FINAL REPORT ON THE FARMER’S AID IN PLANT DISEASE DIAGNOSIS
R.J. Wieringa & P.H. Curwiel
Computer Science Department
Agricultural University
Hollandseweg 1
6706 KN Wageningen
THE NETHERLANDS
july 1986
## CONTENTS

### CHAPTER 1  PROJECT HISTORY

### CHAPTER 2  SYSTEM DEVELOPMENT

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>THE STRUCTURE OF AN INFORMATION BASE</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>DYNAMIC ANALYSIS OF SYSTEM PROCESSES</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>COMPOSITE SURROGATES AND STRUCTURED ATTRIBUTE VALUES</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>ANALYSIS OF PARCEL SELECTIONS</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>ANALYSIS OF PARCEL SEQUENCE AND ITERATIONS</td>
<td>34</td>
</tr>
<tr>
<td>2.6</td>
<td>NULL VALUES</td>
<td>38</td>
</tr>
<tr>
<td>2.7</td>
<td>IMPLEMENTATION</td>
<td>38</td>
</tr>
<tr>
<td>2.8</td>
<td>ANALYSIS OF DIAGNOSIS</td>
<td>43</td>
</tr>
<tr>
<td>2.9</td>
<td>DISCUSSION</td>
<td>52</td>
</tr>
</tbody>
</table>

### CHAPTER 3  KNOWLEDGE ACQUISITION

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>SOURCES OF KNOWLEDGE</td>
<td>53</td>
</tr>
<tr>
<td>3.2</td>
<td>PROBLEMS OF KNOWLEDGE ACQUISITION</td>
<td>54</td>
</tr>
<tr>
<td>3.3</td>
<td>INTERVIEWS WITH SCIENTISTS</td>
<td>58</td>
</tr>
<tr>
<td>3.4</td>
<td>INTERVIEWS WITH CROP PROTECTION SPECIALISTS</td>
<td>60</td>
</tr>
</tbody>
</table>

### CHAPTER 4  INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>INTRODUCTION</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>SITOBION AVENAE</td>
<td>65</td>
</tr>
<tr>
<td>4.3</td>
<td>PSEUDOCERCOSPORELLA HERPOTRICHOIDES</td>
<td>70</td>
</tr>
<tr>
<td>4.4</td>
<td>CORTICIJUM SOLANI</td>
<td>75</td>
</tr>
<tr>
<td>4.5</td>
<td>ERYSPHE GRAMINIS</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>FUSARIUM CULMORUM</td>
<td>86</td>
</tr>
<tr>
<td>4.7</td>
<td>GAUMANNO GRAMINIS</td>
<td>91</td>
</tr>
<tr>
<td>4.8</td>
<td>SEPTORIA NODORUM</td>
<td>96</td>
</tr>
<tr>
<td>4.9</td>
<td>PUCINIA RECONDITA</td>
<td>101</td>
</tr>
<tr>
<td>4.10</td>
<td>PUCINIA STRIIFORMIS</td>
<td>106</td>
</tr>
<tr>
<td>4.11</td>
<td>SEPTORIA TRITICI</td>
<td>112</td>
</tr>
</tbody>
</table>

### CHAPTER 5  USER INTERFACE
## CURRENT IMPLEMENTATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>CURRENT IMPLEMENTATION</td>
<td>121</td>
</tr>
<tr>
<td>6.1.1</td>
<td>The Disease Cause Process</td>
<td>121</td>
</tr>
<tr>
<td>6.1.2</td>
<td>The User Interface</td>
<td>126</td>
</tr>
<tr>
<td>6.1.3</td>
<td>The Report Writer</td>
<td>127</td>
</tr>
</tbody>
</table>

## CONCLUSIONS AND RECOMMENDATIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>EVALUATION OF BABYLON</td>
<td>129</td>
</tr>
<tr>
<td>7.2</td>
<td>CONCLUSIONS</td>
<td>133</td>
</tr>
<tr>
<td>7.3</td>
<td>RECOMMENDATIONS</td>
<td>137</td>
</tr>
</tbody>
</table>

## REFERENCES

- 121
- 126
- 127
- 129
- 133
- 137

## SOURCE LISTING
This report is the final report on the FAD project. The FAD project was initiated in September 1985 to test the expert system shell Babylon by developing a prototype crop disease diagnosis system in it. A short overview of the history of the project and the main problems encountered is given in chapter 1. Chapter 2 describes the result of an attempt to integrate JSD with modelling techniques like generalisation and aggregation and chapter 3 concentrates on the method we used to elicit phytopathological knowledge from specialists. Chapter 4 gives the result of knowledge acquisition for the 10 wheat diseases most commonly occurring in the Netherlands. The user interface is described briefly in chapter 5 and chapter 6 gives an overview of the additions to the implementation we made to the version of FAD reported in our second report. Chapter 7, finally, summarises the conclusions of the project and gives recommendations for follow-up projects.

A number of people working at different institutions have devoted their time to the project at one time or another as consultants or specialists on one or more diseases. These people are listed here in alphabetical order, followed by a code for the institutions where they are employed.

Bollen, Drs. G.J.  
Crijns, Ing. J.W.A.M.  
Daamen, Ir. R.A.  
Dekker, Ir. W.  
Gerlagh, Dr. Ir. M.  
Leemans, Ir. A.M.  
Rabbinge Dr. Ir. P.A. van der  
Smant K.  
Verhage P.  
Vliet G. van der  

DP  
PD  
IPO  
PAGV  
IPO  
IPO  
TT  
PD  
PD  
IPO
The meaning of the codes is

DP  Department of Phytopathology,
     Agricultural University
IPO  Research Institute for Plant protection
PAGV Research Station for Arable Farming and
     Field Production of Vegetables
PD  Plant Protection Service
TT  Department of Theoretical Production
     Ecology, Agricultural University

In addition, the project would not have delivered the results it has
were it not for the stimulating discussions with our colleague John
Simons from the computer science department. We thank all these
people and institutions for their willingness to devote time to an
undertaking of which the positive results for the improvement of plant
protection knowledge or service were not immediately clear. Without
their help, the results would have been a mere shadow of what they are
now.

R.J.W.
P.H.C.
CHAPTER 1

PROJECT HISTORY

In the past 11 months of system development, the following activities have been carried out. The results of these activities are reported in Wieringa & Curwiel [1985] and [1986] and in this report.

<table>
<thead>
<tr>
<th>period</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>sep 85 - nov 85</td>
<td>FAD1, 1st report</td>
</tr>
<tr>
<td>nov 85 - jan 86</td>
<td>2nd report</td>
</tr>
<tr>
<td>jan 86 - mar 86</td>
<td></td>
</tr>
<tr>
<td>feb 85 - apr 86</td>
<td>FAD2, 2nd report</td>
</tr>
<tr>
<td>feb 86 - apr 86</td>
<td>FAD2</td>
</tr>
<tr>
<td>apr 86 - may 86</td>
<td>FAD2, 2nd report</td>
</tr>
<tr>
<td>apr 86 - may 86</td>
<td>FAD2, 2nd report</td>
</tr>
<tr>
<td>may 86</td>
<td>FAD3, 3rd report</td>
</tr>
<tr>
<td>may 86</td>
<td>FAD3, 3rd report</td>
</tr>
<tr>
<td>jun 86</td>
<td>FAD3</td>
</tr>
</tbody>
</table>

Table 1

The major decisions made during the project were the decision not to use Bayesian updating, certainty factors, or Propspector-like updating, the decision to follow the general ideas of Jackson System Development (JSD) (Jackson [1983]) as far as possible, and the decision to use the aggregation and generalization hierarchies of Smith and Smith [1977a & b]. These decisions are argued in Wieringa & Curwiel [1986] and Simons & Wieringa [1986]. Other concerns were relegated to the background, viz. the structure of the user interface and the required database capabilities of the system.
In the first half of the project, many short memo's containing the stray ideas and results from discussions were produced, but once these decisions were made, memo production stopped and work started. The following graph shows the frequency of memo production.

\[\text{Figure 1}\]

The major transition occurring in February was the realisation that an expert system is an information system with one or two special functions, e.g. diagnosis, explanation, or planning, and that consequently all results concerning development method and system structure achieved in information system research are available for expert systems research. (Simons & Wieringa [1986]). The representation of uncertain information has a large tradition in mathematics, operations research and to some extent in information systems on which expert systems research can draw as well. Once this was seen and the relevance of JSD for object-oriented systems was seen, system development got under way.

The project was burdened with a multiplicity of tasks and goals. The goals were

1. To test Babylon;
2. To develop a demonstration expert system for plant disease diagnosis;
3. To find a system development method appropriate for expert systems.

The third goal is not mentioned in the original list of goals in the project proposal (Wieringa & Curwiel [1985], appendix C). It actually became our main concern, since we did not want to hack our way through a jungle without knowing what we wanted to do and how to do it and end up with something which no one would understand, cannot be maintained,
and would not be used.

The main concern during system development is not "Does it work?" but "What do we want to do?" These two questions are asked at the implementation level and model level, respectively. Similarly, the question to be asked when testing Babylon are not "What can I do with it?" but "Can I do what I want with it?" This requires that we know what we want to do with it.

Stepping back a second level of abstraction, the first major concern of system development is "How do we find out what kind of system we want to develop?" This is the question of method, which is followed by the question of model specification, and only then by the question of implementation.

In sum, the project consisted of an iteration of several activities, each with one of the implicit or explicit goals as its major concern. These concerns and activities are shown in table 2.

<table>
<thead>
<tr>
<th>concern</th>
<th>activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. system development</td>
<td>reading expert systems and information system literature</td>
</tr>
<tr>
<td>method</td>
<td></td>
</tr>
<tr>
<td>2. model structure</td>
<td>reading literature on ES and IS</td>
</tr>
<tr>
<td>3. model contents</td>
<td>talking with phytopathologists, crop protection specialists, reading introductory phytopathological literature</td>
</tr>
<tr>
<td>4. implementation</td>
<td>learn Babylon, program the model</td>
</tr>
</tbody>
</table>

Table 2

The iteration of activities is shown below.

![Figure 2](image-url)
The major results of the project concern the relation between system structure (parallel sequential processes, aggregation and generalization, chronicles) and development method (an extension of JSD). The phytopathological contents of the system are minor. Other results, which will have to be developed more, are the requirements for an implementation language. These results are summarized in chapter 7.

The project is left dangling in a state where a new model structure is proposed, without implementing it. The reason for leaving it in this state is that there is no need for a limited system like FAD. But there is a need for a crop management decision support system one of whose functions is disease diagnosis and management. Enough is now known to make a proposal for such a system. This proposal will be made in chapter 7.
CHAPTER 2

SYSTEM DEVELOPMENT

In this chapter we use JSD to develop a diagnostic information system. We change the terminology of JSD somewhat so that the words more accurately reflect their meaning and use and we integrate the modelling techniques of aggregation and specialisation in JSD.

2.1 THE STRUCTURE OF AN INFORMATION BASE

An information system is a quintuple of people, procedures, hardware, software and information base which support decision making in an organization. People act according to procedures, hardware operates according to software. An organization is a group of people acting toward a common goal (Simons & Wieringa [1986]). In our case, the organization is a farm, defined by the goal of making a profit out of crop production or livestock production. We restrict ourselves to winter wheat production. One type of decision to be made in wheat production is whether to spray against wheat diseases, and to be able to make that decision the farmer will have to diagnose the symptoms observed on the field and to be able to have some insight into what the possible consequences of his actions would be. We limit ourselves to the first part of this decision support function, diagnosis of the current situation. The second part concerns predictions of future behavior of diseases and would require models of pathogen population dynamics. These are available for some winter wheat diseases (Zadoks [1984]).

Viewed in this way, a diagnostic expert system for crop diseases is but one function of an information system. To be able to provide the disease diagnosis function, the hard- and software of the information system must be embedded in certain procedures. E.g., daily weather data will have to be communicated to the system and important actions like sowing and parcel treatment will have to be
entered into the system by the farmer. Existence of the information system imposes certain procedures on its environment which will have to be followed or the system cannot be used. This is the price that the environment must pay to be able to take advantage of the system.

It is important to realize that the goal of the organization is to maximize profit, not to heal the crop. Diagnosis should be sufficient to support optimal decisions towards this goal. The information required to diagnose a disease includes the observed symptoms on the field, the soil type of the parcel, the history of cultivation (crops, fertilizations, chemicals sprayed) and diseases on the parcel, and the weather pattern over a period of time preceding the date of diagnosis. To support the diagnosis function, the system should therefore contain a model of these events. We follow JSD in the general idea that the model should be defined and validated before the diagnostic function can be added, and that the function should be defined in terms of the model.

A distinction must be made between on the one hand the model itself, which consists of a structured representation of entity types and the possible state sequences of individuals of those types, and on the other hand a model instantiation, which is an explicit representation of individuals of those types which exist at a particular moment (at least, according to the system). At any particular point of time, the model consists of a representation of a number of objects, each of whom carries out a sequential process and some of whom may exchange messages. If the number of individuals represented is large, the model will be implemented in database. Usually, when the number of individuals is large, the time span of their existence is long, e.g. 30 years or more in models of administrative domains and several months for crops, and they change state slowly as compared to the execution speed of a computer. These processes are called long-running processes in JSD. Every change in modelled reality which is modelled by the information system is communicated to the model by sending a message across the model boundaries. This requires a procedure to be followed by the users of the system. Whatever the procedure, the system will lag behind the modelled processes and the system specification will have to state how much it can be allowed to lag behind. The model will spend most of its time waiting for a message from the environment that a change has occurred and that it will have to go through the same change.

If the number of represented individuals is small, their execution speed is usually much higher than it is in database systems. This is the case in real-time control. The time-lag between model and reality must be much smaller in this case and usually the messages to change state are communicated to the system by hardware connections.
In both the real-time and the database applications, the structure of the processes is the same, and can be defined as a composition of sequences, selections, and iterations of events, where an event is either a state-transition or the sending of a message to another process to change state. We use the term event instead of action because "action" suggests that the initiative of the action lies with the individual carrying out the action, while "event" is neutral with respect to the initiator of the event. For state transitions of an individual the initiative usually lies with another entity (usually not modelled) which causes the change, while for message sending the initiative lies with the individual sending the message. The process structures are called entity structure diagrams by Jackson. Because this term suggests something which has nothing to do with process structures and which we will introduce later, we will use the term event structure diagrams. Happily, this has the same abbreviation (ESD). The use of the term event agrees with the CSP language (Communicating Sequential processes) developed by Hoare (Sridhar & Hoare [1985]). The sequence, selection and iteration structures are universal in that any computation by a Turing machine can be represented by a composition of assignment statements according to these three composition schemas (Boehm & Jacopini [1966]).

During its life an entity goes through one of the possible sequences of events defined by its ESD. At any moment, there is a most recent event which has occurred and a series of possible next events, one of which will occur. With the ESD of every entity there is a last event pointer which marks the most recent event and therefore defines which types of events can occur next. The last event pointer is simply Jackson's text pointer and a high-level version of the program counter.

JSD has no solution for the storage of process histories, which we call chronicles. Histories should be stored, summarized and forgotten selectively. For example, the daily weather of the past 6 months or so should be remembered because it is relevant for the diagnostic function, but weather patterns longer past should be forgotten or summarized. For example, it should be remembered that last season it was cold and wet, but the daily weather pattern of last season should be forgotten. A chronicle contains the recent history of a process and a summary of the events longer past. Operators on chronicles will have to be defined and triggered at the appropriate time. We will largely ignore this problem in this report, but note that the required summarizing functions will probably be the same as the group functions of a DBMS like Oracle (Oracle [1984]).
Every modelled individual is modelled by a chronicle which has at least one entry, the entry representing the current state of the entity (or at least what the system takes to be the current state). Updates to the information base are either error corrections or state transitions in the life of an individual and in the extreme case are all stored in an unbounded chronicle. We assume that every entry in the chronicle for an individual is dated, i.e., it has a time stamp associated with it. When the representation of an individual is entered into the system, a surrogate for it is created (see Wieringa & Curwiel [1986]). All entries in the chronicle of an individual have the same surrogate and are distinguished by their time stamps.

