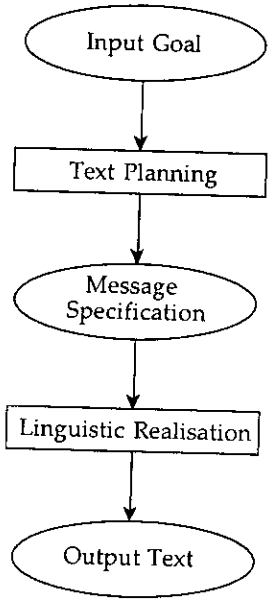
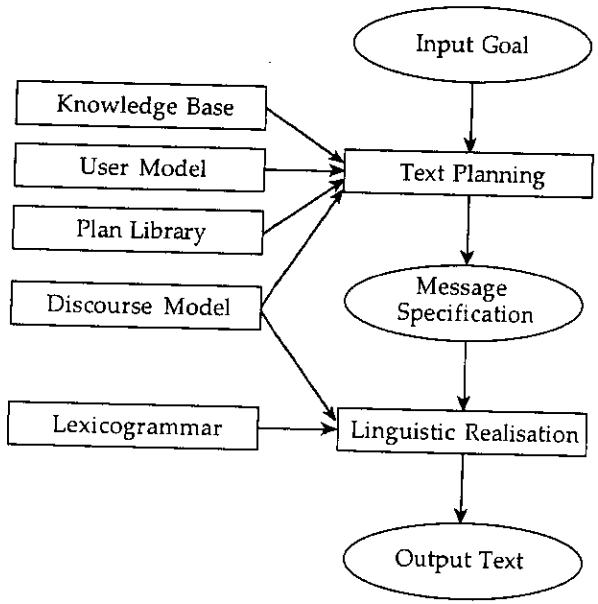
A Standard Archiecture for NLG



A Standard Archiecture for NLG



system could solve missionary/cannibal problems, for example, and still not know a cannibal from a bowling ball or where to pass the ketchup if a missionary has a heart attack. Genuine intelligence calls for a fuller, more versatile familiarity with the objects and events within its ken.

A micro-world can be known (represented) thoroughly: and that can be taken as a first approximation to common sense. To be sure, the phony blocks-world is remarkably unlike any bona fide toy chest; still, it does mimic the spatial properties that are in some sense basic. More important, the domain is not tailor-made for any particular ability or task. There are blocks-world studies of perception, planning, action, talking, learning, and so on; and since they all inhabit the same world, they could presumably be integrated into a single comprehensive system. Thus the artificial simplicity of micro-worlds is just a temporary expedient, a way of cutting through the distracting details of reality, to expose the basic principles of general intelligence—much as the early physicists ignored friction and deformation to get at the underlying universal laws of motion.

The best known and most impressive blocks-world program is Terry Winograd's (1971) simulated robot SHRDLU, designed primarily for communication in natural language (namely English). SHRDLU can carry on surprisingly fluent conversations, including complex noun phrases and ambiguous pronouns, as long as we talk only about the blocks on a particular table (see figure 1). He will often figure out what an unclear sentence must mean by considering what was said earlier or how the blocks are presently arranged. Moreover, SHRDLU is not all talk: if we ask him to move or build something on the table, he will uselessly rumpty, right before our wondering eyes (on a video screen, of course). Further, if our request entails that something else be done first, SHRDLU will take care of that all by himself; and if we then ask why he did whatever it was, he will politely explain. Finally, we can also add to SHRDLU's modest vocabulary at any time, merely by telling him what a new word means; and he will then use it correctly in both conversation and action (see box 5 for a sample dialogue.)

