [4]. Since organizations adapt to change, the process execution architecture (e.g. workflow management systems) and the prevalent processes must also adapt to the needs of the enterprise. However, in systems development, various schemes capture process elements: basically, natural language at the user/customer side, formal specification languages and programming languages at the developer side. Effects of such situation are of

- economical scale: costs, transformation effort
- social scale: conflicts, negotiations about the meaning of representations
- organizational scale: iterations, social tensions, mutual quality control
- cognitive scale: misunderstandings, individual assumptions that do not hold from an organizational perspective (cf. [19], [38]).

These effects are of particular importance when dealing with changes: 'As human cognition often perceives dynamic phenomena by developing a series of snapshots, capturing the true dynamics inherent on a process is challenging. By mistakenly taking snapshots to represent processes, there is a risk of tinkering with the wrong things, destroying natural controls that already exists, and essentially turning the organization into a jumbled mess of confusion [50]' (cited in [30], p. 15).

The mismatch of notational capabilities with respect to semantics also leads to misconceptions and incoherent transformation of information throughout analysis, design and implementation (cf. http://wwwcs.upb.de/cs/kindler/Forschung/EPCTools/). The semantic gap has recently been addressed in the realm of semantic web development [11], is well known in user-interface design [53] and has been elaborated for modelling [37]. Despite those findings, business processes are increasingly used to implement push technologies based on information systems, e.g., pushing tasks to users [7]. As such, the pragmatic quality of business process models is of outstanding importance, as users ground their knowledge on those representations [28].

The pragmatic quality is of vital importance when interoperability needs to be achieved, both on the level of organizations, and the level of information systems (development), enabling cross-boundary operation [1]. According to the Yankee Group [54], still a third of the overall costs could be cut when succeeding in achieving business and technical interoperability. This share is likely to increase due to the latest reports on the mismatch between business and technology modelling entities in the realm of the UN/CEFACT standardization efforts: '… we find it hard to gain any benefit from the conceptual distinction between Core Components and Business Information Entities.' ([49], p. 33).

The proponents of analytical approaches have not tackled the synthesis of modelling elements, even when addressing the context of business (cf. UN/CEFACT [47], p. 116):

- Business processes: common ones, such as Create Order
- Product classifications: Universal Standard Product and Service Specification (UNSPSP), e.g., for packaging machinery, passenger motor vehicles
- Industry classifications: International Standard for Industrial Classification (ISIC), e.g., for Automotive, Consumer Products
- Geopolitical context: ISO 3166.1 Country Code List
- Official constraints, such as Title 20 Restriction (for importing beef)
- Business process roles : common ones, such as supplier, carrier, seller
- Supporting roles: common ones, such as insurance company, chamber of commerce, certifying party
- System capabilities, such as GS1: RosettaNetPIPs:3.0, OASIS:UBL:V2.0, SAP: GDT:2.0

"Regarding the context driver principle, we perceived the definition of the context categories and the corresponding context value list as work in progress, which still needs further elaboration and evaluation." ([49], p. 33)

As the key to coherent modelling and representation, one can consider the integration of business context information in development specifications. As models in terms of specification represent the main means of communication between customers, users and developers (analysts, designers, programmers, evaluators), natural language could bridge communication gaps caused by modelling constructs of formal representations. Users and developers should benefit in terms of completeness and accuracy, as already intended by several projects—see for instance, http://web-imtm.iaw.ruhr-uni-bochum.de/iug/projekte/expect/start. Any mapping scheme should allow propagating the information from a value chain perspective to a software-development perspective in a coherent and consistent way, starting with specifications based on natural language. Today's process modelling notations and languages based on UML or other specification languages neither focus on natural language, nor on a role-specific behaviour specification. They rely on function-oriented flows of control for implementation, e.g., ARIS [39].

A natural language modelling approach could help to bridge the gap between informal and formal representations, due to its capability to reduce the cognitive workload for specification and to focus on relevant design elements. Although language can only be considered 'a window' into thoughts, as Pinker [34] puts it, language offers the richest source of evidence about perceptual and cognitive processes. And, it is the most natural and comprehensive tool for communication, as people need to communicate for model building and information system development (cf. language-action perspective introduced by Winograd [52] and applied to information systems development recently by Rittgen [35]).

As the so-called third wave of Business Process Management let companies and workers create processes on the fly, current process modelling tools are expected to support process responsibilities in managing their process themselves [43]. Just as spreadsheets provide direct manipulation of data, responsible needs direct manipulation of their business processes. For instance, an auto designer does not develop requirements for a new car and hand the specs over to the IT department for rendering; it is done on the designer's 3-D workstation. This type of end-to-end control is what business stakeholders need to build a process-managed enterprise. Business process management systems (BPMS) should directly execute open, standards-based business process models the way that a DBMS executes a SQL query.

BPMS are not only tools but provide frameworks for building adaptable processes. The Business process management initiative (www.bpmi.org), a consortium of software and service vendors, has defined several constituents of such systems:

- A business process modelling language (BPML),
- A process-designer interface called the business process modelling notation (BPMN)
- A simulator that can be used to 'flight test' new process designs.

Of major importance is the duality of expressiveness of any business process language in that context. It needs to be accurate to describe processes performed by stakeholders, such as of applying for a position, yet also precise enough to describe how computer systems have to communicate, in order to support that processes in dispersed multi-agent settings, in the absence of central control. In such a setting, organizations need not only means to conceive (new) processes, but to actually put them into action. Since operation managers cannot be expected to be process managers, process modelling needs to be done in a natural way and generate not more effort than filing a record to repository.

A corresponding BPMS needs a control flow that is not only sequencing functions, but rather coordinating agent-specific control flows (see www.bpmi.org for details). Processing systems must separate data from procedure and process, since only data could be structured in a predictable, reliable and stable way. A similar technique to the specification of business processes needs to be applied, except that these dynamic business activities are not stable or predictable. Because they are so dynamic, it is difficult to encapsulate them in software.

So far, the BPMI initiative has not been taken up by organizations (cf. www.bptrends.com). According to the analysis provided by Gartner, business process models either have been considered as technical means to proceed with Enterprise Application Integration, or to further develop Enterprise Resource Planning Systems or Workflow Management Systems. The conceptual importance and the natural handling of process models to develop the business in a networked ecosystem has not been recognized or tackled. This development still continues when looking to the standards committees. Neither the OASIS technical committee on BPEL (Business Process Execution Language), nor the W3C and OMG address handling of languages in terms of their usability for people operating their business.

This line of development also holds for recent approaches to stakeholder-oriented modelling, such as BPEL4People ([2, 27]). This kind of extensions still focuses on (web) service hierarchies and related execution issues rather than business roles and organization-centred task accomplishment. Typical stakeholder and task-related concepts are business administrator, or responsibilities of 'generic human roles'. The latter, however, are descriptors of how persons interact with processes in terms of process ownership and initiators. The ultimate goal of these approaches is process execution powered by services of distributed systems, rather than coherent business operations. The focus on execution has not changed, even when human tasks have been interpreted in terms of services (cf. [26]).

3 Specification of business processes according to standard sentence semantics

This section motivates and details the use of standard sentence semantics for the representation of business processes. The proposed subject-oriented modelling and execution scheme handles both demands, the one for natural specifications of actual business operations, and the processing of those specifications without loosing the context of specification. In contrast to most of the existing approaches to one of these respects, subject-orientation ensures a coherent organization and processing perspective. Its development is based on the following argumentation:

- When structuring reality, humans use subjects, predicates and objects—each of them can be mapped to natural language entities.
- Natural language supports human communication effectively, both in written and oral form.
- As humans use natural language structures as primary means to ensure mutual understanding, model descriptions for formal modelling could make use of it, in order to facilitate understanding models and ease the use of formal representations.
- Once formal specifications could be aligned to human concepts in this way, not only the communication between developers, experts and ICT users could be streamlined, but also misunderstandings could be reduced in the course of systems development.
- In order to ensure coherence of specifications, the exchange of messages determines the flow of control (in contrast to function-oriented approaches).

3.1 Natural language sentences and modelling

The introduced language for the description of process models captures the constituent elements of natural language sentences. The modelling language Parallel Activity Specification Schema (PASS) is based on the theoretical concepts (process algebra) provided by Milner and Hoare (see Sect. 3.4). It includes subject, predicate and object ([13, 14]). Model descriptions created in this way should be easier understood, and thus, should be usable in a straightforward way, compared to current modelling languages. Still, models described by this language can be transformed to executable code without programming.

The development of any information system or application starts with modelling that part of reality that is supposed to be supported. Models describe qualities and behavioural alternatives, including the interaction occurring in the technical and/or organizational environment. Models are transformed step by step into an executable application. The process of model creation is termed analysis in information systems development. In the course of an analysis, model elements are either considered essential or complementary. Scholz et al. [42] have named the essential model elements 'essential', and the supplementary ones 'Akzidenzien' (accidental factors). The latter are grouped around the essential elements, which trigger modelling processes (cf. [9, 42]) embodied in various modelling paradigms. Most of the traditional ones are listed below.

- Following a functional approach, accidental factors are grouped around functions, e.g., creating control flow diagrams or data flow diagrams according to de Marco [8].
- Using data-oriented modelling, accidental factors are grouped around data, e.g., setting up Entity-Relationship diagrams.
- In an object-oriented setting, e.g., using UML, accidental factors are grouped around objects (classes). It has become common in software and data engineering.

In the course of specification, models are described and documented, using representation schemes as those mentioned above. Modelling or analysing means to represent parts of the observed reality in terms of artificial languages. Models formulated in natural language terms allow for universal use, due to the familiarity of natural language in daily communication, and the availability of a standard semantics for sentences, comprising subject, predicate and object.

Here, the term standard semantics of sentences is used to denote level 2 of sentence semantics. According to Schmidt et al. [40], this level addresses three semantic roles: agent, predication and theme. Expressions, such as 'Mark enjoys tea time', are assigned to that level. Level 1 contains expressions like 'Tea is black', whereas level 3 allows semantic structures within sentences, such as 'Mark enjoying tea time reflects his day'.

Unfortunately, the use of natural language does not prevent from misunderstandings, in contrast to using formal languages. They provide word semantics, but simplified sentence semantics. It is the latter leading to difficulties in understanding formal models for non-trained readers. When they transform the perceived information into sentences of natural language, the addressed incompleteness is like to lead to non-conform pragmatics. In this way, empirical evidence can be gained that humans are used to communicate in complete sentence semantics formed by subject, predicate and object. This standard sentence semantics of natural language subject–predicate–object has triggered the development of S-BPM. In particular,

- A *subject* is the starting point for describing a situation or events,
- Activities denoted by *predicates*, whereas
- An *object* is the target of an activity (denoted by a predicate).

The distinction between essential and supplementary aspects can be kept for the natural language approach. Humans use passive sentences in case they do not reflect or want to ignore who acts or triggers an action in a certain situation. Artificial or formal languages should support complete standard semantics for sentences, in order to prevent from misunderstandings.

The increasing utilization of business process models specifying business-relevant event chains seems to shift the

attention to modelling towards subjects, since it denotes the starting point of any activity. So far, existing modelling approaches tend to focus on predicates or objects, adding the subject for natural language explanations of the represented information. However, information systems have to be considered as subject-sensitive (cf. [3]), i.e., in the context of role-specific activities.

The interplay between roles or employees and activities is specified through business process models, both on the level of individual work tasks, and on the organizational level. Applications are assigned to automated tasks or interactive chains of executions. In business process models acting roles, e.g., the employees are distinguished from predicates defining the activities of acting roles, and objects denoting the purpose of these activities. In the course of accomplishing their tasks, they receive work inputs and pass on results. Hence, interaction and communication, either direct or indirect, are to be considered as an essential activity of acting roles for subject-oriented modelling and specification.

S-BPM is exemplified using a common process application. Employees have to apply for going on holidays or taking days off. This allows to demonstrate the fundamental and supplementary aspects of the standard semantics subject–predicate–object in the various modelling approaches. Figure 1 shows the natural language description of the respective process.

This simple Holiday Process will be modelled following two different approaches. They differ by the starting point of building a process specification. One approach starts with a generic process model that is restricted step by step. All involved actors or systems might interact mutually. The lines of interaction need to be adapted to those required for task accomplishment. The other approach starts with an empty model, and the process model is constructed step by step. Task-relevant actors or systems need to be identified as the process specification evolves, and the lines of interaction need to be included as required for task accomplishment. Figure 2 shows the conceptual difference between the restrictive and the constructive approach to modelling.

Holiday application procedure:

An employee fills in a holiday application form. He/She puts in a start and end date of his/her planned vacations. The responsible manager checks the application and informs the employee about his/her decision; the holiday request might be rejected or approved. In case of approval the holiday data are sent to the human resource department (HR) which updates the days-off the holiday file.

Fig. 1 Natural language description of an application process for holidays

In the following, both approaches will be explained in detail. In sub Sect. 3.2, the stepwise reduction of interaction between actors or acting components is explained. In Sect. 3.3, the stepwise creation of a communication-based process model is detailed. In both cases, actual or envisioned work processes need to be represented in a transparent and traceable way.

3.2 Ensuring coherence restricting communication

The subject-oriented representation scheme focuses on the direct communication between all the parties involved in a process. In order to distinguish concrete persons from roles in a process, the parties in a process are called subjects. Subjects are the acting elements in processes, similarly to sentences of natural languages where the subject is also the acting element.