What is usually called a model in the world of simulation is an explicit representation of expected state transition patterns. Dynamic simulation models of pathogen populations predict something about future changes in population parameters, but this prediction entails no restriction on the possible state transitions which actually will be communicated to the system. The ESD’s of the model define the process structure that cannot be deviated from but they still leave open which of the logically possible sequences of future events will occur. Simulation models predict something about the sequence likely to occur but they must remain within the constraints dictated by the ESD’s. Dynamic simulation models cannot deviate from the logical constraints defined by the ESD’s but the sequence of future events can deviate from what is predicted by the simulation models.

The different components of the information base of an information system can be summed up as follows.

information system = functions + information base + implementation

information base = representation of individuals + taxonomy of individuals + laws for existence and change

functions = report writers + causal functions + simulation models

The representation of individuals is what is usually called a model in database parlance.
The taxonomy and laws for existence are usually called the static integrity constraints while the laws for change are called the dynamic integrity constraints. The taxonomy defines the types of the individuals that can be represented to exist, e.g. PROJECT, PART, EMPLOYEE etc. To each type belongs an ESD defining the possible events and their allowed sequence for individuals of the type, e.g. BUY, SELL, HIRE, etc. The ESD's are dynamic integrity constraints defining what is logically possible and must be distinguished from simulation models which predict what can be expected. Simulation models are not part of the information base because they do not represent what exists. They are part of the function of the information system, which is separate from the information base but defined in terms of it. The types and their ESD's form a kind of high level abstract data type, where the operations defined for the type are not yet defined in any programming language but defined in terms of events in the domain.

The types can stand in certain logical relations to each other, e.g. aggregation, specialisation, and instantiation. For example, each PROJECT-PART is an aggregation of a PROJECT and a PART, an EMPLOYEE can be specialised to ENGINEER or SECRETARY, and an individually numbered PART in the quality control database can be an instantiation of a PART-TYPE of which the quantity is maintained in the inventory database. There may be existence constraints between individuals of different types, e.g. for each PROJECT-PART there must exist a PROJECT and a PART, and for each research PROJECT there must exist one employee participating in the PROJECT of subtype SECRETARY. These constraints can be quite involved and are discussed in Wieringa & Ourwiel [1986], section 2.1. The logical structure of the types and the logical constraints on the existence of individuals are called static integrity constraints.

The different function components will be discussed below.

The representation of individuals is entered when the information system is in operation.

The taxonomy and laws are defined during system development, and entered during system generation. Definition of the taxonomy and laws corresponds to what in expert system development is called knowledge acquisition. Knowledge in this context is an explicit representation of time-invariant structures of individuals and of constraints on the co-occurrence and changes of these individuals. For the purpose of this report, we regard knowledge as eternal and invariant, though more advanced data modelling can introduce higher level changes in the information base.
An information base thus contains much more structure than what JSD provides. Not only the currently existing individuals must be represented, a chronicle must also be maintained, and a taxonomical structure and logical constraints on the existence and change of individuals must be represented as well. At the function level, simulation models needed for prediction of pathogen growth must be added to the report writers and causal functions which already have a place in JSD. What causal functions are will be explained in a moment.

The three-level structure of information systems can be extended with these modules as shown in figure 3. is shown in figure 3.
The difference between the logical model and the empirical models is that the logical model excludes the states of the world which are logically impossible, while an empirical model describes states that can be expected on empirical grounds. The states predicted by an empirical model must be logically possible, but not all logically possible states are to be expected on empirical grounds. What is called a model in the information systems community is a logical description of what is there, while what is called a model in the expert systems community is an empirical model. On the logical level, the logical model simply follows the changes in the world. On the function level, an empirical model can make predictions about likely unobserved states in the past, present or future.

The interface of the system with reality consists of the report writers and the input subsystem. The input subsystem accepts messages from the modelled system. The connection can be either a data stream or state vector connection. In a data stream connection, reality writes messages into an input buffer which is read out by the input subsystem. The messages tell the appropriate model processes to change state. In a state vector connection, the input process continually scans a process in the modelled system to see if the model process has to change state. In both cases the input subsystem checks if the input from reality is correct and, if it can detect no errors, sends a message to one or more model processes to change state. It sends a message to more than one model process if they share common events.

The model reports to reality via output functions reporting on the current state, the chronicle, or the possible future of the model. The report is triggered either by a user request, or by an event in a model process, or by a tick of the system clock (e.g. the end of a day or week).

Both input and report functions can involve a dialogue with the user if they execute on-line. In some cases an input to the system will produce a report which will cause the user to take appropriate action (if he follows the procedures of the information system), the result of which will be communicated to the system again. For example, input of a withdrawal record for a customer account to an accounting system may produce the report "overdrawn account," which will cause the clerk to tell the system to block all accounts of that customer. Or a library member may terminate his subscription, the report of which will cause the clerk to cancel all reservations of this member and request for a list of books still borrowed by the member. These functions involve a loop from a report function back to an input function and can be automized by the system. They are called interactive by Jackson but will be called causal by us, because that
An event in one process (withdrawal from an account, termination of membership), results in message to another process to change state (e.g. tell an account to block itself, or cancel outstanding book reservations) or to a process outside the system (e.g. produce a list of outstanding books). The system knows the caused event must happen. Causal functions impose a regularity on the modelled system by letting the system carry out the prescribed actions.

2.2 DYNAMIC ANALYSIS OF SYSTEM PROCESSES

The part of our model supporting the diagnostic function represents relatively few individuals with a long history. Individuals represented include the parcels with their history of cultivations, and the weather with a history which is digitalized by periodic measurements. The weather is assumed to be the same for all parcels.

The process structure of these individuals is shown in figures 4 and 5. The meaning of the symbols in these diagrams is: The root boxes are entity types (or process types, depending on the language one wants to use), and the leaf boxes are types of events in the life of an entity. The other boxes are subprocesses in the life of an entity. An entity is an instantiation of a root box in an ESD and the events in its life are instantiations of event boxes in its ESD. The instantiation of an event type to an event is usually referred to as the occurrence of that event. To simplify language, we will often speak of events and entities and not of event types and entity types. Boxes which are ordered from left to right represent a sequential ordering of subprocesses or events, boxes with an asterisk (*) represent an iteration, boxes with an 0 represent a selection of subprocesses/events.
Figure 4

DYNAMIC ANALYSIS OF SYSTEM PROCESSES

SYSTEM DEVELOPMENT
These ESD's are adapted from a report of one of our students on the use of JSD in the development of plant disease diagnosis systems (Bruyn [1986]). They would probably be revised after discussions with the user. One possible improvement would be to reduce the PARCEL BODY to an iteration of TREATMENTS and create a separate entity CROP which has common actions with PARCEL and which to indicate the sequence of actions for each cultivation. We will not do this here, for the current structure gives us more material for illustration of the modelling process.

These processes shown in figure 4 and 5 are connected to the modelled system by data stream connections which contain messages that a certain state transition has taken place in reality and must be carried out by the model as well. The ESD's of WEATHER and PARCEL are dynamic integrity constraints on the possible sequences of actions for these entities. Events not shown in the ESD's are not modelled. In particular, real-world events like infection, pathogen dispersal, discolorization, development of spots, arrival of bugs etc. are not modelled by the events of the system. Modelling of these events would require that the farmer regularly observe the situation on the parcel and send appropriate messages to the model. Not only would this be an inordinate amount of work for the farmer, the relevant observations can not be done reliably enough to guarantee that the model processes accurately reflect the situation on the field, and besides process structures describing the possible sequences of all these events would become extremely complicated.

Adding the diagnosis function and report writer as separate processes, the system structure is shown in figure 6. This diagram is called the System Specification Diagram (SSD) in JSD. Each box in the SSD represents an entity type. Boxes are connected by data stream connections, represented by circles, or state vector connections, represented by diamonds. In a data stream connection there is a writer which writes information to an unbounded buffer, which is eventually read in the same order it was written in by a reader. In a state vector connection the reader inspects the state vector of a process without disturbing it. In both cases the direction of information flow is indicated by an arrow. If one instance of one type communicates with many instances of another type, multiplicity is indicated by a double line on the many side. For each PARCEL one DIAGNOSIS process is started and each DIAGNOSIS concerns only one PARCEL. There is only one WEATHER process, which can be queried by many DIAGNOSIS processes. Each DIAGNOSIS process creates (and destroys) one instance of the REPORT process as needed. A DIAGNOSIS process can create many REPORTS in its life. Each REPORT belongs to one DIAGNOSIS process.
The WEATHER data stream is rough merged with the ticks of a clock indicating that a period (e.g. a day) has passed. It is not specified by the diagram if this is a hardware connection or whether it must be carried out by the farmer. The PARCEL data stream will have to be maintained by the farmer. This is part of the procedures of the information system which the people using it will have to follow.

The DIAGNOSIS function receives a request from the user to produce a report on the possible disease causes at the time of the request (for the state of the model) and uses state vector connections to look at the WEATHER and PARCEL processes. The connections in figure 6 are an extension of JSD state vector connections, because they not only look at the current state, but also look at the chronicle of these processes. They could more aptly be called process chronicle connections.

If all actions relevant for diagnosis were modelled by the system, the input to the DIAGNOSIS function would simply be a request to produce a report. Since many of the events to produce the diagnosis report are not modelled, the DIAGNOSIS process must request information about the relevant events from the user. The dialogue which ensues is called a CONSULT. The structure of a CONSULT will be described in chapters 4 and 5. Chapter 4 describes empirical constraints on plant diseases, chapter 5 describes the structure of
the user interface. The DIAGNOSIS process is the simple iteration of CONSULTs shown in figure 7.

![Diagram showing the process of DIAGNOSIS and CONSULT]

The DIAGNOSIS function is used to decide whether at the time of the request an event should take place in the life of a parcel, viz. the SPRAY event. It could not be a causal function, for then input of the relevant data would result in a message to PARCEL to carry out the SPRAYing, which would then go through a state change which has not yet occurred in reality and may never occur, if the farmer decides not to spray. It must therefore be a simple report function.

The DIAGNOSIS function is a process with the results of the last diagnosis as its current state. When chronicles are maintained, results from previous diagnoses can be retrieved and previous diagnoses can even be re-run without interfering with the model processes. The results of diagnosis cannot be stored in the model process of the parcel because it does not model an event on the parcel. The process state of the DIAGNOSIS process contains a description of the symptoms reported to it and the possible disease causes it reported to the user.

2.3 COMPOSITE SURROGATES AND STRUCTURED ATTRIBUTE VALUES

In contrast to Wieringa & Curwiel [1986], we do not regard every structured attribute as an aggregation hierarchy. Instead, we now distinguish two hierarchies, the composition hierarchy and the structure hierarchy.
An unstructured surrogate is a surrogate without attributes. Examples are instances of INTEGER, REAL, or any other Pascal scalar type. A surrogate is structured if it has attributes. A type is structured or unstructured if its instances are structured or unstructured surrogates.

An attribute is a binary relation between two surrogates. The name of the attribute is an intuitive indication of the role which one attribute plays for the other. For example, an instance of EMPLOYEE can stand in relation AGE to an instance of a subrange of INTEGER. AGE is an attribute of EMPLOYEE with an instance of INTEGER as (unstructured) value. An attribute is called structured or unstructured when the surrogates which can be their value are structured or unstructured. This gives rise to a structure hierarchy in which referential integrity is optional, i.e. it is up to the system designer to demand that an attribute value exists or not. The problem of NULL values and referential integrity are thus treated in a uniform way. The structure hierarchy is drawn as in figure 8. In a structure hierarchy each structured type is analysed in its attributes by giving the name and type of each attribute, separated by a semicolon.

```
PART
  PART#: TYPE: AMOUNT: MANUFACTURER:
  PART##: TYPES INTEGER MANUFACTURERS
  NAME: ADDRESS: DISTANCE:
  NAMES ADDRESSES REAL
```

Figure 8

A compound or composite surrogate consists of surrogates. All components of an instance of a compound type must exist; A compound surrogate only exists if all its components exist. There is no surrogate for such an instance apart from the composition of surrogates out of which it is composed. For a compound surrogate referential integrity is demanded, i.e. the components must exist. If referential integrity cannot be guaranteed, then a surrogate with structured attributes must be chosen to represent the entity. For
example, a car with some parts missing is still a car and we select a surrogate representing the identity of the car (e.g. the surrogate for the engine block) and add the parts as structured attributes.

The composition of surrogates can be depicted by a composition hierarchy as shown in figure 9. The attributes of the instances of a type are written behind the type.

\[
\text{PROJECT}-\text{PART} (\text{NUMBER: INTEGER})
\]

\[
\text{PROJECT} (\text{PNR: PNRS, ...}) \quad \text{PART} (\text{PART\#: PART##, ...})
\]

Figure 9

The following problems are still open:

1. Can the composition hierarchy consist if more than one level?
2. Can an attribute value be composite?
3. Referential integrity for a compound is that a component exists during the lifetime of the compound. An interval-based temporal logic is needed to be able to check this.

2.4 ANALYSIS OF PARCEL SELECTIONS

The diagnosis rules are empirical constraints on the DIAGNOSIS, PARCEL, and WHEATHER processes. These constraints represent knowledge of the domain and can be used to infer possible values of unobserved variables from values of observed variables. The results of such inferences are not part of the model but of the function processes, since they do not model reality but simply reflect the state of current knowledge.

To specify the constraints, we need to analyse the PARCEL process. To analyse a process, we add process attributes to its ESD and analyse its selections, sequences, and iterations, using the semantic modelling techniques of specialisation, structuring,
composition, and chronicles.

There is a problem here about the nature of integrity constraints in general. The information base of the information system contains a representation of the individuals known to exist in the domain and of the integrity constraints on the occurrence the individuals, their states and their state sequences. Integrity constraints can either be analytic, in which case they follow from the meaning of the terms by which the domain is described, or synthetic, in which case they are empirical generalizations about the domain. Both types of constraints rule out certain states of affairs. ESD's are analytic constraints and can therefore to be used to rule out many event sequences. The rules connecting symptoms with disease causes are empirical but are hardly ever 100% deterministic and in the domain of plant diseases always allow exceptions. By storing these rules and the result of their application in the DIAGNOSIS process, we keep the part of the system of which we are not certain if it represents anything on the parcel separate from the model itself, which represent events definitely known to have occurred. For example, we definitely know a spraying to have occurred and this is stored in the model, but we do not know if the diagnosis septoria tritici is correct, so we store this in the chronicle of the DIAGNOSIS process, meaning that the diagnosis occurred, not that the disease occurred.

To analyse a process, we write the attributes which change during an event next to the box representing the event. The attributes of a state transition collectively describe the transition. If there are no attributes which change, then the event is not a state transition but a message to another process. The process attributes are the sum of all attributes of its events. Process attributes are local to a process and can only be changed in one of the state transitions of the process. If process A wants to change an attribute of process B, it has to send B a message to carry out a transition. A can only send this message at those points of its life where it appropriate according ot its ESD to send it, and B can only receive it when the event (state transition or message sending) it is called to carry out is allowed according to its ESD.