JOHN HAUGELAND 1985
ARTIFICIAL INTELLIGENCE:
THE VERY IDEA

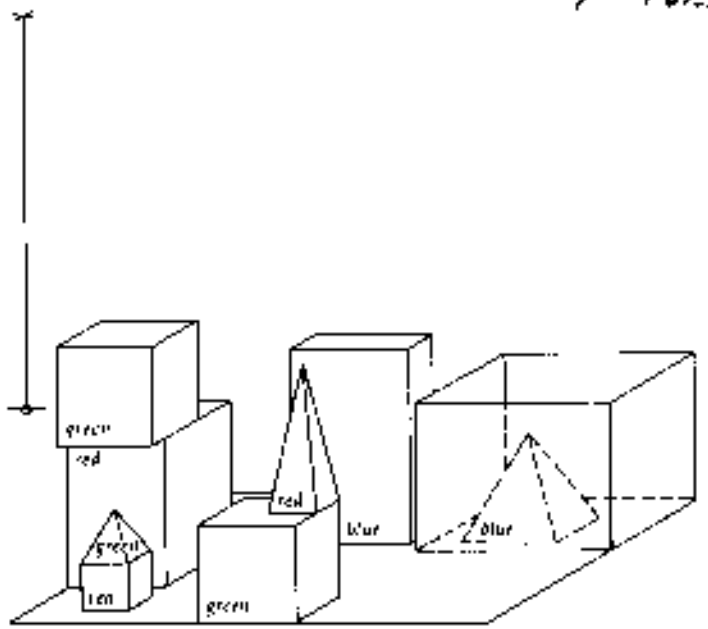


Figure 1 SHRDLU's "World"
This is SHRDLU's micro world (Winograd, 1972). The little circle dangling at the upper left is his "hand." Note that there is no actual tabletop with blocks on it, of which this is a picture; all SHRDLU really "inhabits" is a computer data structure describing a blocks-world situation. (That data structure can be connected to a video driver and screen, producing a drawing like the above for the amusement of spectators—but that's nothing to SHRDLU!)

Box 5
A Conversation with SHRDLU

The lower-case lines with "arrows" were typed in by a person; SHRDLU's responses are capitalized. The italicized paragraphs are explanatory comments added by Winograd 1971, pp. 35–55.

→ Pick up a big red block.
OK.

The system answers "OK" when it carries out a command. In order to pick up the red block, it had to clear it off by finding a space for the green one and moving the green one away.

→ Grasp the pyramid.
I DON'T KNOW WHICH PYRAMID YOU MEAN.

There are three pyramids in the scene, so the program does nothing.

→ Find a block which is taller than the one you are holding and put it into the box.

The system has to figure out what is meant by "one" and "it", as well as using deduction to decide which block it is holding.

BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.

Printed out to keep us informed.

OK.

The hand has to set down the red block to pick up the taller blue one.

→ What does the box contain?
THE BLUE PYRAMID AND THE BLUE BLOCK.

→ What is the pyramid supported by?

Earlier the system responded to "the pyramid" by saying "I don't know which pyramid you mean." Now, it knows that we must mean the one which was just mentioned.

THE BOX.

→ Does a steeple
SORRY, I DON'T KNOW THE WORD "STEEPLE".

The system interrupts immediately when it sees a word it can't figure out.

→ A "steeple" is a stack which contains two green cubes and a pyramid.

The quote marks indicate that we are defining a new word.

I UNDERSTAND.

→ Are there any steeples now?
NO.

The new definition is used as a description to recognize steeples.

→ Build one.
OK.

The description given in the definition is enough to deduce a plan for building new objects as well as recognizing them. The word "one" means "a steeple", and the plan is carried out.

SHRDLU's "natural" versatility and reasonableness are amazingly captivating. Lest anyone get swept away, therefore, let's note a few of his more obvious deficiencies right off the bat. First, he never initiates anything but only responds. In fact, SHRDLU has no motivational or purposive structure whatever, except for the goals (and subgoals) introduced by questions and requests from the outside. Second, he has no sense at all of conversational coherence or continuity; he is not bothered in the least by gratuitous shifts from "Pick up a big red block" to "Grasp the pyramid" to "Find a block which. . . ." A person, on the other hand, might well get testy and suggest you "make up your damn mind." Third, all the hard problems of perception and action involve the interface between *symbolic* cognitions and *nonsymbolic* objects and

number generator defined in Appendix 7 (which takes its seed from the computer's internal clock).