Figure 3 shows a generic subject-oriented specification scheme with 3 involved parties. It fits to the holiday application process, as the 3 subjects are employee (Subject1), HR department (Subject2) and manager (Subject3). Each of the parties exchanges messages with another party. This generic process is restricted step by step in order to get a process specification for the holiday process as described in Fig. 1.

Each subject starting a message exchange is marked with a small white triangle (subject1).

Each subject can send messages with the name Message to any other subject any time. Figure 3 shows the

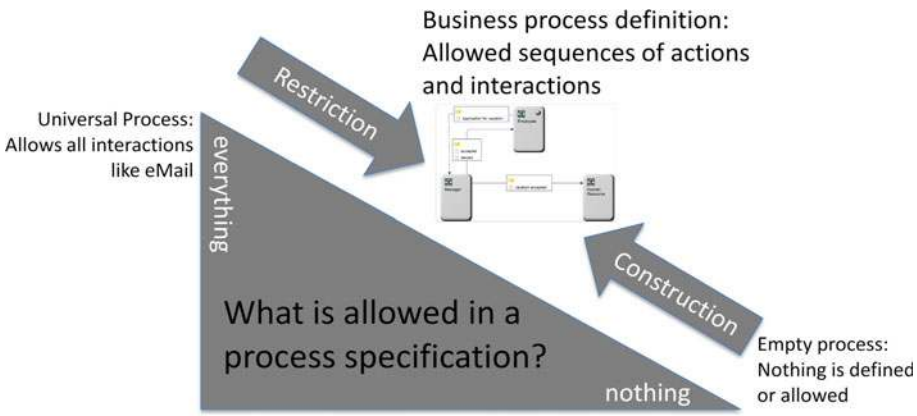


Fig. 2 Approaches to define business processes

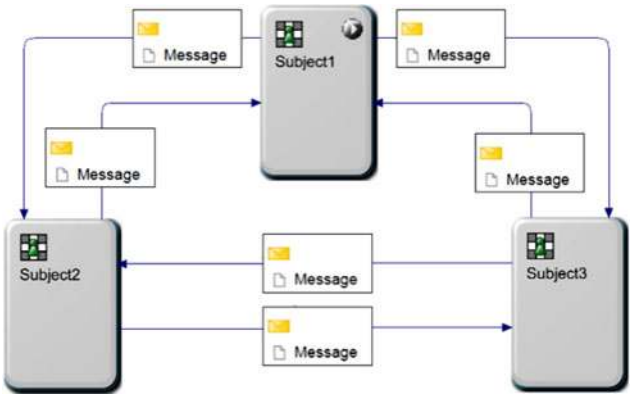


Fig. 3 Subject-oriented representation schema for 3 party process

behaviour of the subject with the name Subject1. Since Subject1 is the subject that starts a process its start state is the state select. The start state is marked with a thick frame. The state 'start' and the transitions to the state select will be never executed in the start subject. This state is the start state in all the other subjects. All the other subjects are waiting for a message from all the other subjects.

In this way, all subjects that are not start subjects have to receive at least one message before they can start to send messages. The start subject sends a message to any other subject. The receiving subject can reach now the state select. In that state, any subject can decide upon its next action without restriction. A subject that is in state select can send a message to other subjects that are still in the state start. Now, these subjects can also reach the select state and can send messages. Finally, all subjects are in the state select and can communicate when addressed.

In the select state, the start subject decides whether it wants to send or to receive a message. In order to start a workflow, it does not make sense to receive a message because the other subjects are waiting for messages. All the other subjects are in the state start that is a receive state. This means that the start subject will start with sending messages. Now, the message exchange can begin. In the select state, a subject decides to use the send transition. In the state 'prepare message and select address', the subject fills out the business object that is transmitted by the message 'message'. After that, the subject decides to which subject the message with the business object as content will be sent (Fig. 4).

In the select state, a subject can also decide whether it wants to receive a message. If a message from the expected subject is available, the message can be accepted and a follow-up action can be executed. It is not specified what the follow-up action is. This is like receiving an e-mail. The receiver can interpret the content of an e-mail and knows what the corresponding follow-up action is. The abort transitions back to the select state enable to step back in case a subject has made the wrong choice (Fig. 5).

The representation scheme can be easily created for any number of participants, following the same principles as shown for three parties. The behaviour of each subject has to be adapted to the corresponding number of subjects in a process. It is necessary to add the corresponding send and receive transitions between corresponding states. In the send area, transitions must be added to send a message to the new subject, as for the receive area. In the receive state, a corresponding transition has to be added. With that extension schema, the behaviour for each type of multi-party process can be generated automatically.

With the message 'Message', a corresponding business object is sent. The structure of this business object corresponds to the structure of a mail with some extensions like keyword and signature. Figure 6 shows the specification of the business object message in a XSD notation.

Whenever a message 'message' is sent, such a business object is sent. The values for the components of the business message object correspond to the content of a traditional mail.

Following the generic subject template approach, the flow of messages has to be restricted according to the business case. The corresponding S-BPM procedure requires several restriction steps:

1. Specify a generic template according to the number of parties involved in handling a certain business case (cf. Fig. 2)
2. Remove message connections between subjects which are not required
3. Name the subjects according to the application domain
4. Name messages and introduce message types according to the application domain
5. Adapt specification to actual subject behaviour.
6. Refine the structure of the business objects transmitted by the various messages

Following these steps, a process specification is constructed corresponding to the business requirements. In the specific example, these steps result in the communication structure shown in Fig. 7, and the behaviour specification of the subject employee shown in Fig. 8.

With each restriction step, the guidance for the subject holders is becoming more stringent to task accomplishment. In this way, a subject-oriented system specification can guide the parties in a process. It can be used to produce an execution protocol recording the sequences messages have been exchanged between the involved parties. Another advantage is that a workflow can be generated automatically (see Sect. 3.4). The only parameter that must be known is the number of involved parties besides the subject starting the process execution.

For the specification of an actual workflow, the various subjects of a process must be assigned to existing roles and

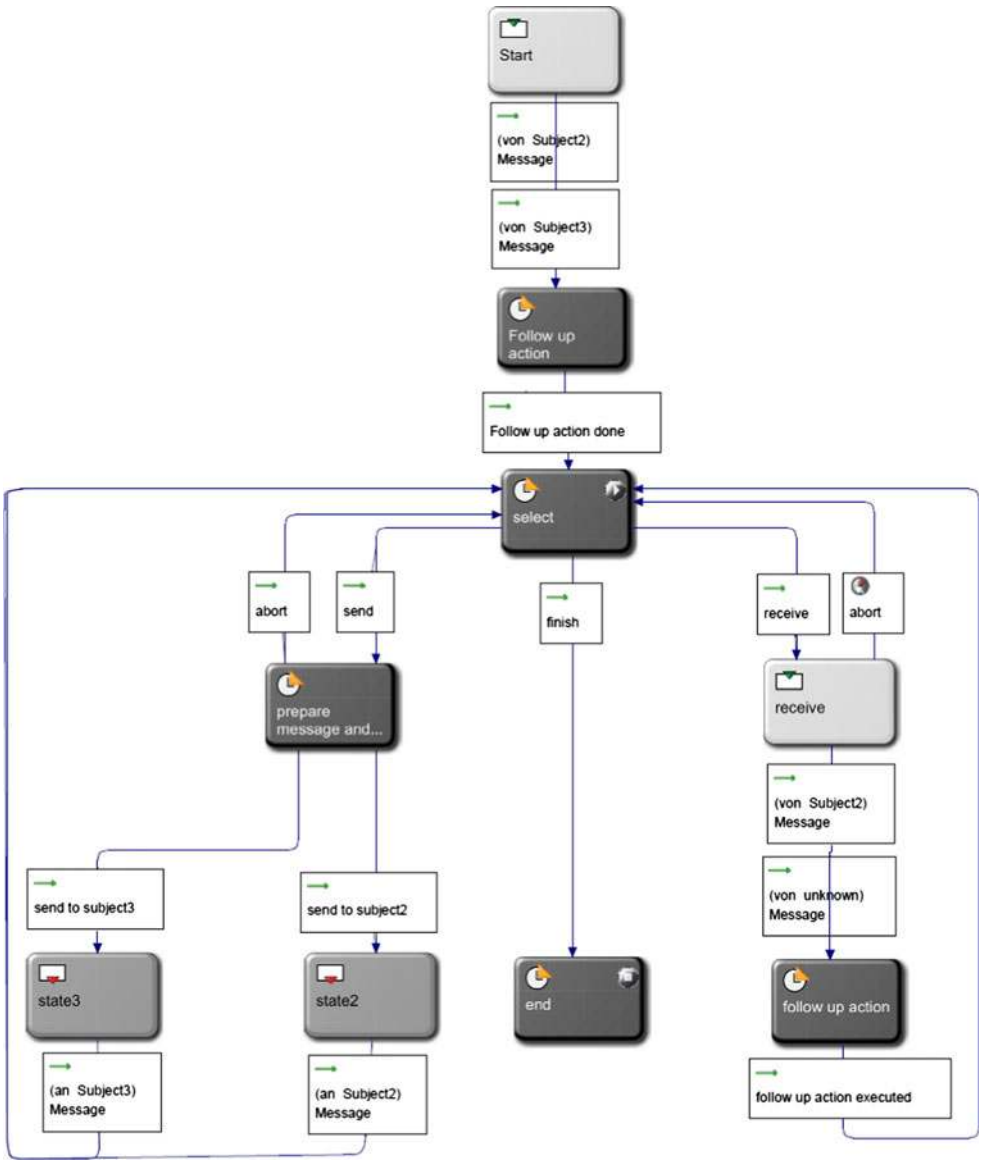


Fig. 4 Generic behaviour of the start subject 'subject1'

persons or agents. The following example shows such an assignment for the three-party process scheme (Fig. 9).

The persons Max Mustermann, Tobias Heinzinger, Uwe Hofmann und Johannes Luther are assigned to subject 'employee'. Since these persons are assigned to the start subject, all of them can start the process. For instance, Max Mustermann creates the message Application for vacation and sends it to Nils Meyer. Nils Meyer, who is assigned to the subject manager, can accept that message and can send a message Accepted or Denied back to subject employee—the message is received by Max Mustermann because he is assigned to the subject employee. Max Mustermann receives the message because in his environment or context, the process is started. If another person assigned to subject employee starts a process, this process instance is executed in his or her environment.

3.3 Constructive behaviour specification

The procedure shown in Sect. 3.2 follows a step-by-step instantiation and restriction process ensuring specification coherence due to its generic scheme. As subjects are abstract resources representing the parties involved in a process, the modelling process might start with identifying the involved subjects and after that define the behaviour specifications of acting parties. Such an approach also leads to a coherent representation, as all required exchanges of messages have to be specified for the procedure to be completed. This time, however, each subject can be directly addressed, rather than instantiating a generic process pattern. In the following, this alternative is illustrated.

Figure 10 shows the identified subjects and the messages they exchange.