This can be implemented by giving every transition a name and defining for every transition a behavior with that name. The ESD's of A and B constrain the possible sequences of message sending and receiving and should be stored centrally, i.e. in a data dictionary. The exact nature of this storage is immaterial at this point. It could be relational if flexibility is required, or as a network if rapid access is required.
The attributes of a process are invisible to other processes unless otherwise stated in the process definition. It is a matter of convenience whether a simple distinction between private and public process attributes should be made or whether for the public variables it should be stated which other processes may see them. All process variables are local, meaning that only process behaviors have update access to them, but public variables can be inspected by other processes via state vector connections. Usually these will be function processes which need to query a model process.

Some process attributes for the Parcel process are shown in figure 10.
A leaf in an ESD is an event, which is either a state transition or a message. A state transition in the real world is modelled by a change of zero or more attribute values in the model. If no attributes are involved in the transition, then we only represent that an event of a certain type has occurred, not which (real-world) attributes changed in the event. If the event is not a state transition but a message, then the individual receiving the message must be named (possibly by an expression evaluating to its name) and the necessary information for the receiving individual to go through the event must be sent along with the name.

For every individual there is a first event which consists of its instantiation and a last event after which no event can take place anymore and it can only be destroyed. The attributes of the first event often are a summary of what is relevant about the prehistory of the entity, e.g. PREVIOUS-OWNER, and the attributes of the last event often summarize posthistory, e.g. NEXT-OWNER. Because every entity will go through at least these two events, the chronicle for every entity will contain at least entries for the entry and exit of the entity in the system.

Apart from prehistory attributes, at instantiation a number of attributes which characterize the entity through its life are instantiated. In figure 8, these attributes are X, Y, AREA, and SOIL-TYPE, which can only be changed by an error-correction procedure with special privileges. There are no events in the PARCEL's life which change their value. X and Y form a uniquely identifying description of PARCELS.

Gathering the attributes of the PARCEL process together, we get the following initial definition of the entity types. Some of the data types used will be analyzed below, others, like MONEY are assumed to have been defined. In a realistic system the PARCEL type will have to be elaborated but as an illustration it suffices.
THE PREDICATE UNIQUE \{X, Y\}\) INDICATES THAT THE SET OF ATTRIBUTES \{X, Y\}\) CONSTITUTE THE ONLY UNIQUELY IDENTIFYING DESCRIPTION OF PARCELS. EVERY ATTRIBUTE VALUE IS TIME-STAMPED BECAUSE EVERY UPDATE TO THE PARCEL ENTITY IS AN INSTANTIATION OF A NEW CURRENT ENTRY IN THE PARCEL CHRONICLE AND EVERY ENTRY IN A CHRONICLE IS TIME-STAMPED. THE MEANING OF THE TIME STAMP IS THAT THE STATE CAME INTO BEING AT THAT DATE, NOT THAT THE UPDATE TO THE SYSTEM OCCURRED AT THAT DATE. FOR EVERY PARCEL REPRESENTED BY THE SYSTEM THERE IS AT ANY MOMENT AN INSTANCE OF THIS TYPE WHICH REPRESENTS THE MOST RECENT STATE OF THE PARCEL.

THE ATTRIBUTES BUY, RENT AND RECLAIM AND PLOUGHED ARE OF A SPECIAL DATATYPE WHICH CONTAINS AS ITS ONLY VALUE AN INDICATION THAT PLOUGHING HAS OCCURRED. FOR THE ATTRIBUTES OF ACTIONS WHICH DID NOT OCCUR NULL VALUES MUST BE USED. THESE WILL BE ELIMINATED IN A MOMENT.

THE THREE ATTRIBUTES BUY, RENT AND RECLAIM HAVE A STATIC INTEGRITY CONSTRAINT THAT AT LEAST AND AT MOST ONE OF THEM MUST HAVE OCCURRED WHEN THE LAST EVENT POINTER HAS SELECTED ONE OF THEM OR HAS PASSED BEYOND THEM. THE TREATMENT ATTRIBUTES CROP-SOWED THROUGH SPRAY HAVE A CONSTRAINT THAT AT MOST ONE OF THEM CAN HAVE OCCURRED DURING ANY UPDATE IN THE TREATMENTS STAGE. THESE CONSTRAINTS LEAD TO THE IMPLEMENTATION OF THE ACQUISITION ATTRIBUTES BY ONE ATTRIBUTE, E.G.
ENTRY of type ACQUISITION, defined by

ACQUISITION = \{BUY, RENT, RECLAIM\};

This definition not only follows implementation considerations (save space) but also reflects a conceptual structure in the domain which we want to capture. But the conceptual structure is not fully brought out by this definition, since it creates the problem that in each of the three possible values of ACQUISITION there will be other extra attributes which we want to remember, e.g. PRICE in the case of BUY, and OWNER and MONTHLY-RENT in the case of RENT. These are just the prehistory attributes of each of the possible entries in the system. This structure is the Pascal variant record structure or Smith & Smith's specialisation/generalization structure and has been reviewed in Wieringa & Curriel [1986]. It can be represented by specialisation hierarchy of figure 11.

![Specialisation hierarchy of figure 11](image)

Figure 11

Specialisation hierarchies are drawn without boxes. Attributes are written behind the type to whose instances they belong. Shared attributes are stored at the root node and inherited by specialisations. Each specialisation can have attributes which are special to it. Some attributes can easily be extended by structure hierarchies, e.g. figure 12.
The logical connection between the dynamic selection structure and the static specialisation structure can be seen as follows. In general, every variable $X$ of type $T = \{X_1, X_2, X_3\}$ defines a specialisation hierarchy with unstructured leaves.

Every expression of type $T$ evaluates to an instance of which $X_1$, $X_2$, or $X_3$ is the prototype. Assignment $X := \text{expr}$ is a selection.
In other words, the type definition of X defines a specialisation hierarchy and during assignment one leaf is selected for instantiation. After assignment, X refers to that instantiation. Analogously, the ACQUISITION attribute has a type which is defined by a specialisation hierarchy and during assignment one leaf of the hierarchy is instantiated and put in the ACQUISITION attribute. This allows assignment of values of different types, themselves possibly highly structured, to one attribute of type T, as long as these types are specialisations of T. It also eliminates NULL values with the meaning "not applicable."

There are other specialisation structures apart from selection, e.g. the specialisation tree for disease causes or crops. Specialisation structures derived from selection are actually simple decision trees.

The exit attributes of the DISPOSE events can be analyzed in similar manner as the ENTRY attributes but this will not be done here.

The body of PARCEL is a SEASON iteration and the SEASON is an iteration of TREATments. We postpone analysis of the time structure of sequences and iteration to the next section and look at the state of a PARCEL after one TREAT in one SEASON. SEASON can be specialized as shown in figure 15 (cf. the PARCEL structure in figure 10). It consists of one specialisation group. Because in one specialisation group only one leaf can be instantiated, this expresses the constraint that in one SEASON a PARCEL either lies fallow or is cultivated. SEASONS has no attributes. The attributes of CULTIVATIONS will be analyzed below.
We can now describe a PARCEL by the following attributes.

TYPE PARCEL
ENTRY : ACQUISITION;
X : COORDINATE;
Y : COORDINATE;
AREA : SURFACE;
SOIL-TYPE : {CLAY-LIKE, SANDY};
SEASON : SEASONS;
EXIT : DISPOSAL;
UNIQUE \{\{X, Y\}\}  
END PARCEL

TYPE ACQUISITION =
    GENERALIZATION OF \{PROPERTY, RENTED, RECLAIMED\}
END ACQUISITION

TYPE PROPERTY =
    PREVIOUS-OWNER : OWNERS;
    PRICE : MONEY;
END PROPERTY

TYPE SEASONS =
    GENERALIZATION OF \{CULTIVATIONS, FALLOW-PERIODS\}
END SEASONS

The PARCEL attributes define a PARCEL during one iteration through SEASON. The PARCEL definition will be slightly changed to express dynamic constraints in the next section.

The notation GENERALIZATION OF \{\ldots\} denotes one specialisation group. An instance of a type is an instance of each specialisation group and an instance of a specialisation group is an instance of exactly one of its types. These two constraints express the demand that each specialisation group of a type consist of an exhaustive set.
of disjunct types.

The ENTRY attribute of PARCEL has type ACQUISITION, which according to the above definitions means it can as its value have an instance of PROPERTY. The implementation of this kind of structure will be discussed in section 2.7.

We will not analyse the attribute types in the definitions, except CULTIVATIONS. The CULTIVATIONS are defined by (cf. figure 10)

```
TYPE CULTIVATIONS =
  SOW : SEED;
  TREATMENT : TREATMENTS;
  REAP : VOLUME;
END CULTIVATIONS
```

VOLUME is left unanalyzed and SEED and TREATMENTS are defined by figure 16, 17 and 18.

```
SEED(PRICE: MONEY, ...)
```

- POTATOES
- WHEAT
- SUGARBEET

- SUMMER
- WINTER

- NAUTICA
- CITADEL
- SAIGA
- ORAPI
- MARKSMAN

Figure 16
Figure 16 specialises SEED to different crops and varieties and figure 17 shows the (structured) attributes all these specialisations have in common. The subtypes can easily provided with more attributes, such as susceptibility for different diseases, etc. The definitions of these types follows.
TYPE SEED =

| PRICE     | : MONEY;          |
| AMOUNT    | : AMOUNT-OF-SEED;|
| MANUFACTURER | : MANUFACTURERS; |
| GENERALIZATION OF {POTATOES, WHEAT, SUGARBEET} |

END SEED

TYPE WHEAT =

| SPECIALISATION OF {SEED}, |
| GENERALISATION OF {SUMMER-WHEAT, WINTER-WHEAT} |

END WHEAT

TYPE WINTER-WHEAT =

| SPECIALISATION OF {WHEAT} |
| GENERALISATION OF {NAUTICA, CITADEL, SAIGA, OKAPI, MARKSMAN} |

END WINTER-WHEAT

TYPE TREATMENTS =

| GENERALISATION OF {SEED, PLOUGH, FERTILISE, WATER, SPRAY} |

AMOUNT-OF-SEED, MANUFACTURERS, SUMMER-WHEAT, PLOUGH, FERTILISE, WATER, SPRAY will not be analysed. The different varieties of WINTER-WHEAT are unstructured types in the current definition. All their structure is defined by their supertype SEED. There are no attributes to distinguish two instances of NAUTICA and they can therefore be denoted by the same notation as we have for the type itself. The WINTER-WHEAT definition can therefore be simplified to

TYPE WINTER-WHEAT = [NAUTICA, CITADEL, SAIGA, OKAPI, MARKSMAN]

The information that WINTER-WHEAT is a subtype of WHEAT is removed from this notation but is redundant anyway. In a non-abbreviated definition, each subtype mentions its supertype(s) and only the top of a specialisation hierarchy has no supertype.

2.5 ANALYSIS OF PARCEL SEQUENCE AND ITERATIONS

During the life of a PARCEL it goes through a sequence of events, selecting some and iterating through others. This introduces a new type of NULL value, that of values which cannot yet be filled in because the event they describe lies in the future. E.g. the EXIT attributes cannot yet be filled in during most of the life of a PARCEL. We call this type of NULL value FUTURE. The UNKNOWN type of
NULL value is hereby restricted to unknown attribute values describing past or current events.

The addition of time to the information base has a more dramatic effect for the representation of past events, for which we introduced the concept of a chronicle. The chronicle of an instance of PARCEL is a summary of the history of that PARCEL. These concepts can be dealt with at the conceptual level as follows. The PARCEL definition is adapted by indicating the sequence in which the attributes get a value by semicolons, and separating attributes for which the order of assignment is immaterial by commas. Iterations are indicated by an asterisk.

```
TYPE PARCEL
  ENTRY : ACQUISITION;
  (X : COORDINATE,
   Y : COORDINATE,
   AREA : SURFACE,
   SOIL-TYPE : {CLAY-LIKE, SANDY});
  SEASON : SEASONS*;
  EXIT : DISPOSAL
UNIQUE {{X, Y}}
END PARCEL
```

With every instance of PARCEL an event pointer is associated which points at a semicolon in the instance definition. After instantiation it points at least at the first semicolon in the definition and possibly at a later semicolon, depending upon the instantiation procedure. All attributes beyond the event pointer have the special type of NULL value called FUTURE. An iteration attribute can get the NULL value called NONE when the event pointer passes beyond them without going through an iteration. The iterations through a group of attributes create a repeating group with a time sequence and a most recent element. Because the PARCEL definition represents a sequence of states, it itself represents a chronicle of the life of a PARCEL. The logical representation of the time sequence and dates is still a very open problem. Special care should be taken of the iterations because they may occur an unpredictable number of times. The definition should include how far back the iteration chronicle should go before the history longer passed should be forgotten or summarized. Adding this to the PARCEL definition, we get...
SYSTEM DEVELOPMENT
ANALYSIS OF PARCEL SEQUENCE AND ITERATIONS

TYPE PARCEL
ENTRY : ACQUISITION;
(X : COORDINATE,
Y : COORDINATE,
AREA : SURFACE,
SOIL-TYPE : {CLAY-LIKE, SANDY});
SEASON : SEASONS* FORGET AFTER 3 ITERATIONS;
EXIT : DISPOSAL
UNIQUE {{X, Y}}
END PARCEL

Other clauses indicating summary of information can be invented, e.g. SUM,MARIZE AFTER 3 ITERATIONS AS <expression>. The syntax of <expression> is not completely trivial because it will have to access attributes of subtypes of SEASONS and combine an existing summary with a new item of data.

At implementation level, an iteration chronicle can be implemented as a number of extra attributes, one for each iteration, and a summary attribute for the summary information -this happens currently for small chronicles of one or two iterations- or a linked list in a file for SEASONS with the most recent element as the head. Perhaps the sequence of attribute values should be represented as a linked list. Implementation will be briefly discussed in section 2.7.

Extending the definitions with this chronological information, and gathering the type definitions together, we get the following type definitions.
SYSTEM DEVELOPMENT
ANALYSIS OF PARCEL SEQUENCE AND ITERATIONS

TYPE ACQUISITION =
    GENERALIZATION OF {PROPERTY, RENTED, RECLAIMED}
END ACQUISITION

TYPE PROPERTY =
    PREVIOUS-OWNER : OWNERS,
    PRICE : MONEY
END PROPERTY

TYPE SEASONS =
    GENERALIZATION OF {CULTIVATIONS, FALLOW-_PERIODS}
END SEASONS

TYPE CULTIVATIONS =
    SOW : SEED;
    TREATMENT : TREATMENTS*
    REAP : VOLUME
END CULTIVATIONS

TYPE SEED =
    PRICE : MONEY,
    AMOUNT : AMOUNT-OF-SEED,
    MANUFACTURER : MANUFACTURERS,
    GENERALIZATION OF {POTATOES, WHEAT, SUGARBEET}
END SEED

TYPE WHEAT =
    SPECIALISATION OF {SEED},
    GENERALISATION OF {SUMMER-WHEAT, WINTER-WHEAT}
END WHEAT

TYPE WINTER-WHEAT =
    SPECIALISATION OF {WHEAT}
    GENERALISATION OF {NAUTICA, CITADEL, SAIGA, OKAPI, MARKSMAN}
END WINTER-WHEAT

TYPE TREATMENTS =
    GENERALISATION OF {SEED, PLOUGH, FERTILISE, WATER, SPRAY}
END TREATMENTS

The definition of specialisation structures contains no timing information. CULTIVATION is part of the SEASONS* iteration and will be forgotten or summarized as defined for SEASONS. All CULTIVATIONS in one remembered SEASONS iteration are retained.
All events in a PARCEL's life consist of instantiation of some attribute values. Once instantiated, these values don't change. Consequently, there are no behaviors to be defined for the types except the standard instantiation behavior.