We list here only the DCG and 'semantics' parts of the ELIZA program. For a full listing, see Appendix 8.

```
reply([can,you,think,of,a,specific,example,?])
                --> ..._, [always], ..._ .
reply([tell,me,more,about,your,family])
                --> ..._, [Keyword], ..._,
                    {family(Keyword), not(already(family)),
                     assert(already(family))}.
                     % this trigger should not be repeated
reply([i,am,sorry,to,hear,that]) --> ..._, [Keyword], ..._,
                     {sad(Keyword)}.
reply([are,you,afraid,of,computers,?])
                --> ..._, [computer], ..._ .

reply([your|X]) --> ..._, [my], ...X.
reply([what,makes,you,think,i,am|X]) -->
                      ..._, [you,are], ...X .
reply([why,do,you,ask,?]) --> ..._, [?] .
reply([you,sound,very,positive]) --> [yes], ..._ .
reply([you,sound,very,negative]) --> [no], ..._ .
reply([do,you,think,coming,here,will,help,you,to,be|X])
                     --> [i,am,not], ...X.
reply([what,would,it,mean,to,you,if,you,got|X])
                     --> [i,need], ...X.
reply([in,what,way,?]) --> ..._, [are], ..._ .
reply([you,are,not,very,talkative]) --> [_]. % one-word answer

% replies if patient's input doesn't trigger anything

reply([i,do,not,follow,you])         --> ..._, {ok(0.25)}.
reply([i,am,not,sure,i,understand]) --> ..._, {ok(0.33)}.
reply([tell,me,more])                --> ..._, {ok(0.5)}.
reply([go,on])                       --> ..._, {ok(0.75)}.
reply([i,see])                       --> ..._.

family(mother).
family(father).
family(brother).
family(sister).
family(parents).

sad(sad).
sad(depressed).
sad(unhappy).
```


Here is a short example dialogue:

```
|?- doctor.
Tell me your problem.
|: all men are crazy.
in what way ?
|: they are always hassling me.
can you think of a specific example ?
|: yesterday my brother beat me up.
tell me more about your family
|: my mother is paranoid.
your mother is paranoid .
|: yes, she is always depressed.
can you think of a specific example ?
|: she gets very unhappy.
i am sorry to hear that
|: i need some help.
what would it mean to you if you got some help .
|: i would feel better.
i am not sure i understand
|: you are not very sympathetic.
what makes you think i am not very sympathetic .
|: your responses are very stilted.
in what way ?
|: you dont say much
go on
|: you are uncommunicative
what makes you think i am uncommunicative .
|: you dont say much .
i do not follow you
|: are you stupid or something?
why do you ask ?
|: your answers are very repetitive.
in what way ?
|: you keep saying the same things
go on
|: like that
tell me more
|: i am fed up with this
go on
|: i am leaving
go on
|: goodbye
goodbye
```

In fact, this dialogue is quite convincing, but you will not need to

Figure 1.3 FOG input: a weather system over Canada. (Courtesy of Environment Canada.)

Figure 1.4 Some example forecasts from FOG. (Courtesy of Environment Canada.)

```
96,122,1,5,2,00,200,-14.41,-3.668,-1.431,345,1023,15.41,15.82,20.07,-11.1,-2.878,104.2,28,153.6,53.19,0.16.26
96,122,1,5,2,25,215,-10.72,-3.241,-1.35,152,1023,15.3,15.78,20.07,-11.42,-2.762,105,208,98.2,822,0,17.05
96,122,1,5,2,50,230,-8.37,-1.282,-.904,2.15,1022,15.3,15.71,20.05,-11.66,-3.206,104.4,2.141.6,42.96,0.17.7
96,122,1,5,2,75,245,-12.81,-2.11,-1.067,2.119,1022,15.33,15.79,19.99,-11.15,-3.093,104.8,2.186,5.11,32,0,17.81
96,122,1,5,3,00,300,-13.68,-3,-1.35,1.075,1022,15.36,15.79,19.96,-10.63,-3.005,104.6,.402.285,8.61.45,0.18.47
96,122,1,5,3,25,315,-10.2,-2.457,-1.13,-.73,1022,15.32,15.66,19.92,-11.17,-3.263,103,6,.304,354,7,36,29,0,19.03
96,122,1,5,3,50,330,-9.33,-1.353,-.942,.902,1022,15.21,15.62,19.9,-10.95,-2.903,104.3,.313,302.2,34,69,0,19.16
96,122,1,5,3,75,345,-7.29,-.285,-.76,2.048,1022,15.24,15.63,19.87,-10.68,-3.27,104,.252,313,29,7,0,19.61
96,122,1,5,4,00,400,-6.822,-.365,-.653,1.531,1022,15.25,15.63,19.83,-9.93,-3.316,104,.331,274,2,52,98,0,20.42
96,122,1,5,4,25,415,-8.78,-.65,-.747,1.602,1023,15.35,15.66,19.79,-9.77,-2.656,103.3,.253,247.7,10,99,0,21.08
96,122,1,5,4,50,430,-8.73,-.641,-.741,1.785,1023,15.46,15.81,19.75,-9.16,-2.782,103.7,.2,295,29,15,0,21.3
96,122,1,5,4,75,445,-11.45,-2.671,-1.03,-.456,1022,15.46,15.82,19.74,-8.81,-2.464,103.7,.2,355.3,23.98,0,21.65
96,122,1,5,5,00,500,-13.12,-4.3,-1.306,-1.359,1022,15.42,15.75,19.76,-9.39,-2.49,103.4,.2,20.67,.188,0,21.83
96,122,1,5,5,25,515,-13.62,-4.621,-1.344,-.842,1022,15.32,15.67,19.81,-9.47,-2.703,103.7,.2,20.65,.183,0,21.98
96,122,1,5,5,50,530,-13.8,-3.534,-1.325,.943,1022,15.23,15.61,19.86,-10.92,-3.384,103.9,.2,20.65,.183,0,22.14
96,122,1,5,5,75,545,-14.7,-3.748,-1.419,.385,1022,15.06,15.47,19.9,-11.62,-2.868,104.4,.2,341.6,18.6,0,22.36
96,122,1,5,6,00,600,-13.61,-2.315,-1.287,2.038,1022,14.98,15.42,19.9,-12.37,-3.092,104.7,.2,298.6,5.173,0,22.54
96,122,1,5,6,25,615,-14,-2.894,-1.293,.669,1022,14.92,15.36,19.88,-12.48,-3.808,104.7,.591,320.3,21.07,0,22.87
```

