



IST-FP6-508794

PROTOCURE II

*Integrating formal methods in the development process of
medical guidelines and protocols*

Specific Targeted Research Project
Information Society Technologies

**Validated new model of guideline development
process.**

Due date of deliverable: 31 May 2005

Actual submission date: 20 September 2005

Start date of project: 1 January 2004 Duration: 30 months

Organisation name of lead contractor: Universitat Jaume I

Revision 1

Project co-funded by the European Commissions within the Sixth Framework Programme(2002-2006)		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Addendum to Deliverable D1.3 New model of guideline development process

Juan Carlos Galán, Mar Marcos
Universitat Jaume I

Wolfgang Reif, Michael Balsler, Jonathan Schmitt
Augsburg University

September 8, 2005

Abstract

This document is an *addendum* to Deliverable D1.3, focusing on a deeper analysis of UML and IDEF0 process modelling languages. In this new study we highlight the differences between the languages, and give a better support for preferring IDEF0 over UML as language to model the guideline development process. Moreover, we have reviewed and updated part of the data related to the selection of the process modelling language.

Contents

1	Introduction	1
2	Main issues after the review of D1.3	2
2.1	Issues related to the information of the process modelling languages	2
2.2	Issues related to the justification of IDEF0 as process modelling language	5
3	Justification of IDEF0 as process modelling language	6
3.1	UML	6
3.2	IDEF0	9
3.3	UML VS. IDEF0	12
4	Conclusions	16

1 Introduction

This document complements deliverable D1.3 [35], focusing on a deeper analysis of UML and IDEF0 process modelling languages. In this new study we highlight the differences between the languages, and give a better support for

preferring IDEF0 over UML as language to model the guideline development process. Moreover, we have reviewed and updated part of the data related to the selection of the process modelling language.

Along this document, we comment on some deficiencies detected in D1.3 and, mainly, provide further support which justifies the choice of IDEF0. In section 2 the main two issues pointed by the reviewers are exposed. Then, in section 3 we make a deeper analysis of UML and IDEF0, splitted into subsections in which we explain these process modelling languages in depth and present a comparison between them. Finally, in section 4, the main conclusions are showed.

2 Main issues after the review of D1.3

According to the reviewers, deliverable D1.3 presented some deficiencies. The main problems were:

- Inaccurate information about some of the process modelling languages, for example, EPC, LOTOS, CSP, EAI, EDOC, etc.
- Insufficient support to justify the choice of IDEF0 as process modelling language. The reviewers proposed the use of activity diagrams of UML to model processes as alternative.

Next, a revision of deliverable D1.3 in relation to the above issues is presented. The second issue, which required a deeper study, is exposed and analysed in section 3.

2.1 Issues related to the information of the process modelling languages

In relation to the steps performed to select the process modelling language, the reviewers argue that part of the analysed languages are intended to model software processes. In addition, they point to inaccuracies of the information of some of the languages. A quote from the review report follows:

“Several of the languages considered were not languages for modelling development processes. Languages like LOTOS and CSP, for example, are related to the modelling of concurrent software components, which are also called processes, but in an entirely different sense. In the case of several languages, information was reported not to be found, when it is actually available; some examples are EPC, WPD, LOTOS, and CSP.”

The reason to include both software process modelling languages and business ones in the analysis was to obtain information about all possible alternatives. Accordingly, languages as LOTOS or CSP were included in the primary list of candidates. However, we realised that business process modelling languages were more adapted to the goal of modelling evidence-based guideline development.

To deal with the inaccuracies in the language information, we have reviewed the deliverable and the literature in order to solve the problems. Besides, we

have checked all the languages referred to in the deliverable, reviewing the table *List of Process Modelling Languages preliminary candidates* accordingly. As a result, we have detected some differences between the information of the languages contained in the annexes, and the data mentioned in this table. Concretely:

- CSP. The reference was missing. The correct reference is an article by *Miller and Sufrin* [38]
- EAI. The *References-Current* column should be *YES*
- EDOC. The *References-Current* column should be *YES*
- EPC. The reference was wrong. The correct reference is an article by *Van der Aalst and al.* [43]. According to the correct reference the *References-Current* column should be set to *YES*
- GEM. The reference was wrong. The correct reference is an article by *Conradi and al.* [39]. According to the correct reference the *References-Current* column should be set to *NO*
- LOTOS. The reference was missing. The correct reference is an article by *Yasumoto and al.* [20]. According to the correct reference the *References-Current* column should be set to *NO*
- PROMENADE. The *References-Current* column should be *YES*. Moreover, the *Year/Period* column should be *2002*
- SDL. The *References-Current* column should be *YES*
- SLANG. The *References-Current* column should be *NO*
- UPM. The *References-Current* column should be *YES*

The table *List of Process Modelling Languages preliminary candidates* of the deliverable has been updated to produce a new version (see table 1). Obviously, the introduced changes have been cross-checked with the following steps and tables in the selection process, in order to find out if the outcome of the selection remained the same, which was the case.

Table 1: List of PMLs preliminary candidates (alphabetic order)

No.	Language	Year/Period	References		
			Direct	Main indirect	Current
1	ADELE-TEMPO			[39]	NO
2	ALF	1987-1992	[9]	[39]	N/A
3	AMBER			[13]	N/A
4	APPEL		[15] [16]	N/A	N/A
5	APPL/A		[6]	[39]	N/A
6	ARIS			[27]	YES
7	Articulator			[14] [36]	N/A
8	BAM			[43]	N/A