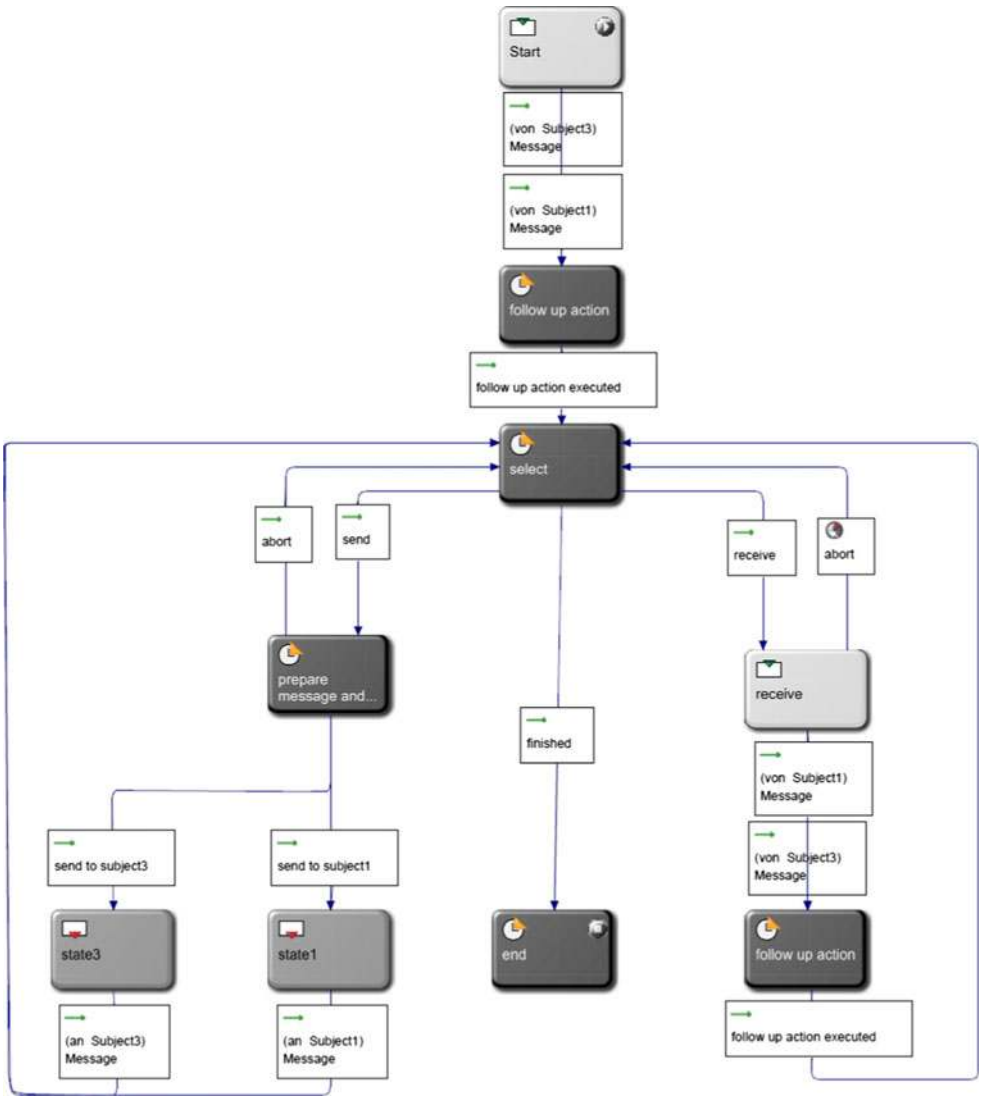


Fig. 5 Generic behaviour of subject2

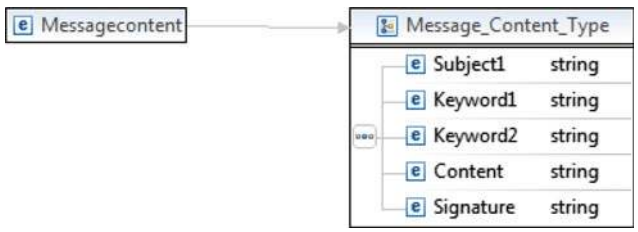


Fig. 6 Generic structure of the mail business object

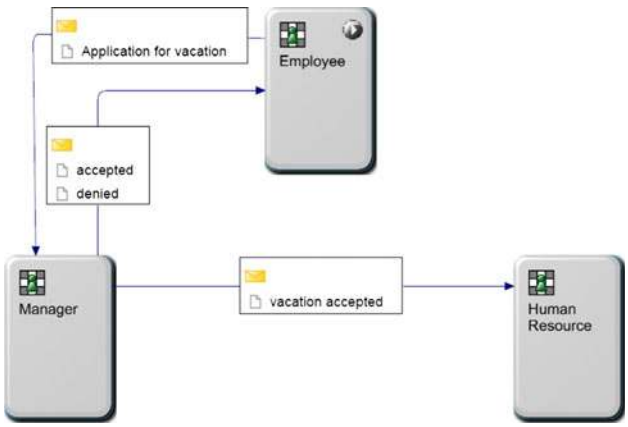


Fig. 7 Subjects and exchanged messages

When specifying the behaviour of each subject, as shown in Fig. 11 for the employee, again, a sequence of sending and receiving messages and activities to be set for task accomplishment need to be represented. The initial state on top is marked in green. In this state, the employee fills in a holiday application form. Upon completion, the employee's state switches to the next state via the transition 'holiday application completed'. This state is a sending state. In this state, the holiday application is sent to the manager. After successful sending, the employee reaches the state 'answer of manager' waiting for approval or rejection. This state is a receiving state. In case of rejection,

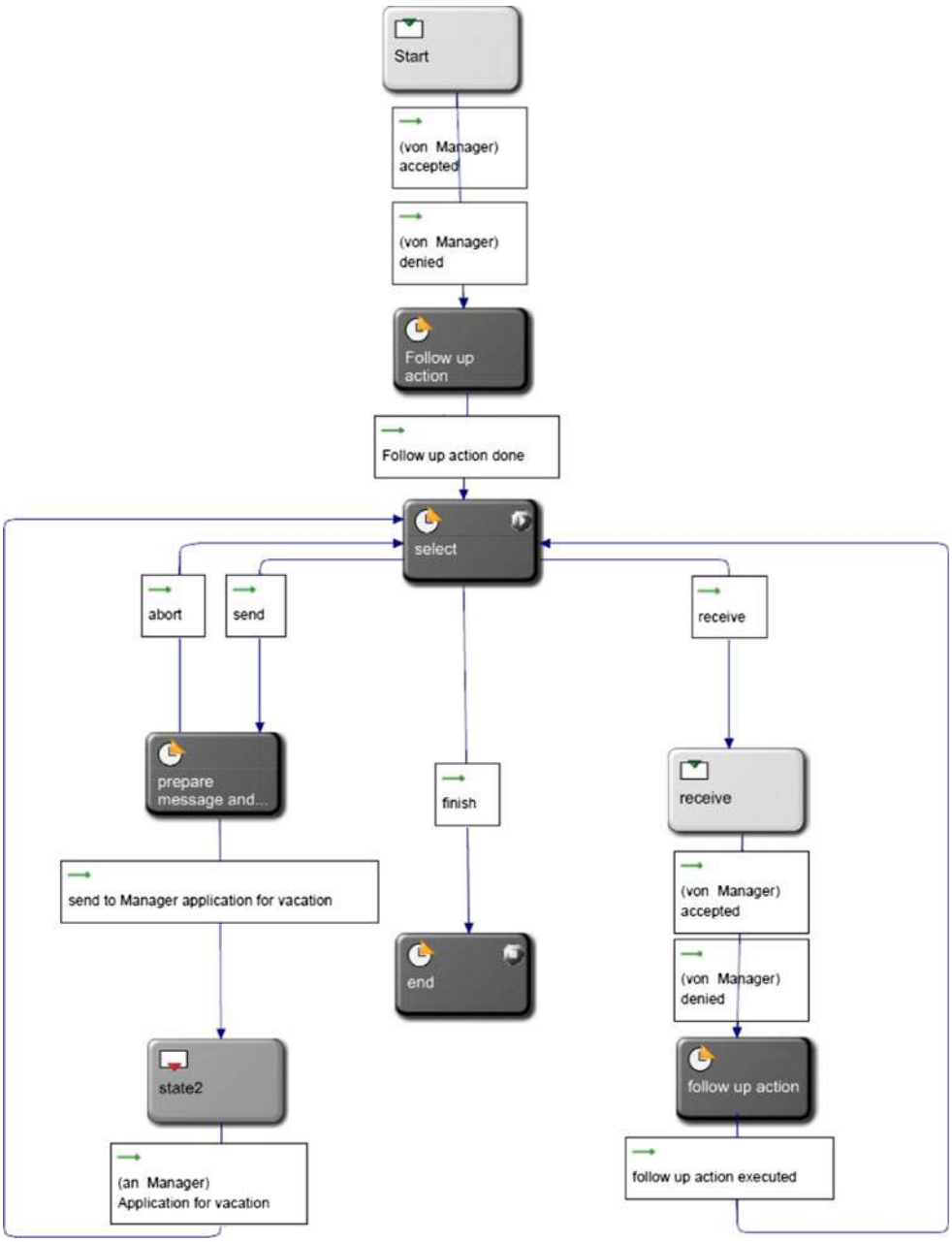


Fig. 8 Instantiated behaviour of subject employee

the process terminates. In case of approval, the holidays can be taken as applied for. Upon return of the employee, the holiday application process also terminates.

The behaviour of the manager is complementary to the employee's. The messages sent by the employee are received by the manager and vice versa. Figure 12 shows the behaviour of the manager. The manager is on hold for the holiday application of the employee. Upon receipt, the holiday application is checked (state). This check can either result in an approval or a rejection, leading to either state, informing the employee. In case the holiday application is approved, the HR department is informed about the successful application.

Finally, the behaviour of the HR department has to be detailed. It receives the approved holiday application and puts it to the employee's days-off record, without further activities (process completion).

So far the model includes:

- The subjects involved in a process,
- The interactions they are part of,
- The data they send or receive through each interaction, and
- The behaviour of each subject.

The description of a subject defines the sequence of sending and receiving messages, or the processing of

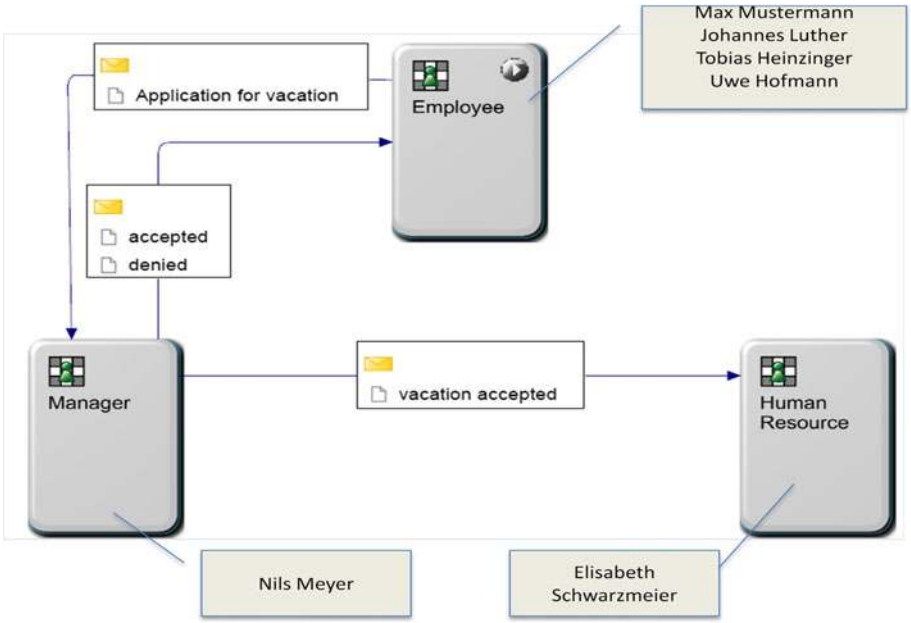


Fig. 9 Instantiating a process scheme

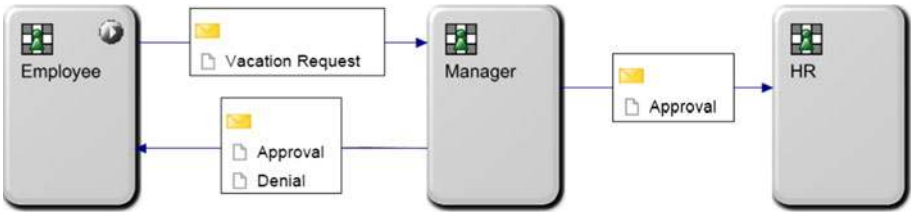


Fig. 10 Communication between identified subjects

internal functions, respectively. In this way, a subject specification contains the pushing sequence of predicates. These predicates can be the standard predicates like 'send' or predicates dealing with specific objects, such as required when an employee files a holiday application form (see Fig. 13). Consequently, each node (state) and transition has to be assigned an operation. The implementation of that operation does not matter at that stage, since it can be handled by object specifications. As we abstract from implementation details, it seems suitable to replace the term operation with the more general term service.

A service is assigned to an internal functional node. If this state is reached, the assigned service is triggered and processed. The end conditions correspond to links leaving the internal functional node.

Each result link of a sending node (state) is assigned to a named service. Before sending, this service is triggered to identify the content or parameter of a message. The service determines the values of the message parameters transferred by the message. Analogously, each output link of a receiving node (state) is also assigned to a named service. When accepting a message in this state, that service is triggered to identify the parameter of the received message. The service determines the values of the parameters transferred by the message and provides them for further processing.

These services are used to assign a certain meaning to each step in a subject. Services allow defining the predicates used in a subject. All predicates are triggered in a synchronous way, i.e., a subject only reaches its subsequent state when all triggered services have been completed. Figure 14 shows how the predicates of a subject are defined by means of objects.

3.4 A first approach to processing

In computer science, the introduction of parallel processes has drawn wide attention (see also [12]). A process in general executes activities or actions in a certain time interval to achieve a certain goal (cf. [20]). A process description defines the behaviour of a process. As it will be shown below, this concept relates to the introduced understanding of subjects and the process of subject-oriented modelling.

In standard sentence semantics, the subject is the initial point of an activity or action as defined by the predicate. Hence, subjects represent the active elements of the observable reality. Subjects can process defined sequences of activities (predicates). Subjects are mutually independent and might communicate with each other, exchanging information. Given this understanding, subjects, to a great

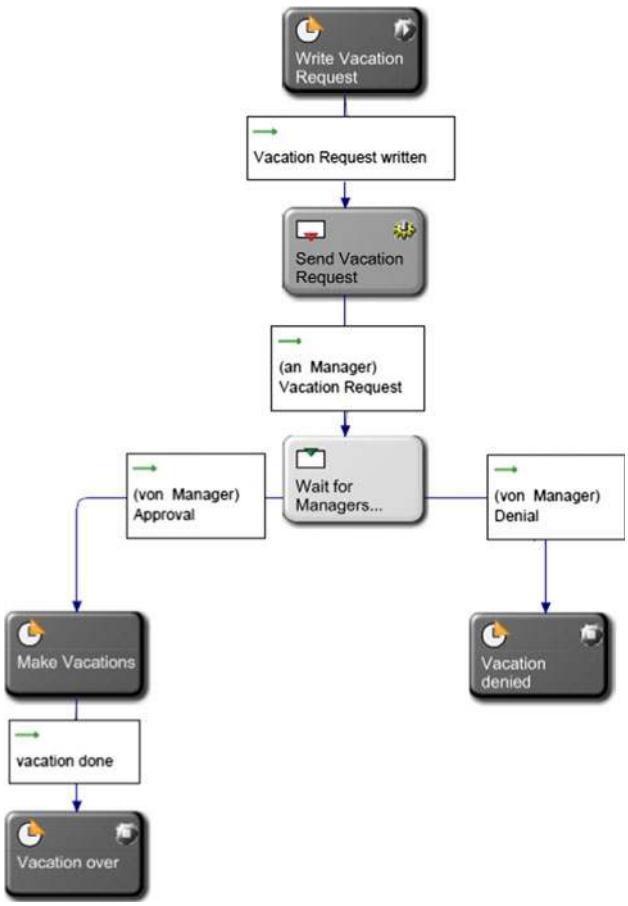


Fig. 11 Employee behaviour in holiday application process

extent, correspond to processes as defined in computer science. The concept of process enables the mapping of subjects into corresponding model constructs.