The 22 data values in each record are as follows: the year; the day of the year as a value between 1 and 365; the month of the year; the day of the month; the time in 24 hour notation; the time in hours and minutes; four radiation readings; the barometric pressure; the temperature, referred to as 'dry bulb temperature'; the wet bulb temperature, which is used to calculate humidity; the soil temperature; the soil heat flux at two depths; relative humidity; average wind speed; wind direction; the standard deviation of wind direction; precipitation within the 15 minute period; and amount of sunlight.

Figure 1.2 Data input for the WEATHERREPORTER system.

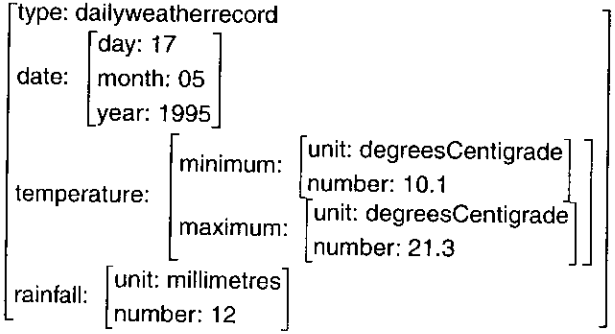
audience is hard wired into the operation of the system. IDAS, on the other hand, utilises a user model which records a variety of information; for example, if a user asks IDAS how to perform a task, IDAS's response is influenced by a user model which specifies which tasks the user already knows how to perform.

The discourse history. This gives information about previous interactions between the user and the system. The discourse history is perhaps most important for the task of generating referring expressions (described in Section 5.4), which is part of microplanning; but it can sometimes affect document planning as well. For example, when PEBA describes an animal, it attempts to relate this description to previous descriptions that the user has seen.

The output of the document planner is a document plan. Although the general notion of a document plan as something that dictates text content and structure is quite widespread within the NLG community – sometimes referred to as a discourse plan or a text plan – the specifics of the construct can vary quite significantly from system to system. In our architecture, a document plan is a tree, whose leaf nodes specify messages (content) and whose internal nodes specify structural information such as how messages are grouped, the discourse relations that hold between messages or groups of messages, and suprasentential structures such as paragraphs and sections. A more detailed description of the notion of document plan used in our architecture is provided in Section 3.4.4.

4.1.3 *A WeatherReporter Example*

For the purposes of producing a summary of the weather over a given month, the information presented in Figure 1.2 is far more detailed than we require. So, rather than work with these source data records directly, WEATHERREPORTER first summarises the data into a collection of DAILY WEATHER RECORDS, an example of which is shown in Figure 4.2. For the remainder of this chapter we will view these daily weather records as the primary knowledge source for WEATHERREPORTER.