9	BPEL4WS		[2]	[43]	YES
10	BPML		[3] [32]	[27]	YES
11	Chou-UML		[41]	N/A	NO
12	CIMOSA			[27]	N/A
13	Conversation Builder			[14]	N/A
14	CSP		[38]		YES
15	CSPL		[19]	[22]	YES
16	E3		[24] [25] [30] [29] [28]	[39]	YES
17	EAI			[27]	YES
18	ebXML			[27]	N/A
19	EDOC		[45]		YES
20	EEML			[27]	N/A
21	ENVI12204			[27]	N/A
22	EPC			[43]	YES
23	EPOS		[31]	[39]	NO
24	EVPL		[17]	[22]	YES
25	FUNSOFT			[14]	N/A
26	GEM			[39]	NO
27	GRAI			[27]	YES
28	GRAPPLE			[14]	N/A
29	Hakoniwa			[42]	NO
30	HFSP			[42]	NO
31	IDEF	1994-2004	[21] [5]	[27]	YES
32	IEM			[27]	N/A
33	ITM			[27]	N/A
34	JIL		[1]	[22]	YES
35	LATIN	1995	[10]	[10]	NO
36	LOTOS		[20]	[42]	N/A
37	LSPL		No references	No references	N/A
38	MARVEL			[39]	NO
39	Melmac			[6]	NO
40	Merlin			[39]	NO
41	MVP-L			[42]	NO
42	OIKOS			[39]	NO
43	OORAM			No references	N/A
44	PADM			[39]	NO
45	PEACE+			[39]	NO
46	Petri Net		[40]	[27]	YES
47	PMDB+			[14]	N/A
48	Process Weaver			[39]	NO
49	PROMENADE	2002	[18]	[22]	YES
50	PSL			No references	N/A
51	RAD			[13]	N/A
52	REA			[13]	N/A

53	RosettaNet			[13]	N/A
54	SDL			[22]	YES
55	SLANG			[39]	NO
56	SOCCA			[39]	NO
57	SPADE			[39]	NO
58	SPELL			[39]	NO
59	SPM	2002	No references	No references	YES
60	STATEMATE			[14]	N/A
61	System Dynamics			[14]	N/A
62	TEMPO			[39] [14]	NO
63	UEML		[27]		YES
64	UML	1997-2004	[33]	[22]	YES
65	UML2	2002-2004	[12] [11]	[13]	YES
66	UPM			[22]	YES
67	Woflan			[43]	N/A
68	WPDL			[27]	N/A
69	XPDL			[27]	N/A
70	YAWL			[43]	N/A

2.2 Issues related to the justification of IDEF0 as process modelling language

The following paragraph, quoted from the review report, points to a possible problem in the justification of the choice of the process modelling language IDEF0:

“Choosing a modelling language (process modelling methodology) for expressing the guideline development model. They defined desired features of a modelling language (maintenance, clear graphic-textual notation, tool support) and screened 70 languages. (...) At the end IDEF0 was selected as the most suitable language. However, no good justification was given to the choice of IDEF0 as opposed to UML, which have the same final score.”

Another consideration related to this issue is showed in the following paragraph, where the reviewers suggest the possibility of using the activity diagrams of UML to model guideline development processes:

“While UML is not a process modelling language, it does include activity diagrams, which support this task. IDEF0 is probably better choice, but it is not fully justified by their work. The problem is probably the inclusion of UML in the list in the first place.”

In summary, the justification given to choose IDEF0 as process modelling language over UML is deemed insufficient, mainly because the score they got was similar. In order to better support this decision, we have carried out a deeper analysis of the two languages, studying in the literature their syntactic

and semantic features, the correspondences between the elements they incorporate, and, lastly, their suitability to model a subprocess of the evidence-based guideline development.

3 Justification of IDEF0 as process modelling language

In this section, we describe UML and IDEF0 process modelling languages, including details of their syntax and semantics. Afterwards, we present a comparison between these languages based on the information in the literature but also in terms of their suitability to model evidence-based guideline development.

3.1 UML

In accordance with *Booch and al.* [8], UML (the Unified Modelling Language) may be defined as a graphical language for visualising, specifying, constructing, and documenting the different aspects of a software system. UML fuses the concepts of Booch, OMT, and OOSE, obtaining a single and usable modelling language for users of these and other methods. Next, we describe the main concepts (goals, syntax, semantics) of UML according to the UML user guide [8].

The main goals of the UML are the following:

- to provide users with a ready-to-use, expressive visual modelling language and exchange models
- to support specifications that are independent of particular programming languages and development processes
- to provide a formal basis for understanding the modelling language
- to support higher-level development concepts such as components, collaborations, framework and patterns
- to integrate best practices

Therefore, UML tries to be an easy and understandable modelling language with a graphical representation, independent of the programming language, and that facilitates the exchange of models.

Next, we will explain the main syntactic elements and semantics of UML. UML has three building blocks which describe the main characteristics of the language, namely:

1. **things**, which are abstractions in a model, the basic elements
2. **relationships**, which link things together
3. **diagrams**, which group collections of things