In the following, two major concepts are introduced, the Calculus of Communicating Systems (CSS), and Communicating Sequential Processes, both dealing with processes as a primary source for modelling. In these approaches, parallel processes synchronize themselves via the exchange of messages, i.e., a process is able to send and receive messages via so-called ports. Sending and receiving are the only possible predicates available. The ports can be considered as objects of the standard sentence semantics, as they are essential for the dynamic aspect of the concept.

3.4.1 Calculus of communicating systems

Calculus of Communicating of system (CCS) is one of Robin Milner's developed process algebras [32]. Process algebra is used for algebraic modelling of parallel processes. It consists of elementary activities (elementary actions) and operators associating actions. Elementary actions cannot be decomposed to smaller categories of activities. Processes can interact with the neighbour processes (i.e., located at the same level of abstraction) or execute independent actions in parallel. CCS targets towards modelling of communication between processes.

A process uses ports for communication with other processes. Each port has a name, either for sending or receiving messages. Figure 15 contains the processes or subjects for holiday applications. The employee sends a completed holiday application form to the manager. Sending ports are denoted with an upper line. The manager sends his/her decision to the employee, and, in case of approval, the approved holiday application to the HR department.

In Fig. 15, only the involved processes and their relations have been included. Neither internal processes nor internal behaviour has been made visible here. They are described using a variety of operators. Figure 16 shows part of them, referring to involved processes in the holiday application process.

According to Fig. 16, the process Employee initially sends the holiday application. Subsequently, it waits either for the rejection/approval message. Upon receipt, the NIL operation is executed—the process stops. The description of the processes Manager and Management can be interpreted similarly. The last line of Fig. 16 shows the decomposition of the entire process using the respective operator.

The example reveals the actor to be essential in CCS, whereas predicates and objects are considered accidental factors. CCS can be considered as a subject-oriented approach.

3.4.2 Communicating sequential processes

Communicating Sequential Processes (CSP) is also a process algebra introduced by Tony Hoare [25]. Originally, CSP was published as a program-linguistic construct (Hoare, Communicating Sequential Processes [24]), before being formalized, influenced by Milner. CSP, in contrast to CCS, does not distinguish between sending and receiving. In case operators connect processes, events of the same name of the linked processes are also connected.

Figure 17 shows the holiday application process in CSP notation. An employee might trigger the event 'holiday application' before the event 'rejected' or 'approved' can occur. The event SKIP denotes a process to be finished. The process Manager captures the event 'holiday application' and its subsequent events. Once the process Employee is connected with the process Manager (using the ‖ operator), they have identical start events and transitions (directed link in the diagram)—see bottom line of Fig. 17.

In CSP, events might be refined to send and receive sequences, operating on ports and transferring data. In this way, in CSP, the predicates 'send' and 'receive' can be used, as well as objects (messages) processed by these

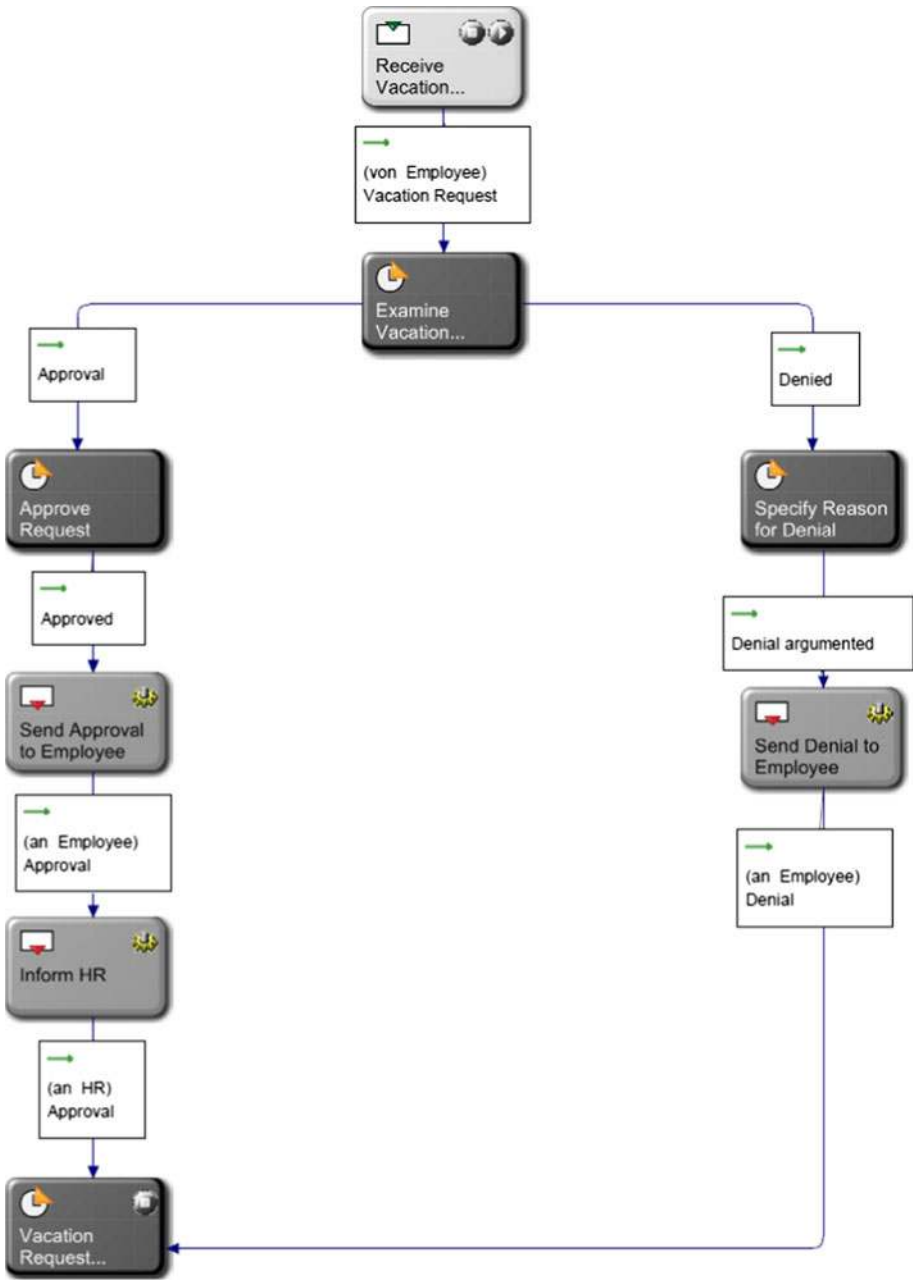


Fig. 12 Manager behaviour in holiday application process

simple predicates. However, CSP focus on subjects as essentials like CCS. Predicates and objects play a minor role. Both lead to incomplete models with respect to semantic sentence semantics, since natural language supplements for predicates and objects have to be provided to complete model—intelligible naming does not contribute to the completeness of standard sentence semantics.

3.5 Tool support

In principle, the execution of subject-oriented representation schemes can be supported by an appropriate workflow system. As a first choice, a simple mail system can be used, although requiring strong discipline from the involved parties. In the current example, a workflow system is used especially developed for subject-oriented process specifications and workflows (cf. [14, 15]). In case process modelling is started with the generic process pattern, each restriction step can be executed with that workflow system. The following example shows how the workflow system is used for the 3-subject process template. This template is used to execute an application for vacation.

Figure 18 shows a screenshot as Max Mustermann creates a new instance of a 3-party-process scheme with 3. This new process instance has the title Request for Vacation (see the circle in Fig. 18).

Fig. 13 HR department behaviour in holiday application process

After creating the process instance, Max Mustermann is guided through the process. He is asked by the workflow system which transition he wants to follow. He knows that he has to fill out the business message form with the corresponding data and that form has to be sent to Nils Meyer. Consequently, Max Mustermann follows the transition 'send'. In the state 'Prepare Message and select Receiver' following the transition 'send', he fills out the business object with the data required for an application for vacation.

In Fig. 19, the user interface of the workflow system is shown.

- Number 1 in that figure shows the name of the current state: 'Prepare Message and select Receiver'
- Number 2 shows the title of that process instance: 'Request for vacation'
- Number 3 shows the creation date of that process instance
- Number 4 shows the form for filling out the business object.

Max Mustermann can add all the required data for a vacation request to the business object and sent it to Nils Meyer who is the owner of subject2. This is all Max Mustermann needs to know, since the behaviour description of his subject would allow sending the vacation request to the human resource group. This is analogous to using a mail system. Every user must know to whom he/she has to send a mail with which content. The workflow used in the example produces a protocol recording executing a certain action at a certain time. Fig. 20 shows an example of an execution path for handling a vacation request with a universal process. The steps executed in each subject are shown in a corresponding column with the subject name as head line.

Subject1 started with the select activity, and the send transition was selected. After that the action 'prepare message and select address' is executed and in state 'state2' the message was sent to subject2. Now subject1 reaches again the state 'select'. In state 'Start', subject2 receives the message. In the following state 'follow-up action', the content of the received message is read, and the corresponding action is executed by Nils Meyer who is the owner of subject2. In the case of the vacation application, this follow-up action is Nils Meyer′s decision whether the vacation application is accepted or denied. This decision must be sent to subject1. In state select, subject2 decides to follow the send transition, prepares the message with the result of the decision and sent it to subject1. This swim lane diagram shows which subject executed which actions in which sequence. If a subject sends a message, the sending state is connected with the corresponding receive state in the receiving subject. Subject1 sends a message to subject2 in state 'state2'. Subject2 receives that message in state 'Start'.

4 Standard sentence semantics in existing modelling approaches

The following lays the ground for enriching existing paradigms and modelling approaches using the standard/sentence semantics, comprising subjects (actors), predicates (activities, functions), objects and their interplay for complete specifications. The elements represent modelling and design dimensions for organization development: the WHO (subjects), the WHAT and WHEN (predicates and their sequencing) and USING WHAT (objects, data).

4.1 Predicates or activities

As long as modelling is focused on activities, predicates are essential, whereas subjects and objects are considered as accidental factors or supplementary elements. Such a focus is in line with algorithmic thinking, as computer systems have been constructed to solve complex arithmetic problems rather than processing huge amount of data.