2.6 NULL VALUES

There are several types of NULL values which have been mentioned in the above treatment. The "Not applicable" NULL value is eliminated by specialisation, while the "Does not exist" NULL value can be represented by the special value NONE. This is relevant for the Null selection (e.g. of TREATMENTS) and for zero iterations, as well as for non-existent attribute values (structured or unstructured). The "Unknown" NULL value can be represented by the value UNKNOWN.

The calculus for these values is too complicated to be worked out here.

2.7 IMPLEMENTATION

Following are some implementation considerations for the above structures.

The concept of surrogate translated conveniently in the concept of key. It is an open question whether the key associated with a surrogate should be a database key or a file key. If the second alternative is chosen, the concatenation of file name and key is unique over the database. This may be the better alternative, in view of the duplication of keys over different files in the implementation of specialisation hierarchies.

In Oracle, each tuple is implemented as a record with a ROWID field, consisting of <logical block number>.<row sequence number>.<partition number> (Oracle [1984]). The row sequence number is a sequence number within a logical block. The physical Oracle database consists of one or more partitions, each partition containing one or more files (called segments in Oracle) such that each file (segment) holds the records of one type (called table in Oracle). The concatenation of logical block number and row sequence number is unique within a partition and can be duplicated over different partitions, but since a partition can contain more than one file, it cannot be duplicated over different files within a partition. To be able to implement specialisations, such duplication should be allowed.
Only structured surrogates are implemented as keys. Unstructured surrogates are anonymous and are not distinguished by different keys. E.g. two instances of the INTEGER 3 are non-distinguishable, but two instances of EMPLOYEE are distinguishable by their key, even if they have identical attribute values. Each file is indexed by key and possibly by uniquely identifying description, if there is one.

Surrogates of unstructured types are simply stored in the attributes of which they are the value. An unstructured surrogate which is not the value of an attribute is not stored.

Structured surrogates of one type are stored in a file. Each type has such a file devoted to it. The records of the file consist of a field for the database key of the surrogate and fields for the attributes of the surrogate, and possibly others as well, e.g. specialisation attributes and date fields.

Unstructured attribute values of a structured surrogate of type T1 are simply stored in the file devoted to T1. If A is a structured attribute of instances of T1 and A has type T2, then in the file for T1 there is a reference to the file for T2, e.g. the field for A contains the database key of a structured surrogate in the file for T2.

Compositions are implemented by storing the database keys for the components in one file. If the composition is unstructured, there are no fields for attributes, if it is structured, there are fields for attributes.

An instance of a specialisation is implemented by spreading the attributes of the instance over one file for each type and supertype of the instance. The file for each type of the instance has a record with the database key of the instance. The supertypes of the leaf type which is instantiated contain a field with a reference to the file(s) for the types to which the supertype is specialised for this instance. Each specialisation group has such a field. This construction leads to problems with indexing on a database key, because the key occurs in several files. A key unique over the database is the combination (key, specialisation file1, ..., specialisation filen). This database key is of variable length N+1, where N is the number of specialisation groups of the type. A better solution is probably to make the key unique within the file for a type and index a record by type name (which is the name for the file containing the instances of the type) and key within the file.
If attribute A has a type T0 which is specialised to structured types T1, ..., Tn, then A's value will be a database key K which is an index in the file for T0. In that file there will at least be one field for each specialisation group of T0 indicating in the file for which subtype(s) K and its attributes are to be found. Figure 19 shows this situation for a specialisation of T0 with three specialisation groups G1, G2, and G3. The instance of T0 which is the value of A has the attributes and values as stored in the records with key K0 in the files Ti, Tj and Tk.

![Diagram of T0 specialisation groups and key values]

There is a duality between structured attribute values and specialisation. A record (K, A1, ..., An) will contain the key of another record as the value of Ai if Ai is a structured attribute. But if the record is specialised, then K itself will be a referential key to another file. Which file this is will be indicated by a specialisation field.
An iteration chronicle can be implemented by a linked list with the most recent event as its head. Only the record containing the most recent event is indexed by the database key, to find past events the list will have to be travelled. The list is stored in the file for the type and has a maximum length indicated by the FORGET AFTER N ITERATIONS clause. The date of each record is stored in an extra date field. We haven’t thought about the representation of the dates at which an entity was in a particular part of a sequence.

To implement the event pointer, each file for a structured type should have an extra field containing a reference to a location in the representation of the ESD for that type. We haven’t figured out yet how the ESD is to be represented.

The type definitions for PARCEL will be implemented by files with a record structure as shown in figure 20.
Each record has a field for a key and one field for each attribute. Below each attribute the type is written, if the type is analysed further. For example, ENTRY can have a key of the ACQUISITION file as
value. The ACQUISITION file maps this surrogate to one of the file names PROPERTY, RENTED, and RECLAIMED via its specialisation field. The possible specialisations are written below the specialisation field. The PROPERTY file, implementing a specialisation of ACQUISITION, contains a key of the ACQUISITION file and the attribute values of PREVIOUS-OWNER and PRICE. PREVIOUS-OWNER will probably be a referential key to the OWNERS file, which is not shown here. The SEASONS file is a file of linked lists. Each linked list consists of records with a field for the key, a specialisation field, and a field for the pointer to the previous record. The value of this field is indicated by an *.

The file structure could be optimised by skipping the ACQUISITION file and putting the value of the specialisation field of ACQUISITION directly in ENTRY. This can be done for every type which has as its only function to indicate which specialisation of one specialisation group is chosen.

The use of NULL values as field values still has to be investigated.

2.8 ANALYSIS OF DIAGNOSIS

The DIAGNOSIS process is a simple iteration of DIAGNOSES, which is reproduced, with attributes, in figure 21.

```
DIAGNOSIS

CONSULT

ARIO
SYNDROM
POSSIBLE-DISEASE-CAUSES
```

Figure 21
The complexity of the DIAGNOSIS process is hidden in the complexity of the structured values of SYNDROM, in the specialisation tree for DISEASE-CAUSES, and in the possibilistic assignment to POSSIBLE-DISEASE-CAUSES. The structure hierarchy for SYNDROMS is shown in figure 22 a-g and the DISEASE-CAUSES specialisation hierarchy in figure 23. These diagrams are extensions of similar diagrams in our second report. Only the underlined attributes shown in figure 22 a-g are implemented as frame slots. The others are used to generate advice for further observations.

Figure 22a

Figure 22b
ROOT-SYNDROM:
ROOT-SYNDROMS

ROOT-MYCELIUM-SYMP'ROM:
ROOT-MYCELIUM-SYMP'TOMS

COLOR SHAPE LENGTH

Figure 22c

FOOT-SYNDROM:
FOOT-SYNDROMS

PATCH:
PATCHES
COLOR
SPORES
COLOR
AROUND
LESIONS

STRIPE:
STRIPES
COLOR
AROUND
LESIONS

DISCOLORIZATION:
DISCOLORIZATIONS
COLOR AROUND LESIONS

SPOT:
SPOTS

Figure 22d

- 45 -
LEAF-SYNDROM:
LEAF-SYNDROMS

SPOT:
SPOTS

MYCELIUM-SYMPOM:
MYCELIUM-SYMPOMS

SPORE:
SPORES

BUG:
BUGS

COLOR

SPORE:
SPORES

COLOR SHAPE

PATTERN

BROKEN

COLOR SHAPE

SIPHONS

SHAPE

COLOR

SUPERFICIAL

SPORE:
BROKEN

SPORES

COLOR

Figure 22f
Figure 22g
DISEASE-CAUSES

BIOTIC

INSECT  NEMATODE  VIRUS  BACTERIA  FUNGUS

ABIOTIC

FOOD-DEFICIENCY

SITOBION NETOPOLOPHIUM AVENAE  DIRHODUM

Figure 23a
Figure 23b
Having analysed the internal structure of the DIAGNOSIS and PARCEL processes, we can now state integrity constraints over these structures. These constraints limit the possible values in POSSIBLE-DISEASE-CAUSES depending upon the observed values in the PARCEL structure. The constraints are not logical, as all the constraints analysed up till now are, but empirical. Even so, they are not deterministic but fuzzy: They do not limit the state of DIAGNOSIS uniquely. They are not probabilistic either, since they do not provide a probability distribution over the possible states of DIAGNOSIS. The rules are stated in the form of finite-state diagrams in chapter 4.

DIAGNOSIS can be viewed as a process which is instantiated once and then iterates indefinitely. Its definition will have to state how many iterations must be retained and what needs to be summarised of the iterations that are forgotten. Alternatively, keeping the fact in mind that users are interested in the CONSULTS of one PARCEL, the DIAGNOSIS process can be instantiated once per PARCEL and iterate indefinitely per PARCEL. Its surrogate then has a 1-1 coupling with a PARCEL surrogate and the DIAGNOSIS instantiation could be a side-effect of the PARCEL instantiation. The definition of the DIAGNOSIS function could then look like this:

```
TYPE DIAGNOSIS
  INSTANTIATED WITH PARCEL
    CONSULT : CONSULTS* FORGET AFTER 20 ITERATIONS
END DIAGNOSIS

TYPE CONSULTS
  SYNDROM : SYNDROMS,
  POSSIBLE-DISEASE-CAUSES : DISEASE-CAUSES
END CONSULTS
```

The PARCEL definition should be extended with a clause

```
INSTANTIATE DIAGNOSIS PER PARCEL
```

This can also be achieved by a separate specification of the dependencies of instantiations could be made. This specification should probably correspond to the Jackson SSD (cf. figure 6).

The implementation of a DIAGNOSIS process per PARCEL is as a simple file with a PARCEL surrogate as key and a linked list of CONSULTS per key.
2.9 DISCUSSION

The foregoing analysis is an extension of JSD with the semantic modelling techniques of instantiation, structuring, composition, specialisation, and chronicling. None of it is essentially new, but the combination of existing ideas in a coherent information system framework is. The source of ideas for the above are Jackson [1983], Smith & Smith [1977a and b], and the remarks of Date [1981] about surrogates.

Among the problems deserving investigation in the near future, in arbitrary order and varying priority levels, are:

1. Implementation of the event pointer;

2. Implementation of ESD's via behaviors and message passing mechanisms;

3. A suitable notation for surrogates, structured values, specialisations and compositions;

4. Formalisation of the instantiation, structure, composition, and specialisation hierarchies.

5. The relation of the structures used with normalisation theory and functional dependencies;

6. The use of temporal logic in integrity constraint checking;

7. The calculus of summary functions for chronicles;

8. The calculus of the different NULL values NONE, FUTURE and UNKNOWN and fuzzy attribute values.
3.1 SOURCES OF KNOWLEDGE

Knowledge represented in the information base consists of the individuals, the structure of the taxonomy, and the laws for existence and change of individuals. Sources for the knowledge to be represented in the system are phytopathological textbooks, phytopathologists, crop protection specialists, and farmers. We only used the first two sources. Phytopathologists study plant diseases at great detail. What can be gained from them is a logical analysis of the terms used to describe the domain. Crop protection specialists advice farmers in their area of the country on the identity of crop diseases and on the possible measures to be taken against the disease. What can be gained from them is a systematic observation discipline to be used in the user interface and an efficient application of the integrity constraints on the implementation level.

After a preliminary reading of textbooks (Butler & Jones [1949], Dekker [1984], Wiese [1977], Zadoks & Schein [1979]) we interviewed one phytopathologist (Zadoks) on symptomatology and one phytopathologist (Leemans) on one disease (septoria tritici). No particular interview technique was used except that we confined ourselves to septoria tritici and tried to understand everything what was said. The interviews took altogether roughly 6 hours per phytopathologist.

After we finished these exploratory interviews, we drew up a list of questions to be asked about each disease in the interviews with other phytopathologists. This list is shown in section 3.4.

At the same time as we conducted the exploratory interviews we scanned the disease description of septoria tritici in Wiese [1977] to draw up a list of about 60 potentially relevant terms for the diagnosis function. The initial list consisted of virtually every
verb and noun in the description. Following advice from a phytopathologist, it was reduced by eliminating terms referring to invisible entities and terms which would probably play no role in the formulation of the diagnosis rules. The rest of the list was ordered with respect to their specialisation and aggregation structures and a time-ordering was indicated where necessary. The result of this was the first crude data model for plant diseases, which after several implementation-abstraction cycles resulted in the model described in Wieringa & Curwiel [1986].

3.2 PROBLEMS OF KNOWLEDGE ACQUISITION

The following is a list of problems we encountered during knowledge acquisition. These problems all have to do with the nature of the domain. With each problem we also list the solution we chose for it.

1. MASS OF FACTS.
The sheer mass of facts and the heap of specialist terminology in a book like Butler & Jones [1949], which spans almost 1000 pages, is appaling for computer scientists trying to build a model in much less than the time needed to train a phytopathologist. This mass of domain knowledge is much greater than the amount of knowledge which has to be acquired about an organization when an information system is to be built, and it was correspondingly more difficult to separate the relevant from the irrelevant data. Obviously, one should not to try to become a specialist, but then the system developer will at least have to understand what the specialists are talking about. In order to know what kind of system must be built, some knowledge of phytopathology must be gathered.

The solution we chose was to limit ourselves to two diseases as long as possible and try to find out what kind of system we wanted to build for this disease and what kind of questions we wanted to ask the phytopathologists. This solution worked to some degree, though a 2 page description of one of the two diseases yielded still a list of nearly 100 specialist terms of which it was difficult to separate the relevant from the irrelevant. Secondly, system structure continued to evolve after we interviewed other experts on other diseases, making different types of knowledge relevant and creating a need for other types of interviews. At the moment of writing, the development of the model structure still continues.

- 54 -
Another way to reduce the amount of facts we needed to know something about was to eliminate everything which is not visible to the unaided eye and everything which would not be needed for the statement of a diagnosis.

Sometimes phytopathologists referred us to the textbooks were we could simply read what they told us. The problem, to repeat, with this is that for laymen like us it is very hard to separate the relevant from the irrelevant information.

2. DIFFERENT TYPES OF KNOWLEDGE
The knowledge of the scientists is very different from that of the extension officers. It turned out that apart from general decision criteria, scientific knowledge is too detailed to be of much use in a diagnosis system to be used by farmers. On the other hand, the knowledge of the extension officers would be very useful in the design of a disciplined input dialogue and a structured report on the possible disease causes and observations. We did not follow up these new ideas about the uses of the different types of knowledge, because the extension officers were too much pressed for time to ask more than one afternoon to interview them. The current project is still exploratory and all we can do at this point is to collect ideas and arguments for a follow-up project.

3. COMPLEX INFRASTRUCTURE
In contrast to organizational information systems, the domain does not consist of actions carried out by the future users but of entities carrying out invisible actions, like plants, pathogens, and parcels. Knowledge about the domain is not gathered by interviewing users but by asking specialists, who themselves gather it not by asking the domain entities but by doing scientific experiments (which is a way to query nature). This may be the reason why the user interface of expert systems has been neglected. The specialists involved in it were too often not the future users.

The specialists themselves are divided into two groups, phytopathologists who work in laboratoria, and crop protection specialists who work as extension officers in the field to advice farmers. The extension officers are organized regionally, which makes sense since different regions have different disease patterns and also different dialects in which the diseases are described.
All of this makes for a complex infrastructure in which a disease diagnosis system is to function. We solved this problem mainly by avoiding it. The current system is a prototype built to investigate the feasibility of the expert system approach in the area of plant disease diagnosis. User requirements, infrastructure and political interests of different groups of people were not taken into consideration, as they would have to be for a system which would really be taken into production.