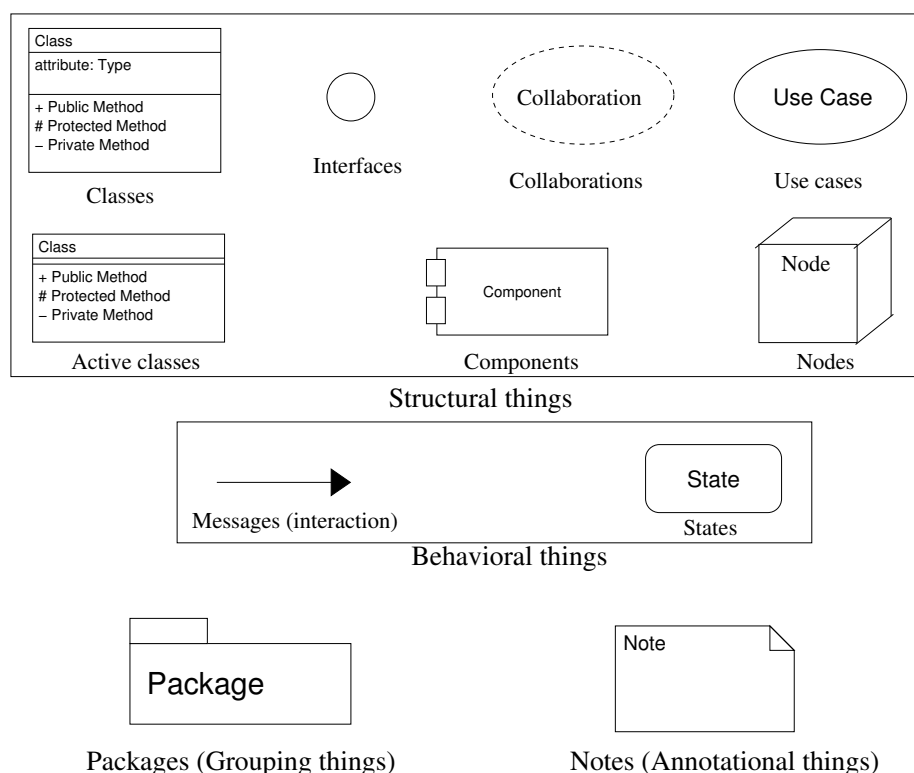


Figure 1: Main Things of UML

Things There are four kinds of things (see figure 1) in UML: structural, behavioural, grouping, and annotational things. Firstly, *structural things* represent conceptual and physical elements, and fall in the categories: class, interface, collaboration, use case, active class, component and node. Secondly, *behavioural things* are the dynamic parts of UML models. The two types of behavioural things are: interaction and state machine. Thirdly, the *grouping things* are the organisational parts of an UML model. There is only one kind of grouping thing, namely, package. Finally, *annotational things* are the comments used to describe any element.

Relationships There are four kinds of relationships (see figure 2): dependency, association, generalisation and realization. A *dependency relationship* is a semantic relation between two things in which the changes to one affect the other. An *association relationship* is a structural relation that describes links between two objects. There is a special kind of association relationship called aggregation. An aggregation represents a relation between a whole and its parts describing a “has-a” relationship. A *generalisation relationship* is a specialisation/generalisation relation between a general thing (superclass or parent) and a more specific kind of thing (subclass or child). In these cases, the child shares the structure and the behaviour of the parent. Finally, a *realization relationship* is a semantic relation between classifiers in which one classifier specifies a contract that another classifier guarantees to carry out. These kind of relationships

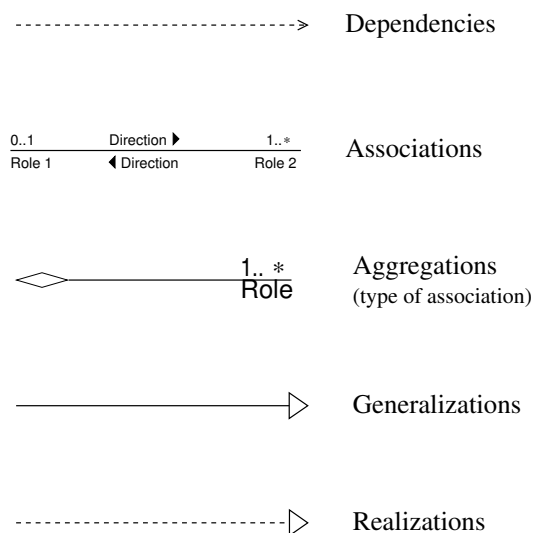


Figure 2: Main Relationships of UML

usually appears between an interface and a class, and between a use case and a collaboration.

Diagrams The third kind of UML block are diagrams, which are a graphical presentation of a set of elements. The diagrams are useful to visualise a system from different perspectives. UML includes nine types of diagrams:

- *use case diagram*. These diagrams show the static use case view of a system. They show a set of use cases and actors, with the relation between the actors and the system. Use case diagrams are especially important in organising and modelling the behaviours of a system.
- *class diagram*. A class diagram shows the static design view of a system. Class diagrams show a set of classes, interfaces and collaboration, and their relationships. Class diagrams are the most common diagrams in modelling object-oriented systems.
- *object diagram*. Object diagrams show a static design view, from the perspective of real cases. They show a set of objects and their relations.
- behaviour diagrams:
 - *state-chart diagram*. State-chart diagrams show the dynamic part of a system. A state-chart diagram shows the sequences of states an object undergoes during its life in response to external stimuli. State-chart diagrams are state machines consisting of states, transitions, events, and activities.
 - *activity diagram*. Activity diagrams are a special kind of a state-chart diagram which show the flow of control from activity to activity within a system. They also show the dynamic part of a system.

- interaction diagrams. They describe interactions, which consist of a set of objects and their interactions, including the messages that are sent among them. Interaction diagrams show the dynamic view of a system. There are two types of interaction diagrams:
 - * *sequence diagram*, which emphasize the time-ordering of messages.
 - * *collaboration diagram*, which emphasize the structural organisation of the objects that send and receive messages.
- implementation diagrams:
 - *component diagram*. They show the organisation and dependencies among a set of components. Component diagrams show the static implementation view of a system.
 - *deployment diagram*. They show the configuration of run-time processing nodes and the components that live on them. It shows the static deployment view of an architecture.

The static parts of a system are described in class, object, component and deployment diagrams. On the other hand, the dynamic parts are shown by use case, sequence, collaboration, state-chart and activity diagrams.

The diagrams of UML which can be used to model business processes are activity diagrams, state-chart diagrams and use case diagrams. In section 3.3 we present a comparison between IDEF0 and UML languages, together with an analysis based on the information in the literature of the diagrams that can be used to model business processes.

3.2 IDEF0

As described in deliverable D1.3, IDEF0 is a business process modelling language widely used in process modelling by organisation engineers. The IDEF0 notation (Integration Definition for Function Modelling) was developed during the 1970s as part of the U.S. Air Force program for Integrated Computer Aided Manufacturing (CAM) and was formalised by its publication in early 1980s [5] [7].