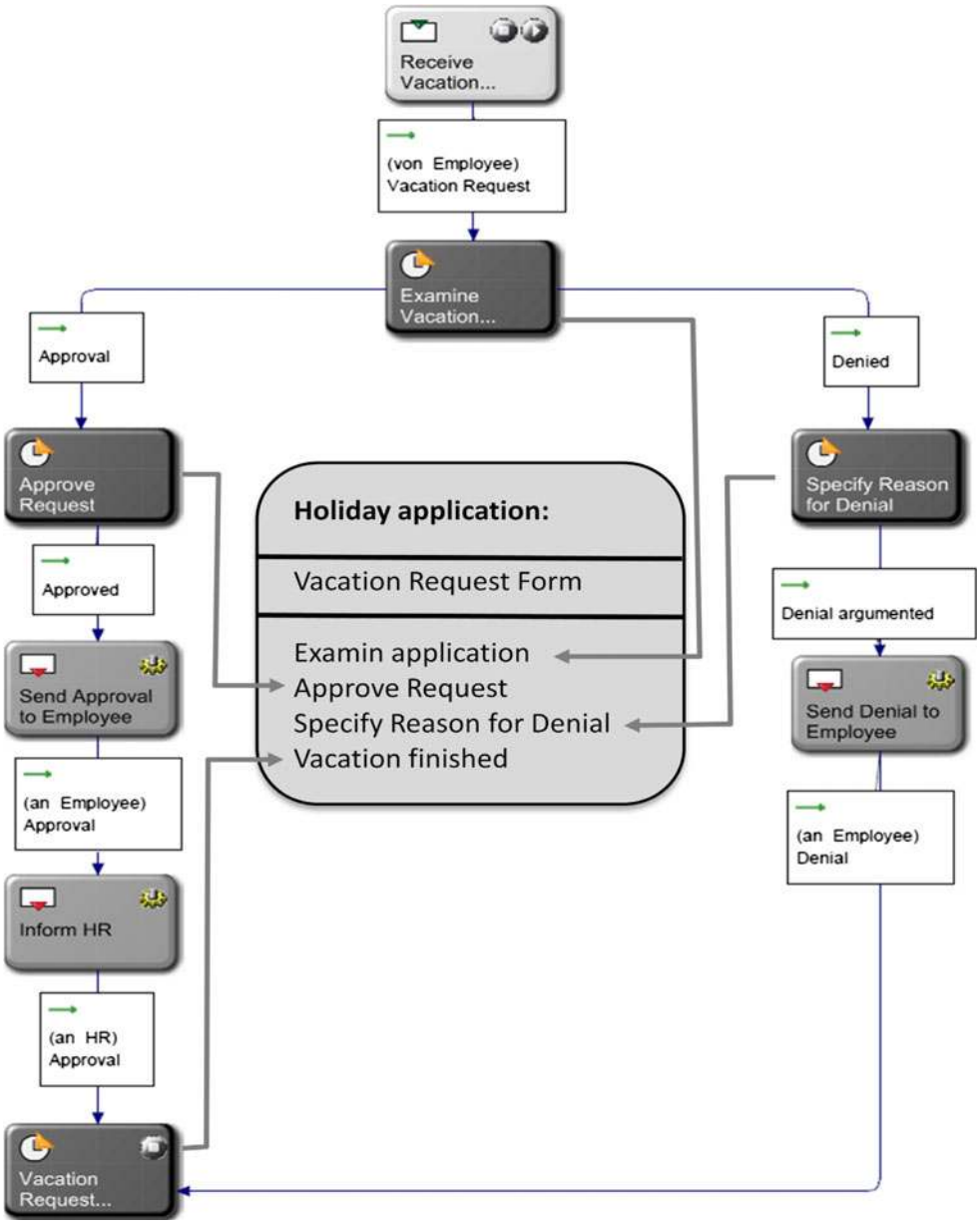


Fig. 14 Subject with predicates and object

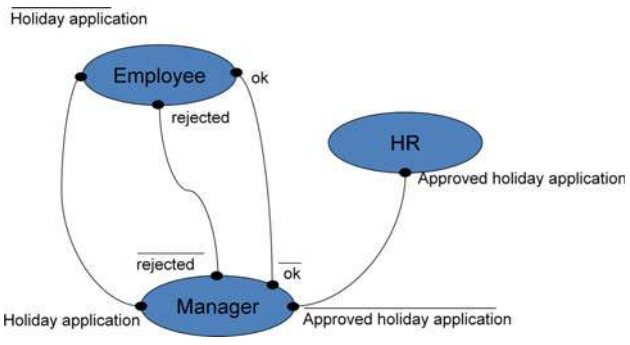


Fig. 15 CCS processes for a holiday application

Consequently, flow diagrams based on algorithmic entities have been designed initially, and later on, extended to represent event-driven activity chains.

4.1.1 Flow diagrams

One of the first models for algorithmic task descriptions has been a flowchart or program structure plans. Flowcharts describe a sequence of operations or activities to solve a task. As such, they can be used to capture work procedures or business processes like the holiday application. In case flowcharts are used to describe arithmetic operations, the role initiating the process is given implicitly. The machine or a human user triggers the activities

Fig. 16 CCS description of the holiday application process

$$\text{Employee} = \overline{\text{Application\_for\_vacation}}.(\text{denied} + \text{accepted}).\text{NIL}$$

$$\text{Manager} = \text{Application\_for\_vacation}.(\overline{\text{denied}} + \overline{\text{accepted}}).\overline{\text{Approved\_application}}.\text{NIL}$$

$$\text{HR} = \text{Approved\_application}.\text{NIL}$$

$$\text{Vacation\_process} = \text{Employee}!\text{Manager}!\text{HR}$$

Employee = Holiday application → (rejected → SKIP ! approved → SKIP)
Manager = Holiday application → (rejected → SKIP ! approved → approval → SKIP)
HR = Approval → SKIP
Holiday = Employee !! Manager !! HR

Fig. 17 Holiday application process in CSP

denoted by the algorithmic elements. Such standard subjects are not mentioned explicitly. In addition, the data requited for executing a flowchart are given rudimentary. Hence, flow charts do not provide complete sentence semantics. They require natural language supplements to describe relevant subjects and objects.

Figure 21 shows as sample flow chart, left only the actions and, partially, the affected objects are included, right subjects as well as objects are captured through activity specifications (rectangles in the figure). However, both aspects are represented linguistically. Enriched flow charts allow subjects and objects to be included using dedicated symbols. Figure 22 shows a chart representing the same content as Fig. 21, however, providing subjects indirectly as inputs and objects using dedicated symbols.

4.1.2 Event-driven process chains

Event-driven process chains (EPCs) are based on control flow diagrams for the representation of business processes. Figure 23 shows the process of the holiday application as an EPC.

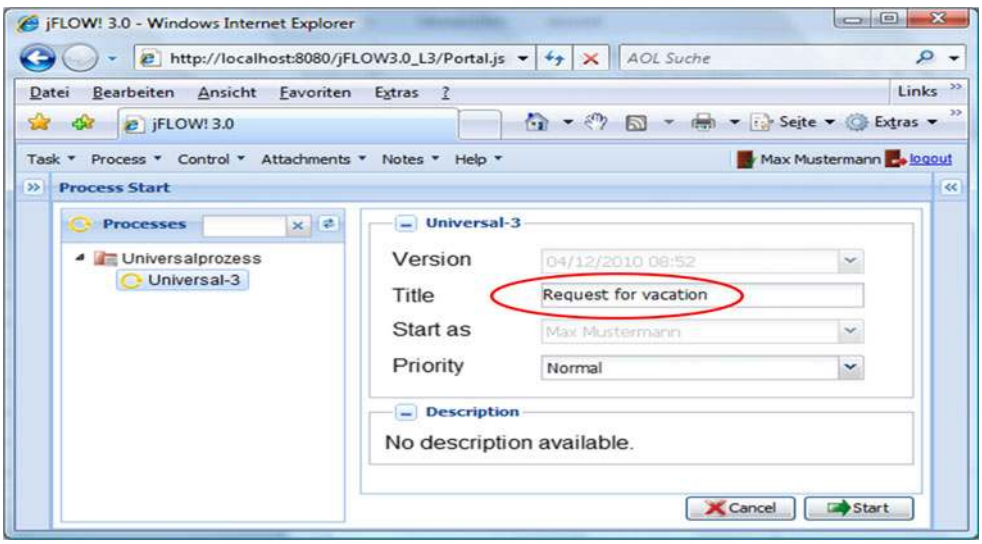
The rectangles represent activities of a process. For clarification, they are inscribed using natural language descriptions. Those inscriptions might contain the addressed objects. Before taking action events (hexagons) have to be specified. They trigger the execution of an activity (also termed function) referring to the previously executed one. Connectors allow different control flows depending on the results of function executions, leading to different events. The function 'application check' can either lead to the event 'Rejected' or 'Approved' using an exclusive or connector (XOR). Beside XOR, OR and AND can be used (cf. [46]).

Enriched EPCs (eEPCs) use various elements to represent business organizations, such as data or goals. These extensions are widely accepted for information system development and correspond to subjects and objects. Functional roles or units of an organization are considered to be starting point for modelling tasks, comprising an activity and concerned data (representing the results). Figure 24 shows an eEPC for the holiday application process.

4.1.3 Petri nets

The most formalized way to use flow diagrams is Petri nets. Their design has been oriented towards the dynamics of systems, however, focusing on activities. Hence, they are predicate-oriented. In contrast to control flow charts, they allow specifying parallel activities. In order to capture data

Fig. 18 Creating a process instance entitled 'request for vacation'



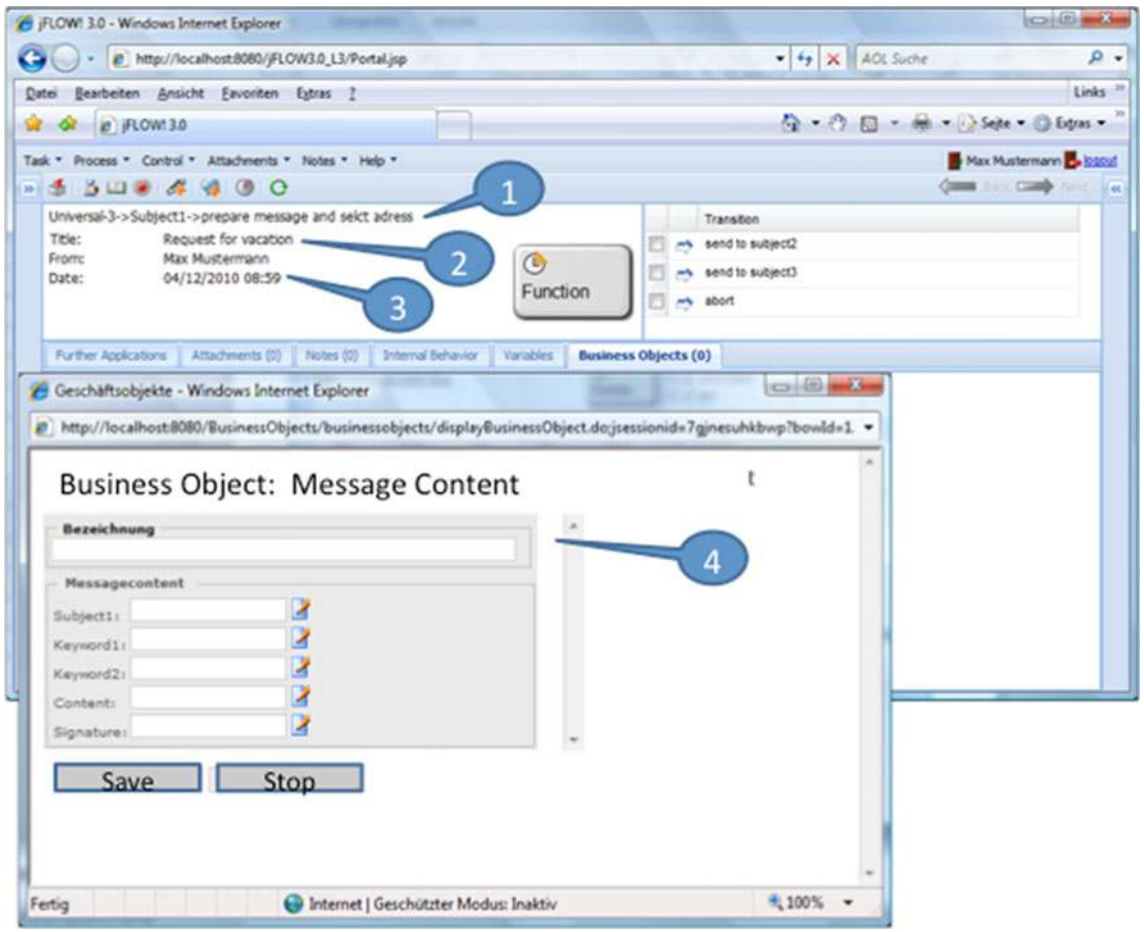


Fig. 19 User interface of the workflow system in state prepares message and selects the person(s) to be addressed

aspects, Petri nets with attributes have been developed. So far, subjects have not been included into Petri nets. Figure 25 shows a Petri net for the holiday application process.

A Petri net consists of alternating states and transitions along a time line, and one or more tokens (markers) where to start ('Employee needs holidays' in Fig. 25). Transitions are interpreted as activities, and states as events to fire a transition. A transition can fire as soon as a (single) token is provided as input. When firing, each subsequent state receives a token. In Fig. 25, passing the token corresponds to the receipt of the application for holidays by the employee's manager. The result is displayed in Fig. 26.

Subsequently, either the transition 'Manager rejects holiday application' or 'Manager approves holiday application' fires. The Petri net is not deterministic anymore. In case the transition 'Manager approves holiday application' fires, the token the process moves to the state 'Approved holiday application', involving both the Human Resources Department (updating holiday entries) and the employee applying for holidays (see Fig. 27).

Moving the sequence of activities to the centre of modelling, subjects and objects have to be added by respective natural language terms. In the sample case, proper modelling has been achieved by the proper naming of states and transitions. However, most of Petri net approaches deal with concurrent events and situations explicitly, compared to flowcharts.

4.2 Objects or targets of activities

Shifting from arithmetic processing to business data modelling, the structure of data has become more and more essential, leading to dedicated fields, such as content management [31]. As such, the object of sentence semantics becomes an essential aspect of modelling. The modelling approaches to that respect focus on the goal or the results of activities, as the subsequent review of Entity-Relationship Models and Relational Data Models reveals.