4. TAXONOMY.
The taxonomy of plant diseases and pathogens carries some problems for the system developer, among which are the following.

1. The taxonomy is not (yet?) standardized;
2. It evolves over time, as new insights develop;
3. It is made according to non-uniform criteria;
4. The placement of individuals in the taxonomy evolves over time.

The taxonomy of symptoms in Wiese [1977], for example, uses temporal (growth stage) and topological (location on plant) criteria on one level of the taxonomy. Each criterion is used non-exhaustively but together they apparently cover all (important?) cases.

We solved these problems by simply choosing a taxonomy and allow for variations in terminology by adding in attributes to hold alternative terms. The criteria we chose were uniform at each level, and higher level changes were ignored. These could be added in a more complex system. New knowledge about the placement of individuals in the taxonomy should be communicated to the system by embedding it in the proper procedures. The last two points bring with it the problem that knowledge becomes dated but should still be remembered because it has been used in previous diagnoses, which are remembered as well. Second order changes can be represented by introducing an instantiation hierarchy where higher levels have the same type structure as lower levels. Some preliminary ideas on this were presented in Wieringa & Curwiel [1986].
5. LAWS

The laws of existence and change for the domain are not yet known with sufficient certainty and may be inherently vague. Probabilities of the occurrence of diseases in different circumstances are not known, and if they are known, they are known to change. The concepts by which the observations are described are fuzzy (e.g. "watery green spots").

The solution we chose for the problem of fuzziness is to place the boundary between explicitness and intuition where it belongs: In the human user. A full-blown system would have to use photos of the symptoms to be recognized by the user. Since probabilities are largely unknown, we limit ourselves to possible diseases by simply ruling out what is impossible. This is not as easy as it looks, because basically everything remains possible after the observations are in. The user should therefore be advised to use the advice to his best judgement, without assuming that the advice is correct. This solution is external to the software and concerns user training.

6. EXISTENCE OF INDIVIDUALS

Most of the individual entities relevant to the diagnosis are not observable or, when they are, have simply not been observed. The observations that are made, even if they do not involve fuzzy concepts, are unreliable because they are made by non-specialist users.

The solution for this is once again to use photos, train the user, and also to produce an intelligent report to advise the user on observations he should make to find out the identity of the disease.

It turned out that the main problem of disease diagnosis is actually not the uncovering of decision criteria, but the correct observation of symptoms. This is a decision problem in itself which in our opinion is nearly intractable in terms of computer programming techniques. The solution to this lies not in automation but in education.

Another problem inherent in the domain is that commonly more than one disease is present and that therefore symptoms of different diseases occur at the same time. This requires the observer to distinguish relevant from irrelevant symptoms.

At the end of section 3.4 on the interviews with crop protection
specialists, we will list some problems of crop disease knowledge and propose a solution to them.

3.3 INTERVIEWS WITH SCIENTISTS

The scientists which we interviewed are phytopathologists who have detailed knowledge of a few diseases. Per interview we concentrated on one disease. Per disease we had the following goals:

1. To describe the disease at the level of abstraction of the model;
2. To write down decision criteria which can distinguish the disease from other diseases in a development stage which is as early as possible;

The questions are clustered around one goal.

1. NOMENCLATURE
The goal of these questions is to determine what we are talking about and what it is called.

1. What is the systematic Dutch, English and American name of the disease.
2. Is there a different name for the disease (a relation between plant and pathogen) and for the disease cause (a pathogen) or do these have the same name?
3. Does the pathogen have different names in different stages of its life.
4. Are there different diseases which are commonly confused with the disease we are talking about.

2. LIFE CYCLE OF THE DISEASE
This cluster of questions should give more meaning to the terminology by describing the context in which the events take place.

1. By what means does the disease cause enter the crop (wind, rain, soil, volunteer wheat, etc.).
2. How does the disease cause spread over the plant;  
3. and to neighbouring plants.  
4. Give a general description of what the crop looks like when the disease is present  
5. and of what happens when nothing is done to stop it.  

3. ENVIRONMENTAL FACTORS  
The goal of these questions is to determine the environmental factors which are favourable and unfavourable to the disease. Whatever is known by the system about these factors can be used to support the diagnosis, though they are too uncertain to make a decision.  

1. What is the susceptibility of different wheat varieties for the disease.  
2. At which growth stages is wheat susceptible.  
3. What are the incubation and latency periods (the period from infection to visibility and the period from infection to potential infectiousness).  
4. What kind of soil is favorable to the disease.  
5. What is the influence of different fertilizers, insecticides etc. on the disease.  
6. What is the influence of the occurrence of other diseases on the occurrence of this disease.  
7. What is the influence of plant density.  
8. Are there other relevant factors, e.g. crop rotation, chemicals applied in previous seasons, diseases in neighbouring parcels.  

4. SYMPTOMS  
The goal of these questions is to get a list of attributes and possible values for each symptoms, together with constraints on their occurrence. It is at this point that most of the complexity, uncertainty and fuzziness of the events is to be found.
1. What is the life cycle of the symptom.
2. What is the dependence of the life cycle of the symptom on the life cycle of the plant, i.e. does the disease have different manifestations in different life stages of the plant.
3. On which plant parts are the symptoms located.
4. Are there symptoms on field level, i.e. visible as patterns in the field.
5. What is the form, color, size, etc. of the symptoms.

5. DECISION CRITERIA
The goal is to find criteria which distinguish this disease from other diseases. For each positive criterion we should be able to say that if it isn't observed, the disease is absent. The problem at this point is that there are hardly such definite criteria.

1. What are the symptoms which make it very likely that the disease is present.
2. What are the symptoms which, if present, would support the conclusion that the disease is present but if absent, would not invalidate the conclusion.
3. Which observations are incompatible with occurrence of the disease.

From these interviews we destilled the decision graphs shown in chapter 4.

3.4 INTERVIEWS WITH CROP PROTECTION SPECIALISTS
Crop protection specialists are agricultural extension officers who advice farmers on difficult disease cases. They speak the language of the farmer and know the peculiarities of their region of the country and are therefore an important source of knowledge on the user interface.
We interviewed three specialists in the North, South-West and South-East of the country, respectively. Each specialist was shown 36 colour slides of diseased crops and asked to comment upon the slide. Altogether, the slides covered the 12 most important diseases in winter wheat in the Netherlands. The interview with each specialist lasted about 4 hours.

The slides were selected by the department of phytopathology following criteria which are derived from Grover [1983]. For each disease we asked for slides which would show

1. a common appearance of the disease;
2. a rare appearance of the disease;
3. a case of heavy damage by the disease;
4. appearances of the type usually shown in textbooks;
5. a case which is easy to diagnose.

Table 2 shows the diseases selected for the interviews, the relative incidence of the disease in the Netherlands (percentage of parcels in a sample affected by it, Stol [1985]), and the number of slides of the disease shown to each specialist.

<table>
<thead>
<tr>
<th>vernacular</th>
<th>systematic name</th>
<th>incidence</th>
<th># slides</th>
</tr>
</thead>
<tbody>
<tr>
<td>yellow rust</td>
<td>puccinia striiformis</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>brown rust</td>
<td>puccinia recondita</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>black rust</td>
<td>puccinia graminis</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>mildew</td>
<td>erysphes graminis</td>
<td>90</td>
<td>3</td>
</tr>
<tr>
<td>leaf spot</td>
<td>septoria tritici</td>
<td>92</td>
<td>5</td>
</tr>
<tr>
<td>glume blotch</td>
<td>septoria nodorum</td>
<td>93</td>
<td>2</td>
</tr>
<tr>
<td>eye spot</td>
<td>pseudocercosporrella</td>
<td>74</td>
<td>4</td>
</tr>
<tr>
<td>take-all</td>
<td>gaumannomyces graminis</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>speckled snow</td>
<td>griphosphaerella</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>mold</td>
<td>nivalis</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>nitrogen deficiency</td>
<td>claviceps purpurea</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>black heady molds</td>
<td>fusarium colmorum</td>
<td>45</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2
For most diseases, there were photographs taken from each of four distances, a few centimeters, one meter, a couple of meters, and an overview over the parcel. The slides were sorted according to distance and in arbitrary order per distance.

At the beginning of the interview we made clear that we were not going to test the knowledge of the specialist but that we would like to know how they described the symptoms in their own language. For each slide we asked the following questions.

1. Describe what you see;
2. Which diseases could be present;
3. What is the evidence for each of these diseases;
4. What further evidence would be needed to make the decision.

In this way we hoped to find clues for the structure of a disciplined dialogue to collect symptoms. It turned out that the diseases were very difficult to recognize from the slides. One reason for this is that the slides were old and sometimes had faded colors. A second reason is that they showed diseases in stages that were too late to do anything about it anymore. Specialists are never confronted with such cases. The most important reason, however, is that more information than that provided by a single slide was needed, such as general patterns in an overview of the parcel, the condition of plant parts not visible on the slide, the structure of the roots, soil conditions, previous treatments and diseases, etc. Some information is not visual at all, e.g. whether the spots can be rubbed off the leaves, etc. Which of these observations are needed, depends upon the first observations made. It is in the dependence of the need for further observations on earlier observations that the intelligence of the system must lie. In other words, we have been trying to find the wrong decision criteria: Not the dependence of disease hypotheses on observations is important, but the dependence of later observations on earlier observations. Once the relevant observations are known, the specialist judges the case as "typically disease X" or "most likely disease X." There is hardly an inference chain from symptoms to diseases.

This situation is very different from that of medical diagnosis, where a number of standard laboratory tests can be expected to have been done before the system is consulted and there is an inference of several steps from data to hypothesis. In medical diagnosis, the search space for all possible human diseases is quite large, while in plant disease diagnosis it is very flat but contains many points which
are difficult to define it at all.

The specialist can usually give the reasons why he decided for a particular disease after a successful diagnosis, but is unable to give, in advance of the diagnosis, an exhaustive list of observations he will make. Typically, he would look at a slide and say "It's disease X" or "I can't see what it is from this slide alone." At no time he would describe the symptoms in advance of a diagnosis except after repeated questions from us, and then only in general language like that found in information booklets. The verbal symptom descriptions would usually leave open the question why these general descriptions apply to this particular case: Why did he call these spots yellow when according to us they were definitively orange? And why were the spots on the next slide brown, since to us they were just as orange as the previous slide?

From these interviews it emerged that the most that can be done by the system is to request some general observations which the farmer can be expected to have made when he consults the system, and to produce a report which in a compact and effective way advises the farmer to make certain further observations if he wants to decide between diseases X, Y and Z. This puts the burden of the observation discipline on the farmer and should be combined with a large set of photographs of all diseases in an early stage of development. The experience of the crop protection specialists should be used to design an effective observation discipline that should be implemented in the dialogue an in the report. Some of the material to design such a discipline is already gathered by our interviews.

This way, the main function of the system is educational. This is in agreement with an evaluation of farmer's experiences with EPIPRE, a computer program which predicts the development of diseases and advises whether or not to spray against the disease. These farmers valued the observation discipline enforced by EPIPRE more than the spraying advice (Blokker [1984]). Also, this proposal uses the computer where its real power lies: in the processing of large amounts of data fast. The use of the system would not lie in its knowledge of one particular disease but in its function as an encyclopedia of crop diseases. To be useful the system would have to be embedded in an infrastructure which takes care of regular updating.

The user interface described in chapter 5 is a first version of such a system. In chapter 7 a proposal for the development a crop management support system will be made.
CHAPTER 4

INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES

4.1 INTRODUCTION

In this chapter all the diseases are described, the information has been obtained by means of interview with the specific experts. The information has been organized as follows:

1. Name of the disease
   1.1 Systematic names
      1.1.1 Vernacular names
      1.1.2 Names of the diseases causes
      1.1.3 Related diseases
   1.2 Disease cycle
      1.2.1 Sources of primary inoculum
      1.2.2 Distribution of the disease over the plant
      1.2.3 Infection of other plants
      1.2.4 General description of the symptoms
      1.2.5 What happens if the disease is not controlled
   1.3 Environmental factors
      1.3.1 Resistance of different wheat varieties
      1.3.2 Growthstages most susceptible to infection
      1.3.3 Soil type
      1.3.4 Weather
      1.3.5 Fertilization
      1.3.6 Plant density
      1.3.7 Controls
   1.4 Relevant criteria
4.2 SITOBION AVENAE

4.2.1 Systematic Names

4.2.1.1 Vernacular Names -

The two kinds of aphids that are of interest for their effect on wheat are described here together, because the symptoms and the effects of both kinds are practically the same.

Dutch name: Grote graanluis
Anglosaxian name: English grain aphid

Roosgrasluis
Rosegrass Aphid

4.2.1.2 Names Of The Disease Causes -

Sitobion Avenae
Metopolophium Dirhodum

4.2.1.3 Related Diseases -

not of importance

4.2.2 Disease Cycle

4.2.2.1 Sources Of Primary Inoculum -

Perhaps inoculum is not the correct phrase to use because the aphids fly to the crops.

1. The grain aphid from the grasses.
2. The rosegrass aphid from the rose plants.

4.2.2.2 Distribution Of The Disease Over The Plant -

The aphids distribute themselves over the plant by means of walking. On wheat we only find the asexual form of the insects, but this form can reproduce itself. So aphids reproduce parthenogenetically, and as many as 20 generations can be completed in 1 year. The sexual form we
find on different hosts.

4.2.2.3 Infection Of Other Plants -

During their stay on the wheat the aphids are not able to fly. They fall off the plants and walk over the ground to other plants.

4.2.2.4 General Description Of The Symptoms -

The little green creatures can be spotted before we can see the damage done to the plants. Their size is about 0.5 cm. One should look at the whole plant.

1. The rosegrass aphid is to found on the stem, its color is darkgreen and the shape is squat. It has 2 red dots and darkgreen syphons.

2. The grain aphid is to be found on the head, its color ranges from green to brown and the shape is oblong. The syphons are black.

The latter of the mentioned aphids causes the most damage.

4.2.2.5 What Happens If The Disease Is Not Controlled? -

The aphids accomplish their feeding by bi-directional pumping mechanisms that withdraw sap with nutrients. Extensive feeding induces plants to lose color and make slow growth. The excretion products of the aphids make the crop dirty and these products also create an environment for other parasites.

4.2.3 Environmental Factors
4.2.3.1 Resistance Of Different Wheat Varieties -
There are no differences in susceptibility for the different wheat varieties.

4.2.3.2 Growthstages Most Susceptible To Infection -
From the growthstages DC50-60 to DC 70 the plants are susceptible.

4.2.3.3 Soil Type -
not of importance

4.2.3.4 Weather -
Populations of aphids are favored by mild winters and cool to moderate spring and summer temperatures.

4.2.3.5 Fertilization -
Fertilization supplies the plants with more nutrients, so the amount of excretion products per unit of aphids will decrease.