The main objectives of this technique are the following [5]:

1. To provide a means for modelling functions required by a system, and functional relationships and data that support the integration of those functions.
2. To provide a modelling technique with the following characteristics:
 - (a) generic (for analysis of systems of varied purpose, scope and complexity);
 - (b) rigorous and precise (for production of correct and usable models);
 - (c) concise (to facilitate understanding, communication, consensus and validation);
 - (d) conceptual (for representation of functional requirements rather than physical or organisational implementations);

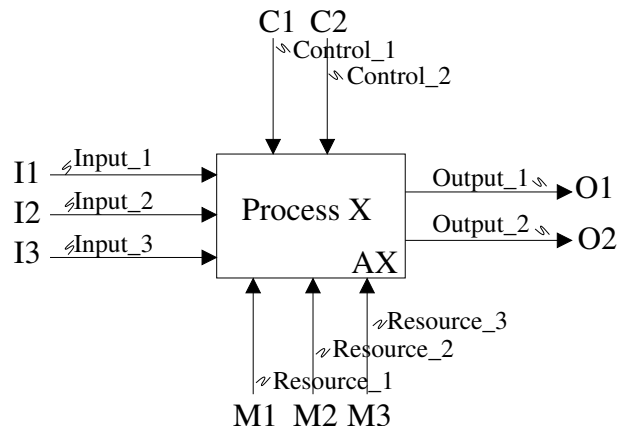


Figure 3: Main elements of IDEF0 syntax

(e) flexible (to support several phases of the life-cycle of a project).

The use of this standard is recommended for projects that:

1. Require a modelling technique for the analysis, development, re-engineering, integration, or acquisition of information systems.
2. Incorporate a systems or enterprise modelling technique into a business process analysis or software engineering methodology.

The main elements of IDEF0 are:

1. Process/activity: a process/activity is shown as a rectangular box. It contains a text description or label that describes what the process is. Process labels and identifiers are unique within a process model [5].
2. Input arrow: a line entering the process on the left. Inputs represent material or information which is consumed or transformed by the process in order to obtain the outputs. Each input has a label that describes information used by the process [26].
3. Output arrow: a line leaving the process to the right. Outputs are the material or information produced by the process. Each output has a label which describes information delivered by the process [26].
4. Control arrow: a line entering the top of the process. Controls regulate how, when, and if a process is performed and which outputs are obtained. They are often in the form of rules, regulations, policies, procedures or standards. The label describes the constraint on the process [26].
5. Mechanism arrow: a line entering the bottom of the process. Mechanisms are resources in the process and may be people, databases, software, machinery or equipment that is used [5].

These basic concepts of IDEF0 can be seen in figure 3.

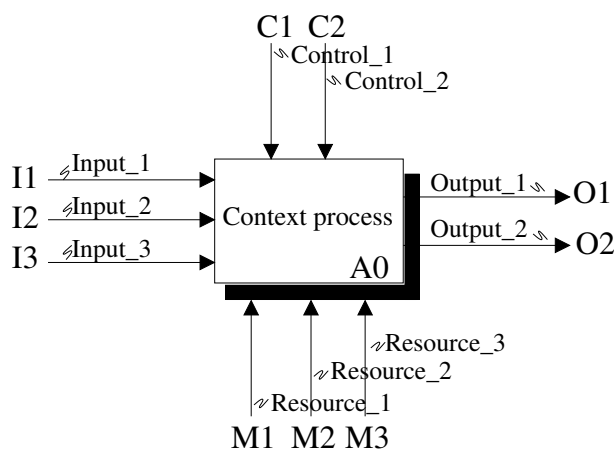


Figure 4: IDEF0 syntax: A-0 Diagram (context diagram)

There are some concepts which are crucial to understand better IDEF0 modelling. These concepts are: A-0 Diagram, parent/child diagrams, ICOM, tunneling and feedback. The *A-0 Diagram* is a special case of process. It is the context diagram which contains the top-level function, and it is composed of one box (see figure 4) [5]. Usually a process is composed of a set of activities. In this case, the activities (called “child boxes”) are child processes of the upper-level process (called “parent box”). Taking into account these concepts, a *child diagram* is a diagram which details a parent box, and a *parent diagram* is a diagram which contains a child box [5].

ICOM (Input, Control, Output, Mechanism) codes relate boundary arrows on a child diagram to arrows connected to its parent [5]. This coding identifies the arrow as an Input, Control, Output or Mechanism on the parent box.

Tunneling is a mechanism used to provide information at a specific level of decomposition which is not required to understand at other levels. There are two possibilities of tunneling. In the first case, the boundary arrows do not correspond to arrows into the child diagram. In the second one, the boundary arrows do not correspond to arrows connecting to the parent box. The latter possibility is advisable if the degree of detail at the parent process is high [5] [7]. IDEF0 allows to include *feedback* flows in between a process and previous processes. These feedbacks are outputs which can be inputs, controls and mechanisms of other processes [26].

Another important notion is the call arrow concept (see figure 5). A *call arrow* is a special type of mechanism that allows links to processes in a different part of the process model or even in a different process model. The arrow points downwards out of the process box instead of upwards as it is normal for mechanisms. If a call arrow calls a process within the same process model, it is labelled using the node number of the model sheet on which the called process is located, followed by the identifier of the process called. If a call arrow calls a process from a different process model, the identifier is prefixed by the name of the model in which the called process is located. When a process is called, only the parent process is identified. All child processes are automatically called along with the parent. A call arrow can call several processes. However, only one can be used at a time [5] [7].

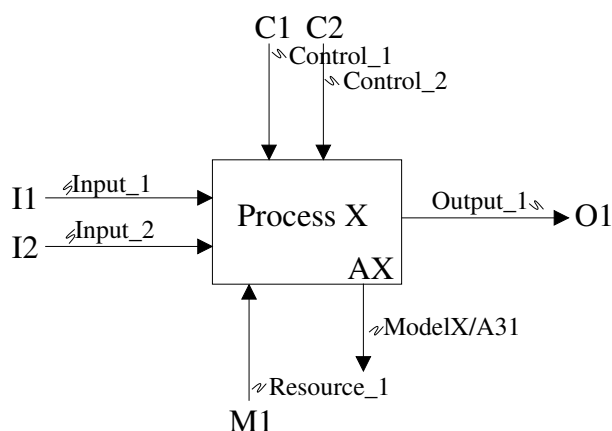


Figure 5: IDEF0 syntax: call arrow

3.3 UML VS. IDEF0

In this section, we focus on the main characteristics of UML and IDEF0 languages to model business processes. Furthermore, the relation between the different IDEF models and UML diagrams is described.