4.2.1 Entity-relationship model

The entity-relationship model (ER-model or ERM) serves as a semantic container for data, i.e., elements and relations, stemming from observing human reality. Most

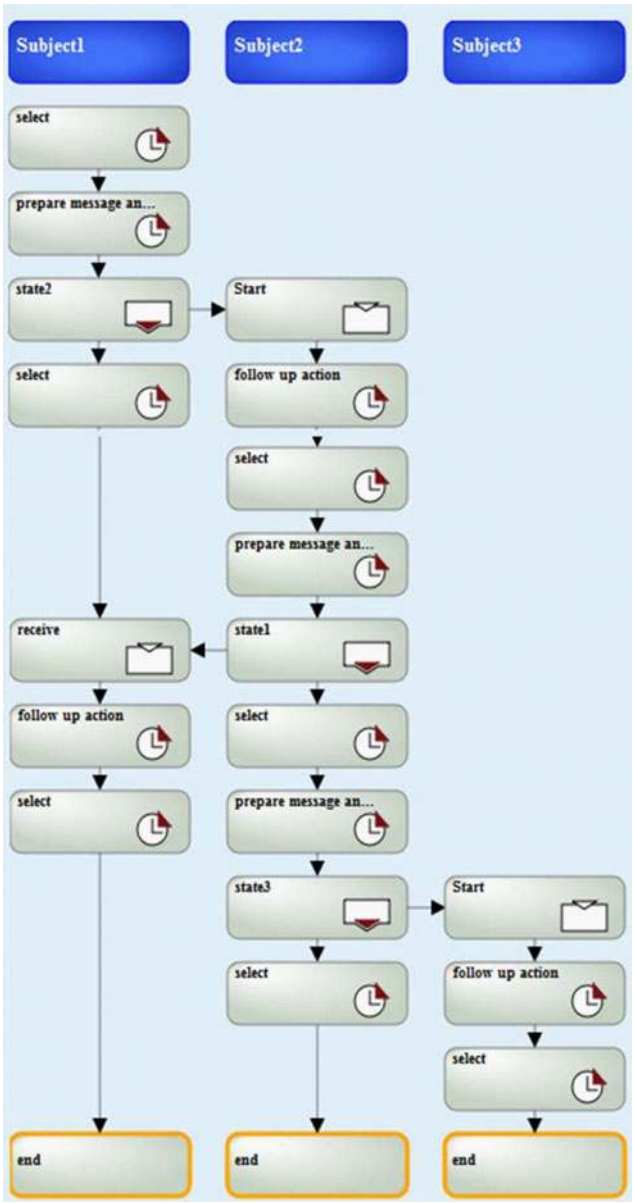


Fig. 20 Execution path of the 3-party vacation application

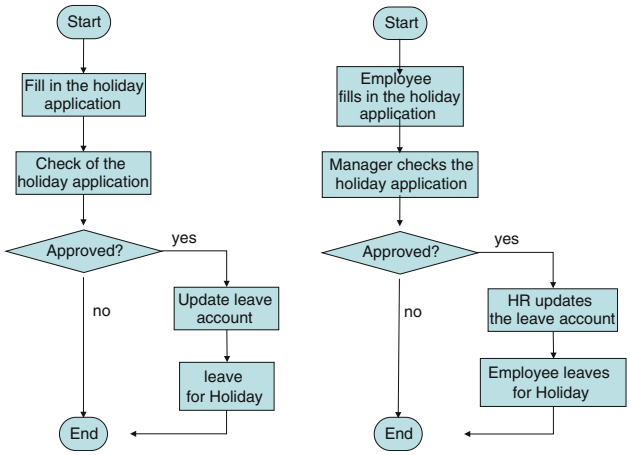


Fig. 21 Sample flow diagram

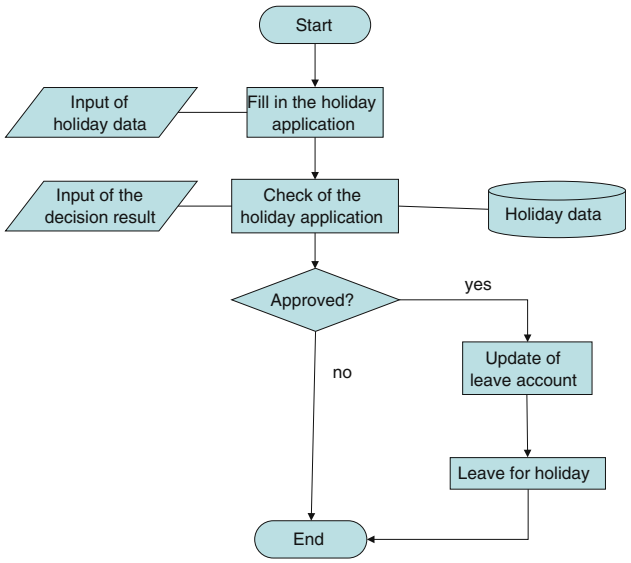


Fig. 22 Flow diagram with explicit data (object) representation

ER-models consist of graphic symbols and a description of the used elements. There exist a variety of diagrammatic notations to present ERM contents. The semantics is specified through categories of data elements and their inter-relationships. When specifying an ERM, concrete entities and relations are used. Entities represent objects of the observed reality, being either material or abstract, such as employee ‘Mark’, manager ‘Max’. Semantic relationships denote relations between two objects, such as ‘employee Mark is assigned to manager Max’.

The model is composed of entity types and relationships types exclusively. An entity type is a classification of entities of the same kind, e.g., ‘employee’, ‘manager’, whereas a relationship type is a classification of relations of the same kind, such as ‘is assigned to’. The semantics of the relationship categories between entity types is represented as text string along the link between entity types. The modeller sets it. Figure 28 shows the ERM of the holiday application process. Each employee is assigned to one manager. Each manager is responsible of one to N employees. Each employee might apply for holidays. Each holiday application contains a specific date denoting the start of holidays and a specific date denoting the end of holidays. A manager might have to check several (0 to M) holiday applications.

As ERMs focus on objects, predicates and subjects are only indirectly addressed, by naming the relationship. In case a predicate is used for a relationship, a sentence according to natural language semantics can be built. However, modellers do not have to use predicates for naming relationships. It depends on their conventions

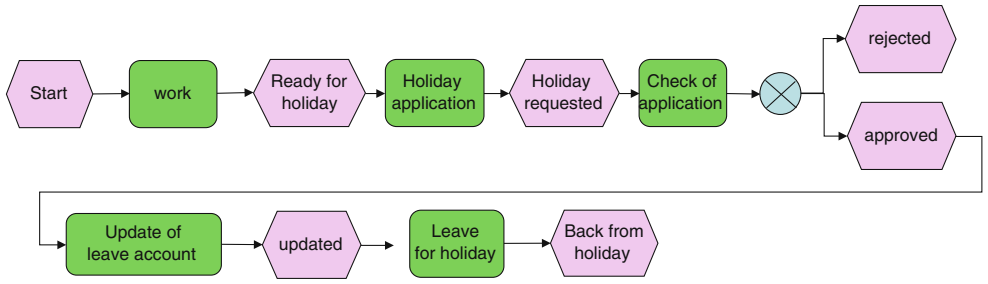


Fig. 23 Application for holidays as event-driven process chain

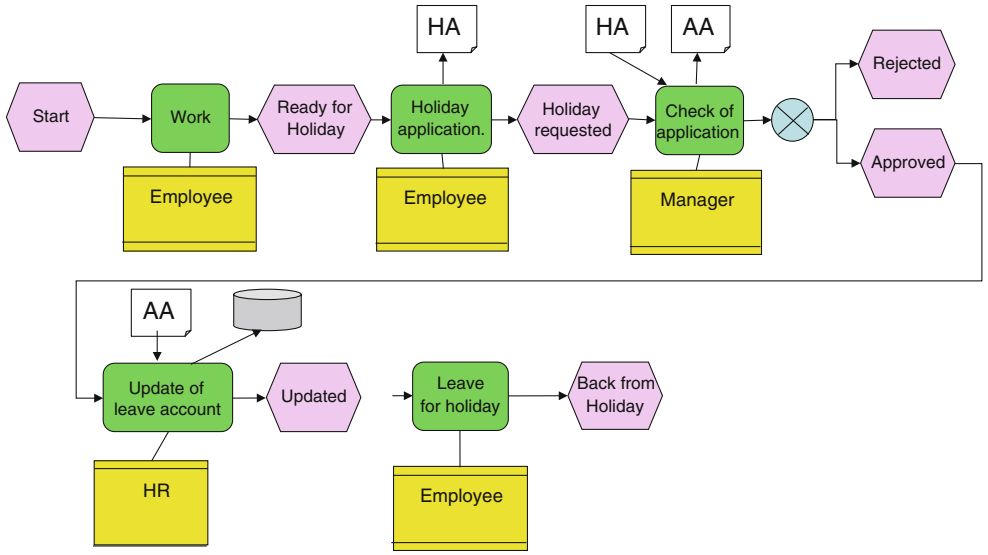


Fig. 24 eEPCs containing subject, predicate and object

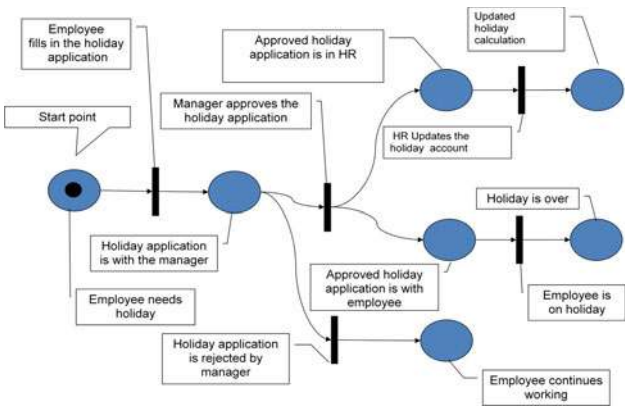


Fig. 25 Applying for holidays in Petri net notation including start marker

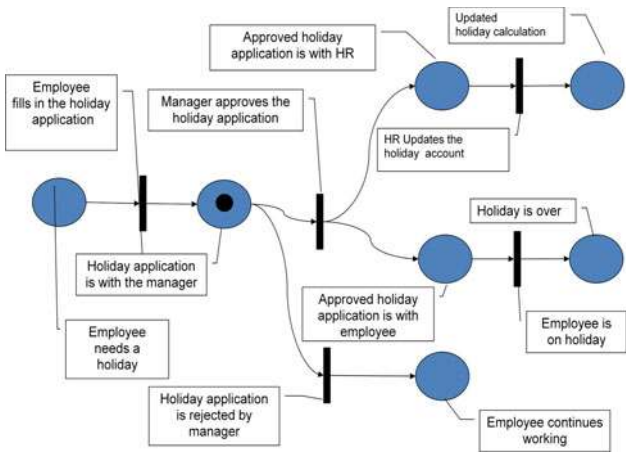


Fig. 26 Firing 'employee applies for holidays'

besides the objectives of modelling whether predicates become part of ERMs. As ERMs also do not provide flow concepts, it is not evident when activities can or have to be executed (predicate). Finally, only conclusive evidence of subjects can be produced, i.e., who is triggering an activity, in case predicates are used for denoting relationships.

4.2.2 Relational data model

In relational data models, only data objects are of interest, like in ERMs. Subjects and predicates are accidental factors (supplementary elements). There is only one type of structural elements for modelling, namely relations. They

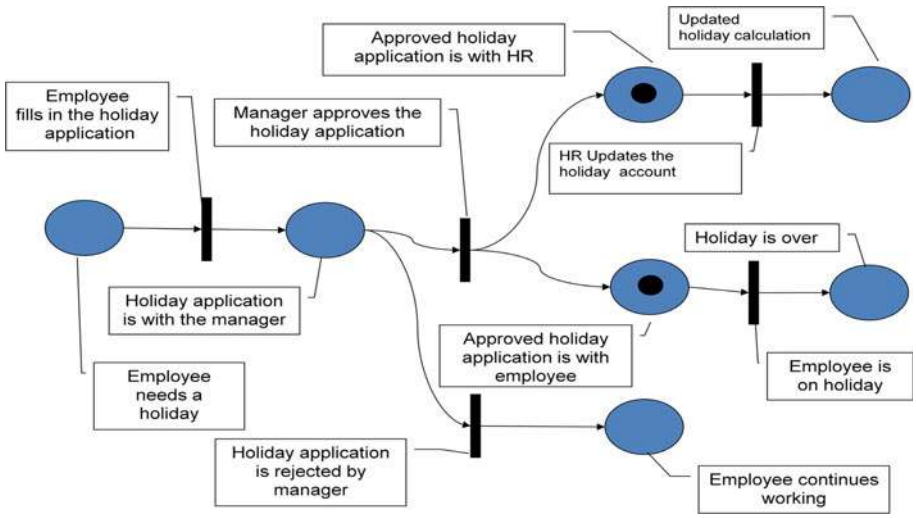


Fig. 27 Tokens after firing 'manager approves holiday application'

are expressed using tables. Entries form data records, with each cell corresponding to a data field entry. A data model mostly consists of several tables. Relationships might exist between different tables containing identical content. Data records are accessed via field entries.

Figure 29 shows a data model for the holiday application process. The update of the time account is not part of the figure due to space limits. The data model consists of three tables: employee, manager and holiday applications. The table Manager contains all managers; the table Employee contains all employees with a reference to their manager (in the column V–No). The table Holiday applications contain all filed holiday applications. The column MA-No. of the table Holiday applications contains a reference to the employee who has filed a certain holiday application (Fig. 29).

The access to relational data models is achieved by logical, set-theoretical queries defined by users (subjects). A traditional relational data model does not contain its users (subjects). The query language, such as SQL, actually contains all possible predicates and is triggered by the users.

In the holiday application example, the manager Meir Max (a user or subject) asks for employees by formulating a suitable query (predicate) addressing the table Employee (objects). The employees assigned to Meir Max are those table elements in Employee containing in column V–No. 1. In a next step, all holiday applications have to be identified,

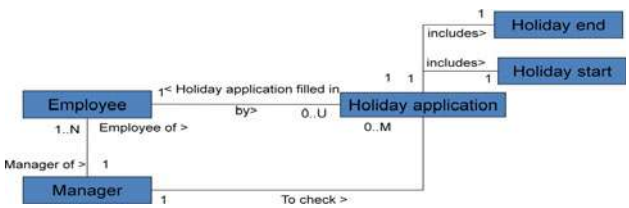


Fig. 28 ERM of the holiday application process

by finding in the column MA- No. denoting an employee of Meir Max (by number). Then, Meir Max has received all holiday applications of his employees and might proceed with his work.

Relational data models have been developed according to implementation capabilities of data engineering technologies. They can be more or less implemented directly by relational data base management systems. In that case, an ERM is used as a modelling language, and the relational model represents program elements. However, in both modelling approaches, subjects are considered to be less important than objects or predicates. Query languages provide predicates, which ERM lacks completely.