4.2.3.6 Plant Density -
Not of importance

4.2.3.7 Controls -
The pesticides Pyrimor and Pyrimicarp just kill the aphids and do not interfere with the natural enemies (predators).
4.2.4 Relevant Criteria

1: Positive criteria

- The little green creatures have to be present, after DC75 they are not to be found anymore and one just finds the symptoms of the damage done.
4.2.5 Decision Tree

\[
\begin{array}{ccc}
0 & \text{DISEASE-OCURRENCE} & \text{PLANT} \\
& \text{LEAF} & \text{BUGS PRESENT} \\
\uparrow & V & V \\
\downarrow & NO & YES \\
\downarrow & - & - \\
\downarrow & 1 \text{ BUGS.COLOR IN (GREEN, DARKGREEN, BROWN)} & V \\
\downarrow & NO & YES \\
\downarrow & 2 \text{ BUGS.SHAPE = SQUAT} & - \\
\downarrow & NO & YES \\
\downarrow & 3 \text{ BUGS.SIPHONS = BLACK} & - \\
\downarrow & NO & YES \\
\downarrow & 4 \text{ DISEASE-OCURRENCE.DISEASE-CAUSE} & - \\
\end{array}
\]

SITOBION AVENAE POSSIBLE
4.3 PSEUDOCERCOSPORELLA HERPOTRICHIOIDES

4.3.1 Systematic Names

4.3.1.1 Vernacular Names -

Dutch name: Oogvlekkenziekte
Anglosaxian name: Eyespot/Strawbreaker

4.3.1.2 Names Of The Disease Causes -

Pseudocercosporella herpotrichoides

4.3.1.3 Related Diseases -

Corticium solani/Sharp eyespot

4.3.2 Disease Cycle

4.3.2.1 Sources Of Primary Inoculum -

1. Water-splashed conidia
2. Mycelium persisting on host debris

4.3.2.2 Distribution Of The Disease Over The Plant -

Symptoms do not appear on roots and rarely develop more than 15 to 20 cm above soil level. Infected plants bear lesions that begin superficially on leaf sheaths then progress laterally and to within the culm.
4.3.2.3 Infection Of Other Plants -

Conidia are distributed principally by splashing rain and have a dispersal radius of 1 to 2 m. Wheat coleoptiles and leaf sheaths are penetrated directly or through stomata near ground level and secondary conidia form on new lesions within 4 to 12 wk. The new inoculum, like secondary inoculum from alternative grassy hosts appears inconsequential to the current epidemic.

4.3.2.4 General Description Of The Symptoms -

The elliptical or "eye" shape of the lesions is diagnostic. The lesions are distinct, white to tan-brown initially, and oriented longitudinally with the stem. They may eventually 'girdle' the culm, develop fungus-darkened centers and increase 4 cm in length. Sometimes a diagnostic weft of mycelium appears within the lumen of the culm. Diseased plants tend to mature early and produce white heads with incompletely filled seed. Such heads frequently support 'sooty' molds.

The lesions weaken the stem so that diseased tillers begin to fall randomly. when the disease is moderate, many damaged plants will be supported by their healthy neighbors. In severe cases, large areas of lodged plants occur. Lodging is nondirectional and lodged plants have no capacity to recover as opposed to wind-lodged plants.

4.3.2.5 What Happens If The Disease Is Not Controlled ? -

The disease normally increases in prevalence where wheat is repeatedly grown. Eyespot may kill individual tillers or plants outright. More frequently, it reduces kernel size and number, causes culms to lodge and renders plants difficult to harvest.

4.3.3 Environmental Factors
4.3.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "61e Beschrijvende Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance.

<table>
<thead>
<tr>
<th>Crop variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>?</td>
</tr>
<tr>
<td>Citadel</td>
<td>5</td>
</tr>
<tr>
<td>Saiga</td>
<td>6</td>
</tr>
<tr>
<td>Okapi</td>
<td>7.5</td>
</tr>
<tr>
<td>Arminda</td>
<td>7</td>
</tr>
<tr>
<td>Marksman</td>
<td>6</td>
</tr>
<tr>
<td>Granada</td>
<td>7</td>
</tr>
</tbody>
</table>

4.3.3.2 Growthstages Most Susceptible To Infection -

Primary sporulation occurs in spring up until the middle of april. This primary sporulation causes the infections and is of importance to the condition of the crop.

4.3.3.3 Soil Type -

Humid soils favor eyespot, so badly drained and claylike soils are favorable.

4.3.3.4 Weather -

Eyespot is favored by high soil moisture and a high humidity near soil level. Mild winters and cool springs prolong sporulation and infection periods. Conidial production is maximum when temperatures fluctuate near 10 C; it does not occur below 0 or above 20 C. Infection can occur within 15 min between 6 and 15 C in a water-saturated atmosphere but is dramatically slowed or prevented above 16 C.
4.3.3.5 Fertilization -

Plants may be predisposed to infection in spring by nitrogen fertilization.

4.3.3.6 Plant Density -

A dense crop creates an environment with a high humidity, so it favors Eyespot.

4.3.3.7 Controls -

Spring wheat and late-planted winter wheat is less exposed to infection. Thin seeded fields limit disease by decreasing relative humidity within the crop. Since P. herpotrichoides dies out with the decay of infested residues, crop rotations in which susceptible cereals are not grown for one or more years are advised. Resistant cultivars are not available but some, because of stiffer straw, are tolerant.

Chemicals like CCC [(2-chloroethyl) trimethylammonium chloride] and benomyl reduce or prevent eyespot damage. The former strengthens straw and the latter limits infection.

4.3.4 Relevant Criteria

1: Positive criteria

- The spots on the leaves have to be present.

2: Supporting criteria

- Clay soils create humid conditions, which favor eyespot.
- Wet spring favors eyespot.
4.3.5 Decision Tree

0 DISEASE- OCCURRENCE, SYMPTOMS | PLANT | FOOT | SPOTS PRESENT

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td>V</td>
</tr>
</tbody>
</table>

1 SPOTS. BORDERS ARE WELL-DEFINED

<table>
<thead>
<tr>
<th>YES</th>
<th>NO</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 SPOTS. MYCELIUM IS PRESENT

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 SPOTS. MYCELIUM. COLOR = BLACK

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 DISEASE- OCCURRENCE, DISEASE- CAUSE | PSEUDOCERCOSPORELLA- HERPOTRICHIOIDES POSSIBLE
4.4 CORTICICUM SOLANI

4.4.1 Systematic Names

4.4.1.1 Vernacular Names -

Dutch name: Scherpe Oogvlekkenziekte
Anglosaxian name: Sharp Eyespot/Root rot

4.4.1.2 Names Of The Disease Causes -

Corticium solani

4.4.1.3 Related Diseases -

Pseudocercosporaella herpotrichoides/Eyespot

4.4.2 Disease Cycle

4.4.2.1 Sources Of Primary Inoculum -

1. Mycelium in host and soil debris.

2. Hard, usually darkened and rounded mass of dormant hyphae (sclerotia) also in host and soil debris.

4.4.2.2 Distribution Of The Disease Over The Plant -

Some pathogens infect wheat culms and others roots, but all infections start at the roots, distinct patches of lodged or white-headed plants may result. Seedlings are occasionally killed outright, but most plant tolerate extensive root browning and outpace the disease by producing new roots.
4.4.2.3 Infection Of Other Plants -

Root infections can occur anytime during the growing season. Normally caused by infectious hyphae. C. solani also is an important opportunist pathogen in roots injured (predisposed) by nematodes. Infections that occur after wheat is headed are rarely consequential.

Culm infections apparently are initiated from unorganized mycelium and from sclerotia.

4.4.2.4 General Description Of The Symptoms -

Sharp eyespot is named for the conspicuous lesions that develop on lower leaf sheaths. They resemble lesions caused by Pseudocercosporella herpotrichoides but are more superficial and more sharply delineated. Their margins are deep brown and their centers are pale or straw-colored and frequently studded with mycelium. The mycelium is dark, superficial and sclerotical and easily removed by rubbing.

Plants with their roots attacked may form 'purple' patches since their leaves are stiff and dull blue-gray. Maturity is delayed rather than hastened as with culm infections.

4.4.2.5 What Happens If The Disease Is Not Controlled? -

Wheat is damaged most by seedling infections, the infected spring-sown wheat may lodge or ripen prematurely.

4.4.3 Environmental Factors

4.4.3.1 Resistance Of Different Wheat Varieties -

no information available

4.4.3.2 Growthstages Most Susceptible To Infection -

All growthstages can be infected, but seedling infections cause the most important damage.
4.4.3.3 Soil Type –
Acid, sandy and dry (<20% moisture holding capacity) soils increase disease risk.

4.4.3.4 Weather –
Sharp eyespot is favored by cool spring temperatures.

4.4.3.5 Fertilization –
Fertilization promoting root growth is advantageous for the plants.

4.4.3.6 Plant Density –
no information available.

4.4.3.7 Controls –
Truly resistant cultivars are not available and concentrated efforts to develop them are rare. Crop rotation is of marginal value because of the broad host range of Corticium solani and its capacity to survive in the soil. Some antagonistic bacteria, _STREPTOMYCES_ and _BACILLUS_ species, provide limited biological control. Pesticides are warranted and economical when pathogens in addition to C. solani can be controlled. Some, like benomyl, that curb eyespot may accentuate sharp eyespot.

4.4.4 Relevant Criteria
1: Positive criteria

- The spots, superficial and sharply delineated, are present on the stem, or

- 77 -
- Runner hyphae on the roots are present.

2: Supporting criteria

- Acid, sandy and dry soils increase disease risk.
- Dry cool springs favor eyespot.
4.4.5 Decision Tree

```
0 DISEASE-OCURRENCE | SYMPTOMS | PLANT | FOOT | SPOTS PRESENT
  | V
  | NO     | YES   | UNKNOWN
  | V      | V     | V

1 SPOTS. BORDERS ARE WELL-DEFINED
  | V
  | NO     | YES   | UNKNOWN
  | V      | V     | V

2 SPOTS. MYCELIUM IS PRESENT
  | V
  | NO     | YES   | UNKNOWN
  | V      | V     | V

3 SPOTS. MYCELIUM. COLOR = BLACK, PURPLE
  | V
  | NO     | YES   | UNKNOWN
  | V      | V     | V

4 DISEASE-OCURRENCE | DISEASE-CAUSE
  | V
  | CORTICICUM-SOLANI POSSIBLE
```
4.5 Erysiphe graminis

4.5.1 Systematic Names

4.5.1.1 Vernacular Names -

Dutch name: Meeldauw
Anglosaxian name: Powdery mildew

4.5.1.2 Names Of The Disease Causes -

Erysiphe graminis f. sp. tritici
(pathogenic on wheat)

4.5.1.3 Related Diseases -

Erysiphe graminis f. sp. hordei
(pathogenic on barley)

4.5.2 Disease Cycle

4.5.2.1 Sources Of Primary Inoculum -

1. Wind-borne asco-spores or conidia

2. Volunteer wheat plants are opportune hosts and support the pathogen between summer and autumn-sown crops.

4.5.2.2 Distribution Of The Disease Over The Plant -

The germ tubes of both ascospores and conidia penetrate wheat directly and give rise to superficial sporulating colonies. Resultant conidia are wind-dispersed and induce secondary infections.
4.5.2.3 Infection Of Other Plants -

Conidia produced in great numbers, are most important epidemiologically. They survive only a few days but withstand dispersal over several kilometers. They germinate over a wide temperature range (1-30°C) and without free moisture. Some germinate utilizing only endogenous moisture. In favorable field environments, germination, infection and secondary sporulation are completed within 7 to 10 days.

4.5.2.4 General Description Of The Symptoms -

Erysiphe graminis f. sp. tritici may infect all aerial portions of the plant, but it is usually most prevalent on the upper surface of lower leaves. The fungus is entirely superficial. You can see the fungus as effuse patches (colonies) of cottony mycelium on the host surface. Colonies initially white and sporulating, later turn dull gray-brown. Sexual fruiting bodies are visible without magnification as distinct brown-black dots within aging colonies on maturing plants.

4.5.2.5 What Happens If The Disease Is Not Controlled ? -

Erysiphe graminis f. sp. tritici utilizes the nutrients, reduces the photosynthesis and increases the respiration and transpiration of its hosts. Infected plants lose vigor and their growth, heading and seed filling are impaired. Heavily infected leaves and even entire plants can be killed prematurely. Losses up to 40% are recorded and are greatest when plants are infected as seedlings and disease development continues through flowering.

4.5.3 Environmental Factors

4.5.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "61e Beschrijvend Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance

<table>
<thead>
<tr>
<th>Wheat variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>?</td>
</tr>
<tr>
<td>Citadel</td>
<td>6.5</td>
</tr>
</tbody>
</table>

- 81 -
INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES

ERYSIPHE GRAMINIS

<table>
<thead>
<tr>
<th>Cultivar</th>
<th>Susceptibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saiga</td>
<td>6.5</td>
</tr>
<tr>
<td>Okapi</td>
<td>5</td>
</tr>
<tr>
<td>Arinda</td>
<td>6</td>
</tr>
<tr>
<td>Marksman</td>
<td>8</td>
</tr>
<tr>
<td>Granada</td>
<td>6.5</td>
</tr>
</tbody>
</table>

4.5.3.2 Growthstages Most Susceptible To Infection -
Symptoms can appear anytime after seedlings emerge

4.5.3.3 Soil Type -
Dry sandy soils favor Erysiphe graminis f. sp. tritici.

4.5.3.4 Weather -
Powdery mildew development is optimal between 15 and 22 C and is markedly retarded above 25 C. Most stages of infection proceed in darkness, except for host penetration and conidial formation that require light. High humidity and cool temperatures favor disease development.

4.5.3.5 Fertilization -
Wheat is most susceptible during periods of rapid growth. Heavy nitrogen fertilization favors disease development. The pathogen is less virulent when nitrogen, potassium and phosphorus fertilization is correctly proportioned.

4.5.3.6 Plant Density -
Dense stands (high humidity) of susceptible cultivars favor disease development.
4.5.3.7 Controls -

Fungicides are used to control powdery mildew where costs are not prohibitive. In parts of Europe, selective, systemic (transported by the plant) fungicides are often employed, for instance: Bayfidan, Tilt, Corbel. Resistant cultivars are perhaps the best defence against powdery mildew, and they are useful so long as the prevalent races of the fungus are combated.

Crop rotations, clean cultivation (especially the destruction of volunteer wheat) and destruction of host residues reduces overwintering inoculum.

4.5.4 Relevant Criteria

1: Positive criteria

- The presence of white cottony mycelium on the host surface

2: Supporting criteria

- Black fruiting bodies within aging colonies on maturing plants.
- Dense crop favors the disease
4.5.5 Decision Tree

0 DISEASE-OCURRENCE | SYMPTOMS | PLANT | LEAF | MYCELIUM PRESENT

| NO | YES | UNKNOWN |

1 MYCELIUM.SHAPE = PATCHES

| NO | YES | UNKNOWN |

2 MYCELIUM.COLOR = WHITE

| YES | UNKNOWN | NO |

3 MYCELIUM.COLOR IN (BROWN, BLACK)

4 MYCELIUM.SPORES ARE PRESENT

5 MYCELIUM.SPORES IN (BROWN, BLACK)

- 84 -
### INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES

**ERYSIPHE GRAMINIS**

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

6 DISEASE-OCURRENCE.DISEASE-CAUSE
ERYSIPHE-GRAMINIS POSSIBLE
4.6 FUSARIUM CULMORUM

4.6.1 Systematic Names

4.6.1.1 Vernacular Names -

Dutch name: Kafjesrood/Fusarium in de aar
Anglosaxian name: Scab/Head blight/White heads

4.6.1.2 Names Of The Disease Causes -

Fusarium culmorum

4.6.1.3 Related Diseases -

Fusarium avenaceum
Fusarium graminearum
fusarium nivale

4.6.2 Disease Cycle

4.6.2.1 Sources Of Primary Inoculum -

1. Wind-borne asco-spores or conidia

2. Pathogens survive as mycelium, conidia, ascospores and perithecia on infested cereal and grass residues in soil.

4.6.2.2 Distribution Of The Disease Over The Plant -

Scab occurs during moist, warm weather after air-borne conidia or ascospores contact wheat heads. The spores germinate and indiscriminately invade flower parts, glumes or other portions of the spike.
4.6.2.3 Infection Of Other Plants -
Secondary infections result from air-borne conidia which can be
dispersed over long distances. Ascospores normally are produced too
late to function as secondary inoculum but persist in host residues
and contaminate seed.