The goal of IDEF0 is to model the business processes of an organisation. Therefore, IDEF0 seeks to represent what actions are performed in an enterprise, and how they are performed, facilitating the understanding and the communication among its members.

The IDEF0 syntax allows to represent the processes as a box, the information flow (Input, Output) as arrows going in and out of the box, and the resources necessary to perform the process (Mechanism, Control) as arrows going up and down. The information flow and the resources are known as ICOM. Therefore, the IDEF0 syntax is centered on the processes and how those processes are interconnected.

Processes in IDEF0 can be decomposed into other processes (sub-processes) (see figure 6). In this way complex processes can be represented at different levels of depth, adopting a structure of hierarchy, and facilitating the refinement of processes into subprocesses.

On the other hand, UML was created with the goal of encompassing the main concepts of different (object-oriented) modelling languages, incorporating the positive aspects from each of them. Although UML was mainly designed to develop executable software, it has some diagram types which can be used to model business processes [8] [5] [4].

Since the final goal of UML is to develop executable software, the different syntactic elements are structured around the use case (context) part, the dynamic part and the static part of the system. According to some authors, the activity diagram is suitable for modelling business processes in UML [34] [8]. In addition to the activity diagram, in our analysis we also consider the utilisation of use case and state-chart diagrams, since some of their concepts are interesting and closer to the business process modelling perspective.

The activity diagram shows the flow of control from activity to activity within a system. It describes the activities and actions occurring in a system, showing part of the dynamics of the system. The activity diagram is considered

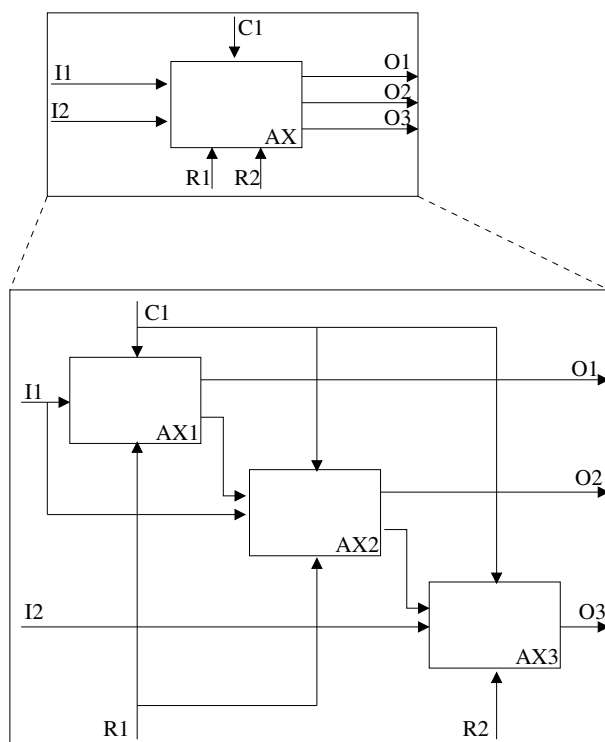


Figure 6: Decomposition of a IDEF0 process. Process AX is decomposed into three sub-processes, AX1, AX2 and AX3. Inputs, controls, outputs and mechanisms of AX process are the same in the decomposition, but now they go in/out the processes where they are used. Afterwards, one of theses sub-process could be decomposed into other sub-processes, and so on.

a special kind of a state machine in which most of states are activity states and most transitions are triggered by completion of activities [8].

In UML, an activity can be decomposed into some actions which are executed changing the state of the system. Moreover, it is possible to apply branches, forks, joins and swimlanes within the activities. A branch specifies alternate paths which can be taken based on Boolean expressions. A fork represents a splitting of a single flow of control into some concurrent flows of control. A join represents a link of two or more flows of control into one. Finally, the swimlanes are used to distinguish different working groups.

An analysis of the two modelling languages reveals several aspects of interest from the Protocure-II perspective. Although recent UML developments go in a different direction, the goal of IDEF0 and UML 1.5 languages was initially very different. Since the main goal of task T1.3 is to model the processes performed to develop a medical guideline, rather than building models that can be used as a guide in the development of any kind of software, UML does not seem to be the most appropriate candidate.

In UML, activity diagrams can be used in combination with swimlanes to capture the dynamic part related to business processes. In this case, the activity diagrams would be more centered on the actors who participate in the activities rather than on the activities themselves, which is our main focus. IDEF0, on the contrary, is centered on the process, which makes it a better candidate.

IDEF0 allows the decomposition into several levels of abstraction. In this way, a process can be split into sub-processes, which in turn can be split again. This decomposition can be performed as many times as we need, until the description of the process is deemed complete. Although the same kind of strategy can be imitated in the case of activity diagrams, by creating diagrams which refine the activities of other diagrams, UML does not support such kind of use. As result, the successive refinement of activities at several levels of abstraction, which is crucial when to model realistic processes, is much harder in UML than in IDEF0.

Some authors, as *Weston and al.* or *Cochran and al.* [4] [23], propose to use both languages to model business processes. *Weston and al.* propose using IDEF0 to represent the business process level (business front-end for UML) and UML at the application development level (standard software engineering back-end for IDEF modellers) [4].

Cochran and al. propose the combined use of IDEF0 process models and UML activity diagrams in order to increase the amount of useful information. Moreover, the UML activity diagrams complement the hierarchical decomposition of the IDEF0 model. The authors claim that a better understanding of the operational concepts can be achieved [23] with the two models.