4.3 Predicates and objects, or activities and goals

On the one hand, for approaches focusing on predicates, problems with respect to implementation are evident, since object aspects are not covered sufficiently. On the other hand, approaches focusing on objects are dealing with predicates indirectly, as the query language covers them through its use. Moreover, these approaches do not support the specification of control flows, i.e., sequences of predicates.

It seems reasonable to develop modelling concepts, which consider activity- and data aspects in a mutually tuned way, containing predicates and objects. This approach allows setting up complete sentences according to the standard semantics, in particular passive sentences. The latter are used in natural languages as soon as the subject plays a minor role. In case it does not matter who is applying for holidays, the passive description of the holiday application process looks as follows:

- The holiday application data are filled into the application form.

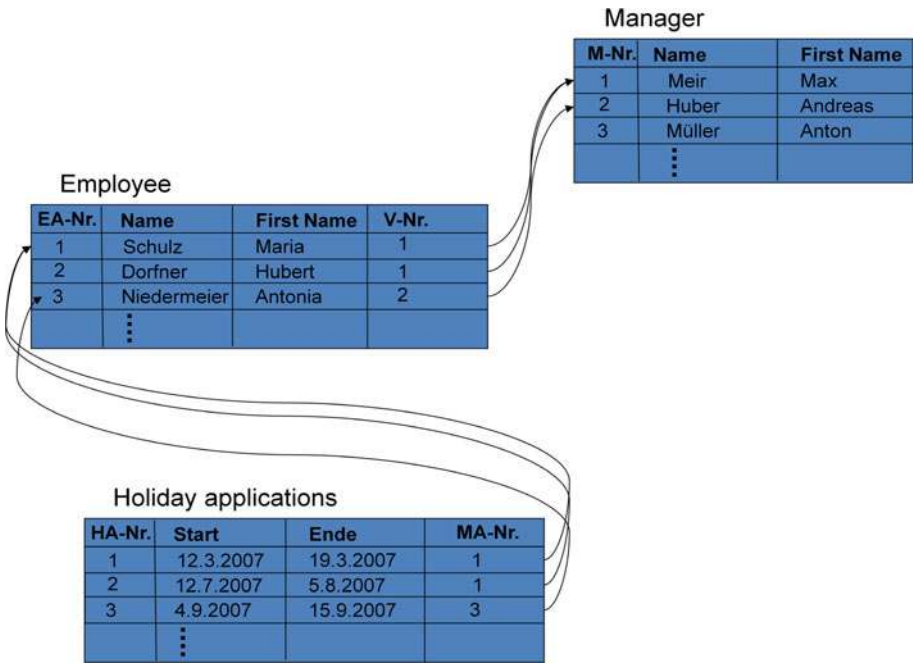


Fig. 29 Relational data model supporting a holiday application procedure

- The holiday application is checked.
- The decision upon approval is documented and communicated.
- The days-off sheet is updated.

Below two integrative approaches are briefly discussed, namely dataflow diagrams as proposed by de Marco [8] and object-oriented modelling.

4.3.1 Data flow diagrams/structured analysis [8]

Data flow diagrams represent the flow of data between functions, data repositories and external components or parties to the system under consideration. Tom de Marco's Structured Analysis (SA) technique allows setting up models using data flow diagrams. In data flow diagrams (DFD), the following graphical symbols are used for the following elements:

- *External interface* (external partner, participant, terminator). External interfaces are displayed as rectangles. They represent relations of the observed system to its environment. They send or receive data, but do not process them. External interfaces trigger the system by providing inputs. They can be considered as subjects to a certain extent.
- *Function* (process, task, function). Circles or ovals present functions. They represent the processing of input data to output data. As such, they represent the algorithms required to process the data. The functions correspond in sentence semantics to predicates. Higher order or complex predicates are further refined by the predicates of a control flow diagram.
- *Data memory* (memory, store). Data memories are presented through parallel lines. They form a repository for data, providing time of generation and use. They can be considered as special functions for storing data.
- *Data flow* (flow of information). The data flow is displayed as arrows between functions or data stores. The arrows are named according to the semantics of the transmitted data. A data dictionary contains the structure of the information used in the DFD. The definition of the structure is provided in Backus Naur Form. However, an ERM could be also used for representation. The data correspond to the objects in the standard sentence semantics.
- *Context diagram*. Fig. 30 shows the context diagram when processing a holiday application. External interfaces are identified, and the system under development is represented as a function. The context diagram shows that the application data are received from an external interface, and the result of processing is delivered to that interface. In this example, the external interface can be considered as a subject. However, the manager has not been modelled explicitly, since being inherent to this system. In case the manager's part and the update of the holiday sheet would be moved to external systems, an empty model would remain, since all activities occur externally.

Figure 31 shows a refinement of the process. It shows the data flow between the functions and data repositories. It

is important to note that the dataflow does not imply any flow of control (which could be interpreted easily).

Although data flow diagram has been developed quite a while ago (in the 1970s), they cover predicates and objects from the standard sentence semantics. Subjects can be introduced providing auxiliary constructions, which might lead to misinterpretations. Data flow diagrams are not used any more. They have been further developed to object-oriented approaches.

4.3.2 Object-oriented modelling

The main idea of object-oriented programming and application design is to couple functions to the concerned data and to provide an encapsulated structure and external interface. Functions together with data form an object in object-oriented modelling. The data of an object can be accessed only with respective methods. Classes capture similar properties of objects. Based on objects (or classes), hierarchies can be specified to represent complex setting. Object-oriented models use dedicated constructs for representation and operations for processing, such as inheritance, polymorphism, aggregation, associations, etc. (see, e.g., www.omg.org for the Unified Modelling Language, UML). Object-oriented modelling can be considered as one of the de-facto standards for information system development. They allow for implementation-independent design representation as well as for detailed design and implementation. Its major advantage is the encapsulation of structure (properties, attributes) and behaviour items (methods, functions), since it facilitates modelling processes in that way.

Object-oriented modelling captures predicates and objects according to standard sentence semantics. An object consists of data and functions. The functions of the object correspond to predicates, while the data correspond to the object in the standard sentence semantics.

Figure 32 depicts the class Holiday Application containing the data start of holidays, end of holidays and the result of decision-making. It also captures the functions filling in the corresponding form, checking the application and documenting the decision. If the application is approved, the holiday sheet is updated, calculating the holidays.

The Holiday Application class allows formulating incomplete sentences, such as ‘fill in holiday application form’ or ‘check holiday application for holidays’. In order to create complete sentences, some original object-oriented methods provide the capability to insert names for subjects in natural language terms. Today, Use Case diagrams, as provided by UML, enable the specification of subjects. For instance, UML provides 13 different categories of diagrams, one of them being the Use Case diagram (see below).

4.4 Subject, predicate and object, or complete sentences

Use Case Diagrams (Use Cases) follow the standard sentence semantics, as the comprise subject, predicate and object. They allow describing a system’s use from the user perspective. A Use Case indicates which user (actor = subject) performs a certain action (predicate) when using a system. A Use Case describes recognizable behaviour by external parties of an observed element (system, class, etc.). It encapsulates a closed collection of actions that are performed in a certain sequence. A use case hides the classes and operations involved for performing actions. Its description is considered complete once the entire behaviour has been specified, either using behaviour modelling constructs of UML or natural language descriptions.

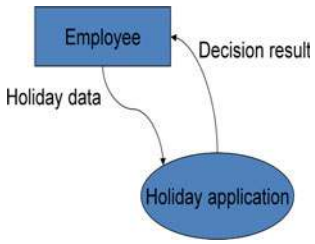


Fig. 30 Context diagram for holiday applications

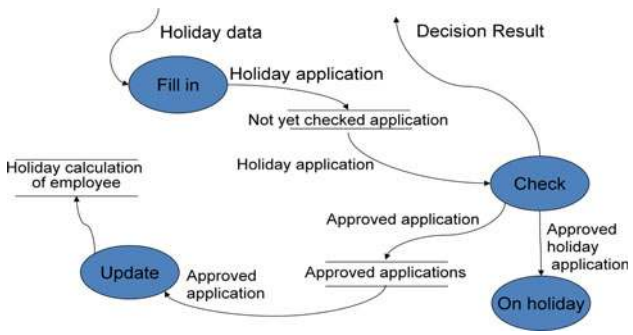


Fig. 31 The holiday application process as DFD

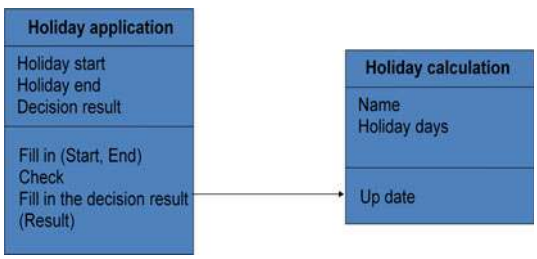


Fig. 32 Classes involved in holiday applications

Actors are considered in UML as dedicated classes having dedicated properties. When using UML, only the sequence of activities between actors and the system is specified. Hence, actors cannot be considered as active components, rather as starting point of activities. Figure 33 shows the Use Case diagram for the holiday application process. The sequence for filing an application for holidays is given as follows: fill in start of holidays, fill in end of holidays, ask manager to check application. The other Use Cases can be described in a similar way. For complex sequences, UML provides activity diagrams, which combine elements of data flow diagrams, Petri's nets, flow charts and others. However, when using several activity diagrams, events and signal can only be exchanged in a restricted form. Consequently, for the sample use case, the mutual exchange of information between the specified diagrams is very limited, as waiting for approval or rejection requires additional modelling effort.

Although UML provides inadequate means for comprehensive modelling, its diagram types allow forming complete sentences using standard sentence grammar. In UML, actors are not part of the model. Consequently, their behaviour cannot be detailed, in particular with respect to communication. As such, actors do not appear in the remaining diagram types of UML, except in Time Sequence diagrams. However, actors are essential in business process specifications, in particular for intra-organizational processes. Standard UML lacks proper elements for complete sentence representations.

5 Creating empirical evidence

Neubauer et al. [33] have challenged the stakeholder-centeredness of S-BPM. The study intended to identify factors that cause confusion, and thus, cognitive work load in the course of modelling, preventing the alignment of individual perspectives of stakeholders when describing and specifying their work procedures. At the centre of interest were notational constructs that allow generating adequate business process models from a stakeholder perspective, by studying how those constructs are utilized in the course of modelling.

As the study has been intended to find out human-centred constructs and procedures for meaningful modelling, an open language format provided by SeeMe [23] has been used. In addition, dynamic switching between particular elements or views on models, such as processes, functions, tasks, has been enabled avoiding any bias towards predefined structures when stakeholders were modelling.

In the study, young adult consumers were asked in 52 individual modelling sessions to contribute to a common scenario, namely purchasing a car. According to the theoretical background of SeeMe, namely communication theory, stakeholders were expected not only to communicate apparent details regarding purchasing a car, but also essential hidden or primarily intangible assets. The latter are required for adequate representations, as they provide valuable input for organizational development and corresponding technological artefacts.

After a short introduction to the elementary SeeMe modelling constructs, procedural instructions have been given to the participants. They were asked to (1) collect all relevant basic elements based on the text description of the case, before (2) identifying relations between the selected basic elements relevant them and (3) identifying situations where connectors are needed according to your understanding. They should capture as many aspects of the textual process description as possible in a process model.

Modelling styles and patterns have been investigated. They direct the modelling process. Besides flow orientation, the orientation towards natural language structures

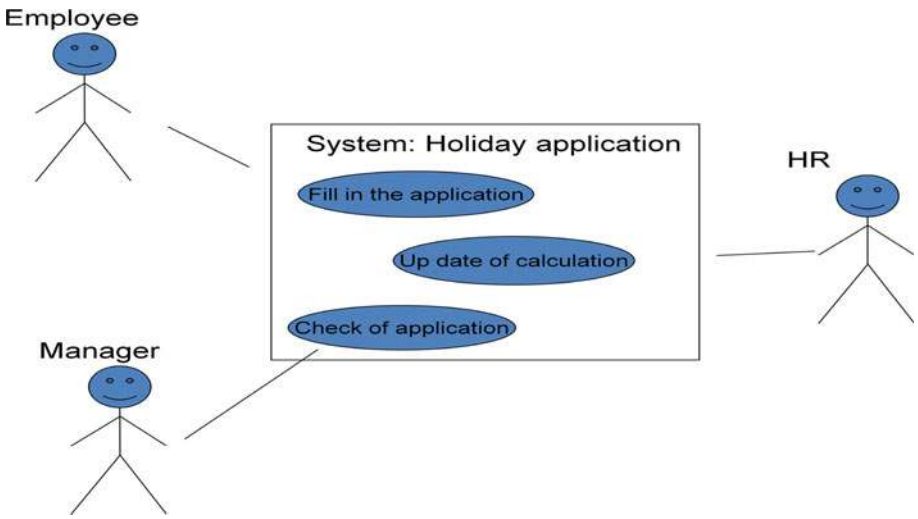


Fig. 33 Use case diagram for a holiday application process

was of interest, since stakeholders initially tend to use natural language to describe their view upon business processes. With respect to natural language orientation, it has been evaluated whether stakeholders have utilized SeeMe constructs to express sentence semantics, such as subject–predicate–object.

In the study, all of the stakeholders have used the following *constructs*: role, activity, entity, associations between elements and vagueness/incompleteness placeholders. Besides nesting representations, connectors have primarily been used to control the flow of activities, e.g., representing optional activities or parallel activities within a process.