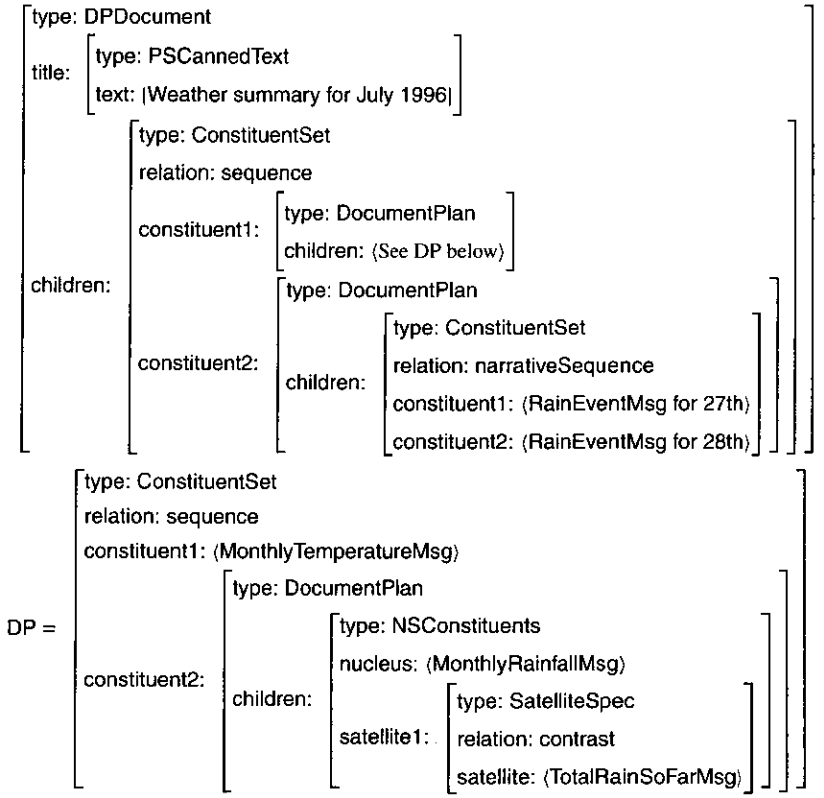


Figure 4.3 The document plan corresponding to the text in Figure 4.5.

An example of a WEATHERREPORTER document plan is shown in Figure 4.3; a schematic version of this is shown in Figure 4.4. This plan relates a set of five messages that have been constructed from the set of daily weather records for a particular month. One text that might be generated from this document plan is shown in Figure 4.5. Figure 4.6 shows two of the constituent messages that make up the plan; the first of these corresponds to the phrase *The month was slightly warmer than average*, and the second to the phrase *Heavy rain fell on the 27th*.

4.2 Representing Information in the Domain

In the context of the kinds of NLG systems we are concerned with here, texts are primarily used to convey information. This information may be expressed in words and sentences, but the words and sentences are not themselves information; the information underlies these linguistic constructs and is carried by them. In this sense, information is the predication of properties to entities or individuals – for example,