There have been some attempts to establish an analogy between IDEF models and UML diagrams, but the correspondences are not clear. According to the UML manual and the literature, activity diagrams can be used to model business processes, which justifies the view that business processes can be modelled with either IDEF0 or UML activity diagrams. However, other authors consider activity diagrams more similar to IDEF3 instead of IDEF0. Furthermore, IDEF0 is sometimes considered similar to use case diagrams, but this similarity is only possible at the context level [4].

UML 2.0 is the latest and the largest member of the UML family [44] [27]. It continues moving away from a modelling language for object-oriented software

to a modelling language for “systems” in general, that need neither be software nor object-oriented. UML 2.0 incorporates many new constructs for modelling hierarchically-structured component-based systems. The two keywords are hierarchy and components. UML 2.0 supports the notion of structured classes, which are classes that have interconnected parts.

In general, UML 2.0 improves UML 1.x in many respects. First, it continues the shift away from object-oriented software to simply dynamic systems. Second, it introduces new low-level constructs that handle a number of process-related features that were missing from UML 1.x (timing diagrams, Petri net concepts in activity diagrams, token queueing concepts, and much richer action semantics). Third, the new emphasis on components and on structural and behavioural aggregation provides the foundation for defining business processes.

In summary, although UML (and more recently UML 2.0) has models that can be used to describe business processes, most modellers prefer to employ modelling languages which are specifically designed for this purpose [37] [34] such as IDEF0.

Finally, to further support our choice, we present an example of process modelling using both the IDEF0 language and the activity diagrams of UML. The example is taken from the processes performed in the Evidence Guideline Development process, concretely *A4-Literature Search*.

Figure 7 shows the IDEF0 model included in deliverable D1.3. There are three processes: *A41-Finding Relevant Literature*, *A42-Extracting Results from Literature*, and *A43-Combining the Results*. There are four inputs going into the first process, and three outputs from the third process. There is a flow of information between the three processes, i.e. *Relevant Literature* between A41 and A42, and *Literature Results* between A41 and A42. The main resources used in the process are the *Working Group*, *Literature Search Specialist*, *Journals*, *Databases*, and *Computing resources*. Only the *Working Group* participates in every process. The *Literature Search Specialist* participates in the first process, and the rest of the resources also in the first. *Working Group Control* performs the control on every process. Moreover, There is a control on the database access.

Figure 8 shows the activity diagram related to the same example. The activities in this case are the following: *Searching DB and Computing Resources*, *Finding Literature*, *Finding Evidence*, *Collecting Relevant Literature*, *Extracting Results*, and *Combining the Results*. Notice that *Searching DB and Computing Resources*, *Finding Literature*, *Finding Evidence*, and *Collecting Relevant Literature* correspond to the process *Finding Relevant Literature* of figure 7.

The previous additional activities have been included in order to separate the tasks carried out by the different actors, as it is required by UML activity diagrams. Note that the actors in this case, namely the working group and the literature search specialist, appeared as resources or mechanisms in the IDEF0 model (see figure 7). Furthermore, the activity *Collecting Relevant Literature* has been included to combine the outcome of the three other activities, with the purpose of obtaining a single data flow similar to the output *Relevant Literature* of figure 7.

The other two activities, namely *Extracting Results* and *Combining the Results*, correspond to the processes of figure 7 with a similar name. Furthermore, there are two swimlanes, in order to distinguish the two actors of the overall process. It is important to notice that in this activity diagram we have not included

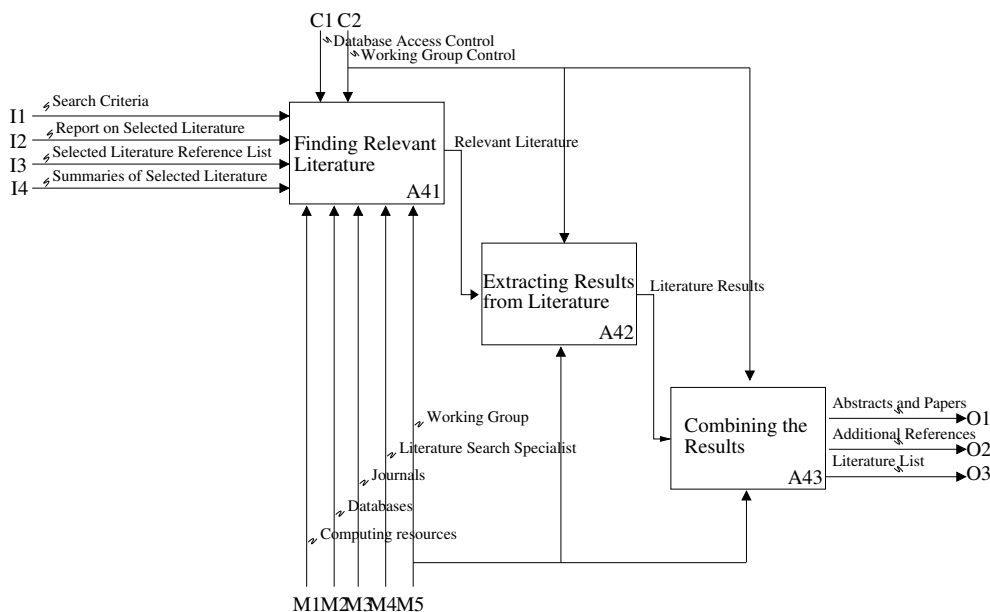


Figure 7: Example of an IDEF0 model. *A4-Literature Search* of the Evidence Guideline Development model

the particular objects which would help to understand the transitions between activities (e.g. the literature results would have to be an object attached to the link that indicates the flow of information between the extraction of results and the combination thereof). The inclusion of the data flow information would be laboursome and, at the same time, would add complexity to the diagram.