With respect to the *modelling process*, the majority of stakeholders have modelled the process flow from top to bottom, i.e., activities have been arranged from top to bottom. Roles and entities associated with specific activities were placed next to the activities. The modelling results differ significantly in the level of detail, ranging from vague and high-level descriptions to concrete process steps. However, the majority of stakeholders have depicted the concrete process description and have not generated an abstract picture of the process.

The following *model categories* could be identified: flow- and natural language-oriented, and combined models. Most of the specified processes reflected a clear flow orientation. They also contained connectors, mainly set between activities to control the process flow. Natural language-oriented models covered models based on natural language structures (subject–predicate–object). Overall, ten per cent of the given models have been identified as language-oriented ones. Interestingly, in this category, connectors have been set mainly between entities, only one language-oriented model contained connectors between activities. Relations have been used to form sentences in natural language. Similar to the flow-oriented models, most models show an arrangement of activities from top to bottom, with roles and entities aside.

Combined models (21% of the models) contained aspects from flow-, language- and/or data flow orientation. They partially reflected inconsistent modelling of control flow between activities: roles triggered activities, whereas entities represented the data flow between activities. Connectors have mainly been used to link entities, besides interrelating roles, and connectors to activities.

All 52 *specified processes* contained standard sentence semantics for purchasing a car, comprising 'Who' does 'What' 'Using what kind of data or element'. All participants have used relations, embodiment with respect to structural aspects, and connectors in the majority of cases. Modelling constructs have been used in a natural language style. In these cases, the reflection and reproduction of already specified information could be achieved in complete sentences.

For S-BPM, another major result is the strong orientation towards flow, as it lays ground to think in behavioural terms when describing task accomplishment. It is a prerequisite for communication-based process execution enabling stakeholders to complete process specifications for direct execution. Finally, the observation that the majority of stakeholders were able to provide concrete process models of the given scenario rather than abstractions facilitates experiencing concrete task executions.

6 Conclusive summary

Organizations are increasingly forced to restructure their business processes in a flexible way during operation. It requires stakeholders to take responsibility for organizational developments. Traditionally, only few members are skilled in specifying and developing business processes. Hence, it is proposed to support them with natural language constructs (subject, predicate, object) and e-mail-like communication patterns between actors (subjects) when describing business processes. In this way, individual members of an organization are able to contribute to coherent and intelligible process specifications, as the resulting specifications can be processed without transformation.

The introduced subject-oriented modelling scheme recognizes actors as starting point for modelling, leading to a tripartite approach, as it takes into account standard sentence semantics (subject, predicate, object). Using subjects, stakeholders avoid conveying information reduced either to content or functional business logic. As the table according to Schmidt et al. [41] reveals, a shift from data- or function-oriented to natural language-based modelling facilitates the intelligibility (see 'Explanation'). It also ensures coherence, as both, the flow of control, and the addressed data can be kept throughout the modelling and execution process. Consequently, stakeholders and developers should experience less misunderstanding and conflicts, as, e.g., experienced by the widespread Use Case Diagrams (in UML) in industrial practice. This benefit becomes essential for networked organizations striving for interoperability, as social interaction, cooperation and collaboration aspects have to be reflected by models. Collaboration across organizational boundaries demands subject-oriented representations, since actor-specific communication between (process) partners is crucial and has to be part of any process model and tool support.

Approach	Subject–predicate–object	Orientation	Explanation
Natural language	Yes–yes–yes	… as I say …	Who does what?
Control flow diagrams	No–yes–no	Function	What has to be done?
Event-driven process chains	No–yes–no	Function	What has to be done?
Extended event-driven process chains	Partially–yes–partially	Function	Who does what?
Entity-relationship model (ERM)	No–no–yes	Data	Which data are handled?
Unified modelling language	Partially (only in use cases)–yes–yes	Object (data)	What happens?
Calculus of communicating systems	Yes–partially (only in terms of responsibility)–no	Subject	Who runs what?
Subject-oriented modelling	Yes–yes–yes	Subject	Who does what?

The tool (see www.metasonic.de) allows generating executable application programs, by implementing complete standard sentence semantics. The next research objective is to examine how knowledge can be processed and managed using standard sentence semantics based on subjects. Such a structure lays ground for organizational learning processes, both at the single and the double loop. Business process specifications do not only encode operational elements (single loop), but also values and cognitive drivers of learning processes (double loop) (cf. [17]). As learning involves both, organizational development and learning support have to encounter this dichotomy. In this context, simulation is considered helpful to experience realistic process scenarios before changes are implemented (cf. [21]). The distribution and assignments of various resources might trigger different actor and business behaviour. Its impact needs to be assessed prior to implementation.

References

1. Acosta, A., Leon, Y.J., Conrad, C.R., Malave, C.O.: Global Engineering: Design, Decision Making and Communication. CRC Press, London (2009)
2. Agarwal, A., Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Plösser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M: WS-BPEL Extension for People—BPEL4People, Version 1.0, June 2007
3. Bach, V.: Role specific portal design (in German) duv, Wiesbaden (2006)
4. Bandara, W., Gable, G.G., Rosemann, M.: Factors and measures of business process models: model building through a multiple case study. Eur J Inf Syst **14**, 347–360 (2005)
5. Brown, J., Duguid, P.: Organizing knowledge. Reflections 1(2). Society of Organizational Learning (1999a)
6. Brown, J., Duguid, P.: The Social Life of Information. Harvard Business School Press, Boston (1999)
7. Chakraborthy, D.: Extending the reach of business processes. IEEE Comput **37**(4), 78–80 (2004)
8. De Marco, T.: Structured Analysis and Systems Specification. Prentice Hall, Englewood Cliffs, New Jersey (1978)
9. Denert, E.: Software Engineering. Springer, Berlin (1991)
10. Eapen, G.: Flexible Companies for the Uncertain World. CRC Press, London (2009)
11. Ehrig, M.: Ontology Alignment. Bridging the Semantic Gap. Springer, Berlin (2007)
12. Fleischmann, A., Chin, S.T., Effelsberg, W.: Specification and implementation of an ISO session layer. IBM Syst J **25**(3), 255–275 (1987)
13. Fleischmann, A.: Distributed systems-software design and implementation. Springer, Berlin (1994)
14. Fleischmann, A.: Subjektorientiertes Geschäftsprozessmanagement, White Paper@ www.jcom1.com: jCOM1 AG, Rohrbach (2007)
15. Fleischmann, A., Lippe, S., Meyer, N., Stary, C.: Coherent task modelling and execution based on subject-oriented representations. In: Proceedings TAMODIA'09, Task Modelling and Diagrams for User Interface Design. Brussels, Springer, September 2009
16. González, A., España, S., Pastor, O.: Towards a communicational perspective for enterprise information systems modelling. In: The Practice of Enterprise Modeling, Lecture Notes in Business Information Processing, Vol. 15, Part 2, pp. 62–76 (2009)
17. Groff, T.R., Jones, T.P.: Introduction to Knowledge Management: Knowledge Management in Business. Butterworth/Heinemann, Amsterdam (2003)
18. Haeckel, S.S.: Adaptive Enterprise: Creating and Leading Sense-AND-Respond Organizations. Harvard Business School Press, Cambridge, MA (1999)
19. Harel, D., Rumpe, B.: Meaningful modelling: what's the semantics of "semantics"? IEEE Comput. **37**(10), 64–72 (2004)
20. Havey, M.: Essential Business Process Modelling. O'Reilly, New York (2005)
21. Heftberger, S., Stary, C.: Partizipatives Organisationales Lernen–Ein Prozessorientierter Ansatz. DUV, Wiesbaden (2004)
22. Herrmann, T., Hoffmann, M., Kunau, G., Loser, K.U.: A modelling method for the development of groupware applications as socio-technical systems. Behav. Inf. Technol. **23**(2), 119–135 (2004)
23. Herrmann, T.: SeeMe in a nutshell—the semi-structured, socio-technical modeling method. http://www.imtm-iaw.rub.de/imperia/md/content/seeme/seeme_in_a_nutshell.pdf (2006)
24. Hoare, C.: Communicating sequential processes, CACM (1978)
25. Hoare, C.: Communicating sequential processes, Prentice Hall (1984)
26. Holmes, T., Vasko, M., Dustdar, S.: VieBOP: extending BPEL engines with BPEL4People. Parallel, distributed and networked-based processing. In: Proceedings of the 16th Euromicro Conference on PDP, pp. 547–555 (2008)
27. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: WS-BPEL extension for people—BPEL4People, A Joint White Paper by IBM and SAP, July 2005

28. Krogstie, J., Sindre, G., Jorgensen, H.: Process models representing knowledge for action: a revised quality framework. Eur. J. Inf. Syst. **15**, 91–102 (2006)
29. Lankhorst, M., et al.: Enterprise Architecture at Work. Modelling, Communication, and Analysis. Springer, Berlin (2005)
30. Lewis, M., Young, B., Mathiassen, L., Rai, A., Welke, R.: Business process innovation based on stakeholder perceptions. Inf. Knowl. Syst. Manag. **6**, 7–17 (2007)
31. Lohr, J., Deppe, A.: Der CMS-Guide. Content-Management-Systeme: Erfolgsfaktoren, Geschäftsmodelle, Produktübersicht. Vieweg, Wiesbaden (2001)
32. Milner, R.: Calculus of Communicating Systems. Springer, Berlin (1980)
33. Neubauer, M.; Oppl, S.; Stary, C.: Towards intuitive modelling of business processes: prospects for flow- and natural language-orientation. In: Proceedings TAMODIA'09, Task Modelling and Diagrams for User Interface Design. Brussels, Springer, September 2009
34. Pinker, St.: The stuff of thought: Language as a window into human nature. Allen Lane, London (2007)
35. Rittgen, G.: A language-mapping approach to action-oriented development of information systems. Eur. J. Inf. Syst. **15**, 70–81 (2006)
36. Rouse, W.B. (ed.): Enterprise transformation: understanding and enabling fundamental change. Wiley, Hoboken, NJ (2006)
37. Rumpfhuber, M.; Fichtenbauer, C.; Stary, C.: Sprachgerechte unternehmensnahe Modellierung von ereignisgesteuerten Prozessketten—Zur adäquaten Aus- und Weiterbildung von ModellierInnen. In: Proceedings EPK-2002, Workshop 'Ereignisgesteuerte Prozessketten', GI, Springer Lecture Notes, pp. 109–118 (2002)
38. Sarini, M., Simone, C.: Recursive articulation work in Ariadne: the alignment of meanings. In: Proceedings of COOP 2002, pp. 191–206 (2002)
39. Scheer, A.-W.: ARIS—Modellierungsmethoden, Metamodelle, Anwendungen, 4th edn. Springer, Berlin (2001)
40. Schmidt J.E., Rabanus S., Vilmos A.: Syntax http://www.web.uni-marburg.de/dsa//Direktor/Rabanus/SS2005/Grundlagen.pdf (2005)
41. Schmidt, W., Fleischmann, A., Gilbert, O.: Subjektorientiertes Geschäftsprozessmanagement. In: HMD—Handbuch der Wirtschaftsinformatik (2009)
42. Scholz, M., Holl, A.: Objektorientierung und Poppers Drei-Welten-Modell als Theoriekerne der Wirtschaftsinformatik. In: Schütte, R. et al. (eds.) Universität Essen, Essen (1999)
43. Smith, H., Fingar, P.: Business Process Management: The Third Wave. Meghan-Kiffer, Tampa (2003)
44. Stary, C.: TADEUS: seamless development of task-based and user-oriented interfaces. IEEE Trans. Syst. Man Cybern. A **30**(5): 509–525 (2000)
45. Stephenson, S.V., Sage, A.: Architecting for enterprise resource planning. Inf. Knowl. Syst. Manag. **6**, 81–121 (2007)
46. Strosnider, J.K., Nandi, P., Kumaran, S., Ghosh, S., Arsanjani, A.: Model-driven synthesis of SOA solutions. IBM Syst. J **41**(5), 415–432 (2008)
47. UN/CEFACT: UN/CEFACT core components technical specification, version 3.0. http://75.43.29.149:8080/download/attachements/3801818/Specification_CCTS3p0+2nd+Public+Review+16APR2007.pdf?version=1 (2007)
48. Vernadat, F.B.: Enterprise Modelling and Integration: Principles and Application. Chapman and Hall, London (1996)
49. Vogel, T., Schmidt, A., Lemm, A., Österle, H.: Service and document based interoperability for European eCustoms solutions. J. Theor. Appl. Electron. Comm. Res. **3**(3), 31–37 (2008)
50. Weick, K.E.: The Social Psychology of Organizations. Addison Wesley, Reading, MA (1999)
51. Wieden, W.: Corporate linguistics: a knowledge management approach to language. AAA—Arbeiten aus Anglistik und Amerikanistik **31**(2), 185–207 (2006)
52. Winograd, T.: A language/action perspective on the design of collaborative work. In: Greif, I. (ed.) Computer-Supported Cooperative Work. A Book of Readings, pp. 623–653. Morgan Kaufman, San Mateo, CA (1988)
53. Wood, L.E. (ed.): User Interface Design: Bridging the Gap from User Requirements to Design. CRC Press, Boca Raton, FL (1998)
54. Yankee Group: The Yankee group report: interoperability emerges as new core competency for enterprise architects. http://www.intersystems.com/ensemble/analysts/yankee.pdf (2008)