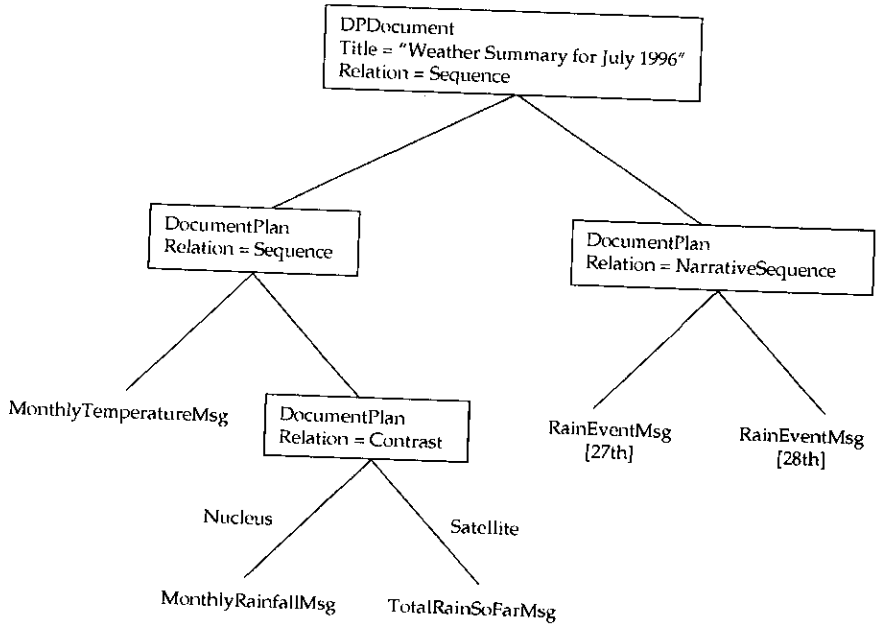


Figure 4.4 A document plan.

between entities – for example, that the echidna's body covering consists of long spines. We are concerned, therefore, with issues of knowledge representation and with the mapping of knowledge structures into semantic representations.

In any given domain, only particular predications or relationships will be of interest. We will conceptualise these significant predications and relationships as MESSAGES, informational elements that are to be considered for inclusion in the generated texts. Other terms sometimes used for these informational elements are FACT or PROPOSITION. We will avoid the second of these because there is a tendency to assume that propositions are expressed via sentences in a one-to-one manner, and, as we will see below, an important characteristic of messages is that they need not correspond to single sentences. By removing the assumption of a one-to-one mapping, we make it possible to contemplate combining messages to make more complex sentences. Messages are the basic elements or packages of information that the NLG system manipulates.

The messages themselves have constituent elements: the entities which are related or of which predications are made, the content of the predications, and the

> The month was slightly warmer than average with almost exactly the average rainfall, but rainfall for the year is still very depleted. Heavy rain fell on the 27th and 28th.

Figure 4.5 The weather summary for July 1996.

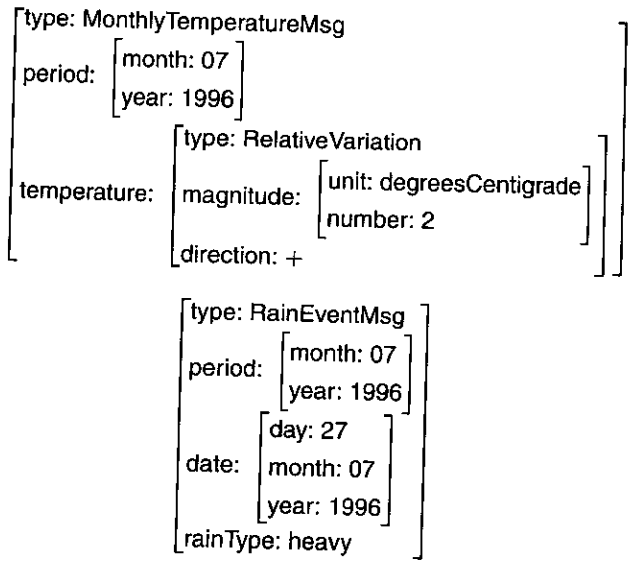


Figure 4.6 Example MonthlyTemperatureMsg and RainEventMsg messages from the document plan shown in Figure 4.3.

particular relationships identified. These are all elements of the DOMAIN. Prior to defining the messages to be used, it therefore makes sense to consider what an appropriate model of the application domain looks like. That is, what kinds of entities are there? what kinds of properties? what kinds of relations? By first developing a clear understanding of the raw material that makes up the domain, we are then in a better position to define the messages that may be derived from the underlying data; the messages are meaningful configurations of these lower-level elements.

Before we look at the process of message definition, we first look in more detail at this process of domain modelling. Both of these tasks are key aspects of the design of a document planner. Domain modelling and message definition are especially important in applications where the input data needs to be summarised or reasoned with before it can be made use of; this is the case in WEATHERREPORTER, for example, where simply reading off the contents of a set of daily weather records would not result in particularly interesting or useful texts. Domain modelling and message specification may appear less critical in systems such as IDAS which just select some information from their domain knowledge base to be communicated in the text. However, in such cases the knowledge base itself is often based on an implicitly-derived domain model. The domain modelling task has already been carried out, and the chunks of information in the knowledge base correspond closely to the packages of information we want to express in the text.

Ultimately, the heart of domain modelling and message definition is the determination of concepts which can be expressed linguistically and are appropriate