In the light of the above examples, we can see that the description of guideline development processes is much more intuitive and easy to represent using IDEF0 than using UML activity diagrams. For example, within an activity some useful information (about the input information, output information and information flow) cannot be directly (and easily) incorporated in the case of the activity diagram. In some cases the solution would require defining classes and objects. All in all, the clarity of the diagrams would decrease significantly. Also at the activity level, the representation based on swimlanes gives a view of parallel tracks which not always corresponds to the view of our experts, as opposed to the IDEF0 single-track process view, using simple arrows to indicate control and/or resource roles played by different actors.

4 Conclusions

Through these sections we have seen the main elements of IDEF0 and UML, and the different aspects of both languages. Moreover, we have shown that the models obtained using IDEF0 are easy to build and understand. Therefore, according to the main goal of the workpackage WP1, we have considered that IDEF0 could be the best option to communicate ideas between medical experts (guideline developers) and non-medical experts (computer scientists and knowledge engineers). Although UML can be also used to represent business

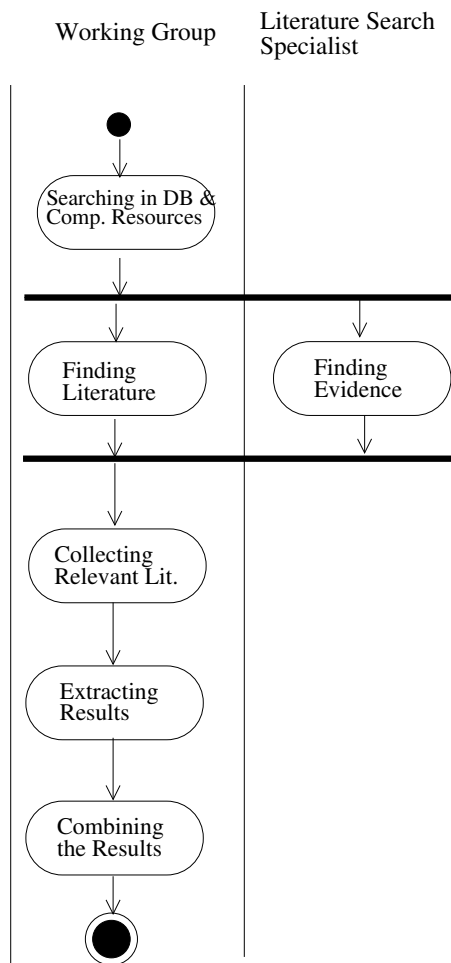


Figure 8: Example of an activity diagram. View of *A4-Literature Search* of the Evidence Guideline Development model

processes, it would require much more effort and would result in more complex models. Since our main goal is different from the development of software, UML is definitely not the best option.

IDEF0 is a language with 30 years of life and it is widely used in industry. Furthermore, IDEF0 is an intuitive language to represent business processes, and it is a standard in Department of Energy (DOE) in USA. On the contrary, UML is a young language, although widely accepted as well. Although the new version of UML constitutes a new step to the modelling of general systems, the goals of both languages were initially different, which results in an element set highly specific to their respective initial goals. As for future versions of UML, it is reasonable to assume that the main changes will be related to the (more recent) business process modelling part. Consequently, using any version of UML to model business processes would result in models with small chances of future adaptation. Probably this is the reason why many modellers prefer instead to employ languages that are specifically designed to describe business processes.

In summary, we have chosen IDEF0 because it is a language easy to use and understand, it has a wide trajectory and tools, and, mainly, because we wanted to capture the guideline development process in a way that facilitates the communication of ideas between guideline developers and computer scientists and knowledge engineers. Although UML can be used to represent business processes to some extent, it is a language much less mature in this regard. We are currently considering other modelling alternatives, but they are in the line of IDEF3 rather than UML. In addition to IDEF0-like constructs for modelling enterprise processes, IDEF3 includes useful features for the description of e.g. branching and temporal precedence of processes.

References

- [1] Cass A.G., Lerner B.S., McCall E.K., Osterwill L.J., Sutton S.M.Jr., and Wise A. Little-JIL/Juliette: A Process Definition Language and Interpreter. In *Proceedings of the 22nd International Conference on software Engineering*, pages 754–757, June 2000.
- [2] BMPI.org. Convergence Path toward a Standard BPM stack, August 2002. www.bpmi.org/downloads/BPML-BPEL4WS.pdf.
- [3] BMPI.org. *The BMPL Specification.*, June 2004. <http://www.bpmi.org/bmpl-spec.esp>.
- [4] Kim C.H., Weston R.h., Hodgson A., and Lee K.H. The complementary use of IDEF0 and UML modelling approaches. *Computers in Industry*, pages 35–56, 2003.
- [5] National Institute of Standards Department of Commerce and Computer Systems Technology. *Integration Definition for Function Modelling (IDEF0)*, December 1993.
- [6] Sutton S. et al. APPL/A:a prototype language for software process programming. In *ACM Transactions on Software Engineering and Methodology*, volume 4 N, pages 221–286, July 1995.
- [7] Internacional Alliance for Interoperability. Process Modelling For IFC Development . www.iai-international.org/iai-international/Technical_Documents/iai_documents.html.
- [8] Booch G., Rumbaugh J., and Jacobson I. *The Unified Modeling Language User Guide*. Addison-Wesley Longman, 1999.
- [9] Canals G., Boudjlida L., DerniameJ.C., Godart C., and Lonchamp J. ALF: A Framework for Building Process-Centered Software Engineering Environments. In Finkelstein A., Frammer J., and Nuseibeh B.A., editors, *Software Process Modelling and Technology*, pages 153–187. Research Studies Press, 1994.
- [10] Cugola G., Di Nitto E., Ghezzi C., and Mantione M. How to Deal with Deviations during Process model Enactment. In *Proceedings of the 17th international Conference on software Engineering*, April 1995.
- [11] OMG Object Management Group. *Catalog of OMG Modelling and Metadata Specifications*, March 2004. An adopted formal specification of the Object Management Group, Inc - <http://www.omg.org/technology/documents>.
- [12] OMG-SPEM Object Management Group. *Software Process Engineering Metamodel Specification - SPEM*, November 2002. An adopted formal specification of the Object Management Group, Inc - <http://www.omg.org/technology/documents>.
- [13] Mili H., Jaoude G.B., Lefebvre E., Tremblay G., and Petrenko A. Business Process Modelling Languages: Sorting Through the Alphabet Soup, 2003.