4.6.2.4 General Description Of The Symptoms -
Scab is best recognized on emerged immature heads where one or more
spikelets or the entire head appears prematurely bleached. If the
axis of the head is infected, all tissues above that point are faded.
Small dark spots (perithecia) and superficial pink or orange mycelium
and spore masses may be seen on, and especially at the base of,
diseased spikelets. Bleached spikelets usually are sterile or contain
only partially filled seed. The causal fungi are facultative
nonspecific parasites that may also infect other plant parts. From
growthstage DC30 we can see little brown stripes at the basis of the
stem. These stripes can develop into a complete discolorisation of
the stem. Bread made from scabby wheat has been described as
intoxicating.

4.6.2.5 What Happens If The Disease Is Not Controlled ? -
Significant yield losses result from floret sterility and poor seed
filling.

4.6.3 Environmental Factors
4.6.3.1 Resistance Of Different Wheat Varieties -
The figures are taken from the "61e Beschrijvend Rassenlijst voor
Landbouwgewassen 1986". A high figure stands for a high resistance

<table>
<thead>
<tr>
<th>Wheat variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>?</td>
</tr>
<tr>
<td>Citadel</td>
<td>5.5</td>
</tr>
<tr>
<td>Saiga</td>
<td>6.5</td>
</tr>
<tr>
<td>Okapi</td>
<td>6.5</td>
</tr>
<tr>
<td>Arminda</td>
<td>6.5</td>
</tr>
</tbody>
</table>

- 87 -
4.6.3.2 Growthstages Most Susceptible To or Infection -

Infections of the head are most frequent and most serious at the growth stage at which wheat flowers open and shed pollen. Blight symptoms develop within 3 days after infection when temperatures range between 25 and 30°C and moisture is continuous.

4.6.3.3 Soil Type -

Dry sandy and peat-soils are favorable

4.6.3.4 Weather -

Humid conditions are important for germination, infection and sporulation. Moisture usually is the limiting factor for parasitism. The optimum temperature range is 25 to 30°C.

4.6.3.5 Fertilization -

Fusarium is favored by weak plants, so fertilization does not favor the disease.

4.6.3.6 Plant Density -

no information

4.6.3.7 Controls -

No highly resistant cultivars are available but some, like the soft white wheats, are somewhat tolerant. Some cultivars are infected less frequently, apparently because of physical barriers to floret and
spikelet infection. Chemical seed treatments are a partial deterrent. Many fungicides are ineffective against internal inoculum, but most eradicate superficial inoculum and protect seedlings in infested soil.

Crop rotations with at least a 1 year break in cereal and grass cultivation is advised. Plowing to bury crop residues is also recommended since the fungi survive best on surface debris. Application of lime to soil, in some instances, reduces inoculum levels.

4.6.4 Relevant Criteria

1: Positive criteria

- The presence of brown stripes at the basis of the stem
- The bleached spikelets usually are sterile or contain only partially filled seed.

2: Supporting criteria

- Moisture is important
- Fungicides do not have any effect
- Wet summer = Fusarium year
- 25 C is favorable
4.6.5 Decision Tree

0 DISEASE-OCURRENCE|SYMPTOMS|PLANT|FOOT|STRIPES PRESENT

<table>
<thead>
<tr>
<th>NO</th>
<th>UNKNOWN</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

2 STRIPES.COLOR IN (BROWN, BLACK)

1 FOOT-DISCOLORISATION IS PRESENT

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 ROOT-MYCELIUM IS PRESENT

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 FOOT-DISCOLORISATION.COLOR = BROWN

<table>
<thead>
<tr>
<th>UNKNOWN</th>
<th>NO</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 DISEASE-OCURRENCE|DISEASE-CAUSE|

FUSARUM CULMORUM POSSIBLE
INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES
GAUMANNO GRAMINIS

4.7 GAUMANNO GRAMINIS

4.7.1 Systematic Names

4.7.1.1 Vernacular Names -
Dutch name: Tarwehalmdoder
Anglosaxian name: Take-all

4.7.1.2 Names Of The Disease Causes -
Gaeumannomyces graminis var. tritici
(pathogenic on wheat)

4.7.1.3 Related Diseases -
Gaeumannomyces graminis var. avenae
(pathogenic on oats)

4.7.2 Disease Cycle

4.7.2.1 Sources Of Primary Inoculum -
1. Infected wheat
2. Infected grasses
3. Host debris
4. Infected soil

4.7.2.2 Distribution Of The Disease Over The Plant -
Root infections in autumn or early spring often progress to the crown and foot. Infections occurring later are slowed by host maturation and normally remain confined to roots. Root death results from vascular occlusion and root domination by the pathogen.

- 91 -
4.7.2.3 Infection Of Other Plants -

1. Most plant-to-plant spread of take-all occurs via runner hyphae advancing through soil and across "root bridges".

2. Ascop-spores are released after wet periods and disseminated by splashing rain and somewhat by wind. Their effective dispersal beyond a few meters is infrequent.

4.7.2.4 General Description Of The Symptoms -

Take-all is most obvious near heading on plants grown in moist soil. Diseased crops appear uneven in height and irregular in maturity. Severely diseased plants easily break free at their crown when pulled from soil. Infected plants are stunted, mildly chlorotic, have few tillers and ripen prematurely. Their heads are bleached (white-heads) and sterile. Some may contain shrivelled grain and all are subject to darkening by 'sooty' molds. Roots are sparse, blackened and brittle from fungal invasion. A black-brown dry rot extends to the crown and basal stem where a superficial, dark shiny mycelial plate beneath the lowest leaf sheath is diagnostic.

Although these symptoms often are the first obvious indications of take-all, they are terminal stages of the disease. When blackened stems are not produced, diagnosis depends on darkened roots with internal and superficial dark mycelium and runner hyphae. Coarse runner hyphae of the pathogen are grouped in strands several mm long.

4.7.2.5 What Happens If The Disease Is Not Controlled ? -

Cultivars highly resistant to take-all are not available. Damage to wheat is related to the extent of root and basal stem (foot) colonization. Most plants withstand mild infections without visible symptoms of stress so yield losses often go unnoticed. When symptoms become obvious, yields normally are reduced by more than 50%.
4.7.3 Environmental Factors

4.7.3.1 Resistance Of Different Wheat Varieties -
no information available

4.7.3.2 Growthstages Most Susceptible To Infection -
The plant is susceptible in all growthstages

4.7.3.3 Soil Type -
Take-all is favored by alkaline, compacted, infertile (especially nitrogen- and phosphorus-deficient) and poorly-drained soils. Like many other soil-borne diseases, it increases in severity during the initial 3 to 6 years of continuous wheat cultivation.

4.7.3.4 Weather -
Take-all is a major disease in temperate climates where wheat and grass culture is intensive. Infections occurs throughout the growing season, but temperatures between 12 and 18 C are optimum.

4.7.3.5 Fertilization -
Applications of lime and nitrate fertilizers generally increase take-all. Ammonical and slow-release forms of nitrogen are less favorable. Similarly, spring applications of nitrogen normally support lower levels of take-all than autumn applications. Adequate phosphorus and potassium should be provided to promote root growth and differentiation.

4.7.3.6 Plant Density -
No information
4.7.3.7 Controls -

Take-all is a soil-borne disease, so it is hard to control. Crop rotation is recommended to limit inoculum levels. A phenomenon called 'take-all decline' occurs in successive wheat crops when, after an initial increase, the disease becomes less severe. The decline is a form of biological control caused by microorganisms antagonistic to G. graminis var. tritici. It is not economically feasible, however, for wheat growers to wait 3 to 5 years for the phenomenon to develop. In experimental tests, small amounts of 'decline' soils applied to infested 'virgin' soils reduce take-all in the succeeding wheat crop.

4.7.4 Relevant Criteria

1: Positive criteria

- Premature ripening of the plants
- Roots break off easily
- Roots are short and covered with brown runner hyphae

2: Supporting criteria

- Black mycelium plates at the basis of the stem

3: Negative criteria

- Lodging is no direct effect of take-all
4.7.5 Decision Tree

0 DISEASE-OCURRENCE | SYMPTOMS | PLANT | ROOT | MYCELUM PRESENT
\[ \begin{array}{c|c|c|}
| NO & YES & UNKNOWN \\
V & V & - \\
\hline
1 MYCELUM.COLOR IN (BROWN, BLACK) & - \\
\hline
| NO & YES & UNKNOWN \\
V & - & - \\
\hline
2 MYCELUM.SHAPE = HYphaE & - & - \\
\hline
| NO & YES & UNKNOWN \\
V & - & - \\
\hline
3 DISEASE-OCURRENCE | DISEASE-CAUSE | GAUMANNOMYCES-GRAMINIS POSSIBLE
4.8 SEPTORIA NODORUM

4.8.1 Systematic Names

4.8.1.1 Vernacular Names -

Dutch name: Bladvlekkenziekte
Anglosaxian name: Glume blotch

4.8.1.2 Names Of The Disease Causes -

Septoria nodorum (asexual stage)
Leptosheria nodorum (sexual stage)

4.8.1.3 Related Diseases -

Septoria tritici (asexual stage)
Mycosphaerella graminicola (sexual stage)

4.8.2 Disease Cycle

4.8.2.1 Sources Of Primary Inoculum -

1. Straw
2. Seed
3. Volunteer wheat
4. Stuble

4.8.2.2 Distribution Of The Disease Over The Plant -

The pathogens advance within killed tissues, but necrosis extends well beyond colonized cells apparently because of diffusible toxins.
4.8.2.3 Infection Of Other Plants -

1. Conidia are produced during wet periods, disseminated by splashing rain and initiate infections throughout the growing season.

2. Asco-spores are prevalent during late summer and autumn and are predominately wind-borne.

4.8.2.4 General Description Of The Symptoms -

Symptoms develop throughout the growing season on all aerial plant parts. Initial symptoms are chlorotic flecks usually on lowermost leaves, especially those contacting soil. The flecks expand into irregular lesions, normally 1-5x4-15 mm. Lesions of Nodorum are lens-shaped. Lesions are water-soaked briefly then become yellow, and finally, red-brown. Some develop gray-brown or ashen centers. Fruiting-bodies (Pycnidia) are brown in color.

4.8.2.5 What Happens If The Disease Is Not Controlled? -

The Septoria complex presently destroys nearly 2% of the world's wheat annually. Seed set is not appreciably modified but seed filling is impaired and shrivelled grain is lost with chaff at harvest. Some fungicide-protected fields yield 10 to 20% more grain than field in which foliar diseases. Losses are greatest when epidemics develop before heading.

4.8.3 Environmental Factors

4.8.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "6de Bschrijvende Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance.

<table>
<thead>
<tr>
<th>Crop variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>?</td>
</tr>
<tr>
<td>Citadel</td>
<td>5.5</td>
</tr>
</tbody>
</table>
4.8.3.2 Growthstages Most Susceptible To Infection -

Plant is infectable from DC10 (first leaf above ground) onwards. Infection visible about 2 weeks later, from DC13 (3 leaves) onwards. Latency period is about 2 weeks for varieties of average susceptibility at about 15 degrees Centigrade and high relative humidity.

4.8.3.3 Soil Type -

Clay grounds are usually more humid than sandy grounds, so there we find a wet environment in which septoria nodorum is more likely to occur.

4.8.3.4 Weather -

Infection requires 6 hr of wetness and secondary spores are generated within 10 to 20 days. Spore germination and infection are optimum between 15 and 25 C but can occur between 5 and 35 C. Septoria nodorum is most virulent between 20 and 27 C. Wet windy weather favors an epidemic while dry periods not only prevent infection but halt lesion and pycnidial development. Conidia remain viable for months between 2 and 10 C; those of S. nodorum tolerate temperatures above this range.

4.8.3.5 Fertilization -

Infectability increases with higher nitrate fertilization.
4.8.3.6 Plant Density -

There is an optimum plant density for occurrence of glume blotch. Crops in the Netherlands usually are sown at this density. With high plant densities plants don’t get wet very much, with low densities water evaporates quickly.

4.8.3.7 Controls -

Controls include the use of disease-free seed and the destruction or avoidance of infested straw, stubble and volunteer wheat. Stubble is burned occasionally in Europe and Asia. In the USA, stubble mulch and minimum tillage practices may increase disease risk. Rotations in which wheat or other cereals appear every third year will eliminate most carryover inoculum.

Wide row spacing and adequate, but not excessive, fertilization limits foliage density and humidity that favors disease.

4.8.4 Relevant Criteria

1: Positive criteria

- Spots on the leaves have to be present

2: Supporting criteria

- If fruiting-bodies are present, then they are brown in color.
0 DISEASE-OCURRENCE. SYMPTOMS | PLANT | LEAF | SPOT PRESENT

V

| NO | YES | UNKNOWN
V
-
V

1 SPOTS. COLOR IN (YELLOW, OCHRE, BROWN, GREY)

V

| NO | YES | UNKNOWN
V
-
V

2 SPOTS. FRUITING. BODIES ARE PRESENT

V

| NO | YES | UNKNOWN
V
-
V

3 SPOTS. FRUITING. BODIES. COLOR = BROWN

V

| NO | YES | UNKNOWN
V
-
V

4 DISEASE-OCURRENCE. DISEASE-CAUSE
SEPTORIA-NOORUM POSSIBLE

- 100 -
4.9 PUCINIA RECONDITA

4.9.1 Systematic Names

4.9.1.1 Vernacular Names -

Dutch name: Bruine roest
Anglosaxian name: Brown rust/Leaf rust

4.9.1.2 Names Of The Disease Causes -

Pucinia recondita f. sp. tritici

4.9.1.3 Related Diseases -

Pucinia striiformis/Yellow rust/Stripe rust
Pucinia graminis f. sp. tritici/Black rust/Stem rust

4.9.2 Disease Cycle

4.9.2.1 Sources Of Primary Inoculum -

1. Urediospores that survive locally or are wind-borne from distant hosts.
2. Wheat at lower elevations sometimes is infected by urediospores from grasses at higher altitudes.

4.9.2.2 Distribution Of The Disease Over The Plant -

Urediospores are one-celled, they are nutrient-independent and germinate in contact with water films. The formation of substomatal vesicles and intercellular hyphae completes the infection process. With free moisture and temperatures between 15 and 25 C, infection is completed in 6 to 8 hr and secondary urediospores are produced in 7 to 10 days.
4.9.2.3 Infection Of Other Plants -

Urediospores, produced in great numbers in spring and summer, are most important epidemiologically. They are dispersed by wind to other plants where they generate new infections and spores in intervals as short as 8 days.

4.9.2.4 General Description Of The Symptoms -

Uredia (frUITing bodies) are brown, principally on leaves and arranged into an irregular pattern of pustules. Urediospores are 15 to 30 micrometer in diameter. The epidermis is broken around the pustules. The parasite does not grow within the plant tissues, secondary pustules are created by self-infection.

4.9.2.5 What Happens If The Disease Is Not Controlled? -

Wheat rusts have been of great historical importance. The diseases, especially stem rust, are mentioned in the earliest records of wheat cultivation. They changed the course of early civilizations by destroying a major food source.

In the last decade in North America, rusts were conservatively estimated to decrease wheat yields by over 1 million metric tons annually. Similar statistics could be quoted for most wheat-growing regions of the world.