-
- [14] McChesney IR. Toward a classification scheme for software process modelling approaches. In *Information and Software Technology*, volume 37, pages 363–374, 1995.
- [15] Estublier J. Federations of Process support Systems, July 1998. www-adele.imag.fr/Les.Publications/intConferences/WPM1998Est.pdf.
- [16] Estublier J., Dami S., and Amieur M. "APEL: A graphical yet Executable Formalism for Process Modelling". *Automated Software Engineering ASE Journal*, 5, 1998.
- [17] Grundy J.C. and Hosking J.G. Visual Language Support for Planning and Coordination in Cooperative Work Systems. In *Proceeding of VI'96*. IEEE CS Press, 1996.
- [18] Ribó J.M. and Franch X. A Precedence-based Approach for proactive Control in Software process Modelling. In *SEKE 2004*, pages 457–464, July 2002.
- [19] Chen J.Y. and Tu C.M. CSPL: A process-centred environment. In *The Journal of Information and Software Technology*, volume 36, pages 3–11, 1994.
- [20] Yasumoto K., Higashino T., and Taniguchi K. Software process description using LOTOS and its enactment. In *Proceedings of the 16th international conference on Software engineering*, pages 169–178, 1994.
- [21] Inc KBSI Knowledge Based Systems. IDEF Family of Methods. A structured approach to enterprise modelling and analysis, June 2004. <http://www.idef.com>.
- [22] Zamli K.Z. and Lee P.L. Taxonomy of Process Modelling Languages. In *ACS/IEEE Int. Conf. on computer systems and Applications (AICCSA)*, pages 435–437, 2001.
- [23] Cochran L. and Wheaton K. A conceptual Operational Model for Command and Control of International Missions in the Canadian Forces.
- [24] Baldi M., Gai S., and Jaccheri M.L. An Initial Experiment with Object Oriented Software Process Modelling, October 1993.
- [25] Baldi M., Gai S., and Jaccheri M.L. Object Oriented Software Process Design in E3. In Nuseibeh B.A., editor, *Software Process Technology*. Research Studies Press, 1994.
- [26] Oskarsson M. A decision support system for scheduling a shop floor work center. Master's thesis, Dept. of Mathematics Chalmers Universtiy of Technology, January 1999. ISSN 0347-2809.
- [27] Petit M. and Doumeingts G. Project UEML: Unified Enterprise Modelling Language. Deliverable D1.1. Report on the State of the Art in Enterprise Modelling, September 2002.

- [28] Jaccheri M.L., Picco G.P., and Lago P. Eliciting Software Process Models with E3 Language. In *ACM Transactions on Software Engineering and Methodology*, volume 7, pages 368–410, October 1998.
- [29] Jaccheri M.L. and Gai S. Initial Requirements for E3: an Environment for Experimenting and Evolving Software Processes. In *Proc. EWSPT'92*, LNCS 635, pages 99–102. Springer Verlag, September 1992.
- [30] Jaccheri M.L. and Stalhane T. Evaluation of the e3 Process Modelling Language and Tool for the Prupose of Model Creation. In *PROFES*, LNCS 2188, pages 271–281. Springer Verlag, 2001.
- [31] Nguyen M.N., Wang A.I., and Conradi R. Total Software Process Model Evolution in EPOS. Experience Report. In *ICSE 97 Boston*, pages 390–399, 1997.
- [32] Tech Target Network. *BMPL-a whatis definition*. http://whatis.teachtarget.com/definition/0,,sid9_gci751337,00.html.
- [33] OMG-UML. *OMG Unified Modeling Language Specification*, March 2003. An adopted formal specification of the Object Management Group, Inc - <http://www.omg.org/technology/documents>.
- [34] Noran O.S. Business Modelling: UML vs. IDEF, June 2003. www.cit.gu.edu.au/noran.
- [35] Lucas P., Hommersom A., Galán J.C., Marcos M., Coltell O., Polo C., Rosenbrand K., Wittenberg J., and Croonenborg J. van. Deliverable D1.3: New model of guideline development process, February 2005.
- [36] Mi P. and Scachi W. Modelling articulation work in software engineering processes. In *Proc. 1st Int. Conj: Spftware Process*, pages 188–210, October 1991.
- [37] Rittgen P. Business Process Diagrams: An UML Extension.
- [38] Miller Q. and Sufrin B. Eclectic CSP: a language of concurrent processes. In *Proceedings of 2000 ACM symposium on Applied computing*, pages 840–842, 2000.
- [39] Conradi R. and Letizia J.M. Process Modelling Languages. In Derniame J.C., Kaba B.A., and Wastell D., editors, *Software Process: Principles, Methodology and Technology (PROMOTER2)*, number 1500 in LNCS, pages 27–52, jan 1999.
- [40] Wang S. and Sheldon F.T. PCX: A Translation Tool from PROMELA/Spin to the C-Based Stochastic Petri Net Language.
- [41] Chou S-C. A process modelling language consisting of high level UML-based diagrams and low level process language. In *Journal of Object Technology*, volume 1, pages 137–163, 2002. <http://www.jot.fm/issue/issue.2002.09/article3>.

- [42] Sutton S.M. and Osterweil L.J. The design of a next-generation process language. In Jazayeri M. and Schaure H., editors, *Software Engineering - ESEC/FSE'97*, LNCS 1301, pages 142–158. Springer, 1997.
- [43] Aalst W.M.P. van der, Hofstede A.H.M. ter, and Weske M. Business Process Management: A Survey, 2003. is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2003/BPM-03-02.pdf.
- [44] Unified modeling language - uml. <http://www.uml.org>.
- [45] Edoc uml profile, final adopted specification, February 2002. <http://www.omg.org>.