Damage to wheat depends on its stage of growth relative to rust development. Epidemics that occur before or during flowering are most detrimental. Head infections are especially damaging regardless of whether infections occur elsewhere on the plant. Rusts decrease the crop’s forage value and winter survival, and predispose plants to certain other diseases. Overall, they reduce plant vigor, seed filling and root growth. Cattle foraging on wheat find rusted plants less palatable and sometimes develop an allergic response to rusted forage.

4.9.3 Environmental Factors
4.9.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "61e Beschrijvende Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance.

<table>
<thead>
<tr>
<th>Crop variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>7</td>
</tr>
<tr>
<td>Citadel</td>
<td>4</td>
</tr>
<tr>
<td>Saiga</td>
<td>6.5</td>
</tr>
<tr>
<td>Okapi</td>
<td>5.5</td>
</tr>
<tr>
<td>Arminda</td>
<td>6.5</td>
</tr>
<tr>
<td>Marksman</td>
<td>8</td>
</tr>
<tr>
<td>Granada</td>
<td>8</td>
</tr>
</tbody>
</table>

4.9.3.2 Growthstages Most Susceptible To Infection -

Symptoms are most obvious summer but may occur anytime after seedlings emerge. Leaf rust has a higher temperature optimum than Stripe rust. The infection is in most cases limited to the leaves.

4.9.3.3 Soil Type -

Yellow as well as Brown rusts occur relatively often on claylike soils and are rather rare on sandy soils. Yellow rust is also rare on the loss-soil of limburg, the explanation for this is not clear.

4.9.3.4 Weather -

Urediospores germinate optimally between 15 and 22 C when moisture is not limiting.

4.9.3.5 Fertilization -

Nitrate fertilization favors the development of rusts under the conditions in the Netherlands.
4.9.3.6 Plant Density -

With very high plant densities (>500 stems per square meter) the infection decreases. These high densities are not to be found in the Netherlands, but one can find them in France.

4.9.3.7 Controls -

Rusts are best controlled by resistant cultivars. In the last 70 years the heritability of rust resistance has been utilized by breeders to systemically develop resistant wheats. Destroying alternate hosts interrupts the life cycle of rust fungi, limits their diversity and indirectly increases the stability of resistant cultivars.

Low cost, protectant or eradicant fungicides are sometimes used for rust control. They are applied as foliar sprays.

Avoiding monocultures of a given cultivar over vast areas restricts rust damage since heterogeneity in wheat requires corresponding heterogeneity in the rust population for epidemics to occur.

4.9.4 Relevant Criteria

1: Positive criteria

- The brown pustules are arranged in an irregular pattern

2: Supporting criteria

- The plants have to be healthy, because the parasites are obligate.
- A dry summer checks the infections by rusts
- A severe winter causes all leaves to fall, so the rusts can not get through the winter.
4.9.5 Decision Tree

0 DISEASE-occurrence, Symptoms | Plant | Leaf | Spores Present

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

1 Spores. Shape = patches

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 Spores. Epidermis is broken

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 Spores. Color in (Orange, Orange-Brown, Brown, DarkBrown)

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Disease-occurrence, Disease-Cause

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PUCINIA RECONDITA POSSIBLE
4.10  PUCINIA STRIIFORMIS

4.10.1  Systematic Names

4.10.1.1  Vernacular Names -

Dutch name: Gele roest
Anglosaxian name: Yellow rust/Stripe rust

4.10.1.2  Names Of The Disease Causes -

Pucinia striiformis

4.10.1.3  Related Diseases -

Pucinia recondita/Brown rust/Leaf rust
Pucinia graminis f. sp. tritici/Black rust/Stem rust

4.10.2  Disease Cycle

4.10.2.1  Sources Of Primary Inoculum -

1. Urediospores that survive locally or are wind-borne from distant hosts.

2. Wheat at lower elevations sometimes is infected by urediospores from grasses at higher altitudes.

4.10.2.2  Distribution Of The Disease Over The Plant -

Urediospores are one-celled, they are nutrient-independent and germinate in contact with water films. The formation of substomatal vesicles and intercellular hyphae completes the infection process. With free moisture and temperatures between 15 and 25 C, infection is completed in 6 to 8 hr and secondary urediospores are produced in 7 to 10 days.
4.10.2.3 Infection Of Other Plants -

Urtediospores, produced in great numbers in spring and summer, are most important epidemiologically. They are dispersed by wind to other plants where they generate new infections and spores in intervals as short as 8 days.

4.10.2.4 General Description Of The Symptoms -

Symptoms of yellow rust vary but usually appear earlier in spring than symptoms of leaf or stem rust. Uredia (fruiting bodies) are yellow, principally on leaves and heads and often arranged into conspicuous stripes, because the parasite grows within the plant tissues. Individual pustules (spots) are 0.3-0.5x0.5-1 mm, but their linear orientation between vascular bundles can progress the length of the leafblade. In heads, uredia normally occur on the ventral surface of glumes and seeds sometimes are infected.

4.10.2.5 What Happens If The Disease Is Not Controlled ? -

Wheat rusts have been of great historical importance. The diseases, especially stem rust, are mentioned in the earliest records of wheat cultivation. They changed the course of early civilizations by destroying a major food source.

In the last decade in North America, rusts were conservatively estimated to decrease wheat yields by over 1 million metric tons annually. Similar statistics could be quoted for most wheat-growing regions of the world.

Damage to wheat depends on its stage of growth relative to rust development. Epidemics that occur before or during flowering are most detrimental. Head infections are especially damaging regardless of whether infections occur elsewhere on the plant. Rusts decrease the crop's forage value and winter survival, and predispose plants to certain other diseases. Overall, they reduce plant vigor, seed filling and root growth. Cattle foraging on wheat find rusted plants less palatable and sometimes develop an allergic response to rusted forage.
4.10.3 Environmental Factors

4.10.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "61e Bschrijvende Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance.

<table>
<thead>
<tr>
<th>Crop variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>7</td>
</tr>
<tr>
<td>Citadel</td>
<td>7</td>
</tr>
<tr>
<td>Saiga</td>
<td>6</td>
</tr>
<tr>
<td>Okapi</td>
<td>5</td>
</tr>
<tr>
<td>Arinda</td>
<td>7</td>
</tr>
<tr>
<td>Marksman</td>
<td>6</td>
</tr>
<tr>
<td>Granada</td>
<td>4</td>
</tr>
</tbody>
</table>

4.10.3.2 Growthstages Most Susceptible To Infection -

Symptoms are most obvious in spring and summer but may occur anytime after seedlings emerge. All aerial plant parts are susceptible.

4.10.3.3 Soil Type -

Yellow as well as Brown rusts occur relatively often on claylike soils and are rather rar on sandy soils. Yellow rust is also rare on the loss-soil of limburg, the explanation for this is not clear.

4.10.3.4 Weather -

Infections may occur throughout the autumn and winter since mycelium remains viable to -5 C. Urediospores lose viability rapidly at temperatures above 15 C. They germinate optimally between 5 and 15 C with limits near 0 and 21 C. Disease development is most rapid between 10 and 15 C with intermittent rain or dew.
4.10.3.5 Fertilization -

Nitrate fertilization favors the development of rusts under the conditions in the Netherlands.

4.10.3.6 Plant Density -

With very high plant densities (>500 stems per square meter) the infection decreases. These high densities are not to be found in the Netherlands, but one can find them in France.

4.10.3.7 Controls -

Rusts are best controlled by resistant cultivars. In the last 70 years the heritability of rust resistance has been utilized by breeders to systemically develop resistant wheats. Destroying alternate hosts interrupts the life cycle of rust fungi, limits their diversity and indirectly increases the stability of resistant cultivars.

Low cost, protectant or eradicant fungicides are sometimes used for rust control. They are applied as foliar sprays.

Avoiding monocultures of a given cultivar over vast areas restricts rust damage since heterogeneity in wheat requires corresponding heterogeneity in the rust population for epidemics to occur.

4.10.4 Relevant Criteria

1: Positive criteria

- The uredia have to be arranged into stripes

2: Supporting criteria

- The plants have to be healthy, because the parasites are obligate.
- A dry summer checks the infections by rusts
- A severe winter causes all leaves to fall, so the rusts cannot get through the winter.
4.10.5 Decision Tree

0 DISEASE-OccURRENCE. SYMPTOMS | PLANT | LEAF | SPORES PRESENT

<table>
<thead>
<tr>
<th></th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

1 SPORES. SHAPE = STRIPES

<table>
<thead>
<tr>
<th></th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

2 SPORES. COLOR IN (YELLOW, YELLOW-BROWN, BROWN)

<table>
<thead>
<tr>
<th></th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

3 DISEASE-OccURRENCE. DISEASE-CAUSE

| PUCINIA-STRIFORMIS POSSIBLE |
4.11 SEPTORIA TRITICI

4.11.1 Systematic Names

4.11.1.1 Vernacular Names –

Dutch name: Bladvlekkenziekte
Anglosaxian name: leaf spot disease

4.11.1.2 Names Of The Disease Causes –

Septoria tritici (asexual stage)
Mycosphaerella Graminicola (sexual stage)

4.11.1.3 Related Diseases –

Septoria nodorum (asexual stage)
Leptosphaeria nodorum (sexual stage)

4.11.2 Disease Cycle

4.11.2.1 Sources Of Primary Inoculum –

1. Straw
2. Seed
3. Volunteer wheat
4. Stuble

4.11.2.2 Distribution Of The Disease Over The Plant –

The pathogens advance within killed tissues, but necrosis extends well beyond colonized cells apparently because of diffusible toxins.
4.11.2.3 Infection Of Other Plants -

1. Conidia are produced during wet periods, disseminated by splashing rain and initiate infections throughout the growing season.

2. Ascospores are prevalent during late summer and autumn and are predominately wind-borne.

4.11.2.4 General Description Of The Symptoms -

Symptoms develop throughout the growing season on all aerial plant parts. Initial symptoms are chlorotic flecks usually on lowermost leaves, especially those contacting soil. The flecks expand into irregular lesions, normally 1-5x4-15 mm. Lesions of Tritici tend to be restricted laterally and assume parallel sides. Lesions are water-soaked briefly then become yellow, and finally, red-brown. Some develop gray-brown or ashen centers. Fruiting-bodies (Pycnidia) are black in color.

4.11.2.5 What Happens If The Disease Is Not Controlled? -

The Septoria complex presently destroys nearly 2% of the world's wheat annually. Seed set is not appreciably modified but seed filling is impaired and shrivelled grain is lost with chaff at harvest. Some fungicide-protected fields yield 10 to 20% more grain than field in which foliar diseases, principally Septoria leaf spots are allowed to develop. Losses are greatest when epidemics develop before heading.

4.11.3 Environmental Factors

4.11.3.1 Resistance Of Different Wheat Varieties -

The figures are taken from the "61e Bschrijvende Rassenlijst voor Landbouwgewassen 1986". A high figure stands for a high resistance.

<table>
<thead>
<tr>
<th>Crop variety</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nautica</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>- 113 -</td>
</tr>
</tbody>
</table>
4.11.3.2 Growthstages Most Susceptible To Infection -

Plant is infectable from DC10 (first leaf above ground) onwards. Infection visible about 2 weeks later, from DC13 (3 leaves) onwards. Latency period is about 2 weeks for varieties of average susceptibility at about 15 degrees Centigrade and high relative humidity.

4.11.3.3 Soil Type -

Clay grounds are usually more humid than sandy grounds, so there we find a wet environment in which Septoria tritici is more likely to occur.

4.11.3.4 Weather -

Infection requires 6 hr of wetness and secondary spores are generated within 10 to 20 days. Spore germination and infection are optimum between 15 and 25 C but can occur between 5 and 35 C. Septoria tritici is most virulent between 15 and 20 C. Wet windy weather favors an epidemic while dry periods not only prevent infection but halt lesion and pycnidial development.

4.11.3.5 Fertilization -

Infectability increases with higher nitrate fertilization.
INTEGRITY CONSTRAINTS FOR WINTER WHEAT DISEASES
SEPTORIA TRITICI

4.11.3.6 Plant Density -

There is an optimum plant density for occurrence of leaf spot. Crops in the Netherlands usually are sown at this density. With high plant densities plants don’t get wet very much, with low densities water evaporates quickly.

4.11.3.7 Controls -

Controls include the use of disease-free seed and the destruction or avoidance of infested straw, stubble and volunteer wheat. Stubble is burned occasionally in Europe and Asia. In the USA, stubble mulch and minimum tillage practices may increase disease risk. Rotations in which wheat or other cereals appear every third year will eliminate most carryover inoculum.

Wide row spacing and adequate, but not excessive, fertilization limits foliage density and humidity that favors disease. In cooler climates, late-autumn planting delays crop development in summer so most foliage develops during warmer periods less favorable to Septoria tritici.

4.11.4 Relevant Criteria

1: Positive criteria

- Spots on the leaves have to be present

2: Supporting criteria

- If fruiting-bodies are present, then they are black in color.
4.11.5 Decision Tree

0 DISEASE-OCURRENCE | SYMPTOMS | PLANT | LEAF | SPOT PRESENT

V

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>
-   -   -

1 SPOTS COLOR IN (YELLOW, OCHRE, BROWN, GREY)

V

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>&lt;</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 SPOTS FRUITING BODIES ARE PRESENT

V

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

3 SPOTS FRUITING BODIES COLOR = BLACK

V

<table>
<thead>
<tr>
<th>NO</th>
<th>YES</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

4 DISEASE-OCURRENCE | DISEASE-CAUSE | SEPTORIA TRITICI POSSIBLE

V

- 116 -
CHAPTER 5

USER INTERFACE

The user interface of a decision support system should be designed after extensive analysis of user information requirements and interface ergonomics. No such analysis took place for FAD and the current interface is just a clumsy dialogue which probably would annoy any user. An obvious improvement would be to use screen forms instead of a sequential dialogue. After requesting the identification of the parcel the consult is about, the dialogue requests symptoms observations. The structure of this dialogue is shown in figure 24. The tree of figure 24 is walked through from top to bottom and from left to right, as far as it is needed. When the answer to a question is positive, the dialogue proceeds with the next question on a deeper level. When the answer is negative, it proceeds with the next question on the same level. The dialogue tree derived from the syndroms structure hierarchy of figure 22 by deleting any attribute not needed to decide whether a disease cause should be excluded.
The output of a consult is a report like the one shown in figure 25.
Symptoms of Gaumannomyces graminis:

<table>
<thead>
<tr>
<th>Value</th>
<th>Observed</th>
<th>Value</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color of root mycelium</td>
<td>yes</td>
<td>Brown</td>
<td></td>
</tr>
<tr>
<td>Shape of the mycelium on the roots</td>
<td>yes</td>
<td>hyphae</td>
<td></td>
</tr>
</tbody>
</table>

Symptoms of Septoria nodorum:

<table>
<thead>
<tr>
<th>Value</th>
<th>Observed</th>
<th>Value</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color of leaf spots</td>
<td>yes</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>Presence of spores on the leaves</td>
<td>yes</td>
<td>Present</td>
<td></td>
</tr>
<tr>
<td>Color of spores on the leaves</td>
<td>no</td>
<td></td>
<td>Brown</td>
</tr>
</tbody>
</table>

Symptoms of Septoria tritici:

<table>
<thead>
<tr>
<th>Value</th>
<th>Observed</th>
<th>Value</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color of leaf spots</td>
<td>yes</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>Presence of spores on the leaves</td>
<td>yes</td>
<td>Present</td>
<td></td>
</tr>
<tr>
<td>Color of spores on the leaves</td>
<td>no</td>
<td></td>
<td>Black</td>
</tr>
</tbody>
</table>

The symptoms are:

- Mycelium on the roots: GAUMANNOMYCES-GRAMINIS
- Observation of Brown mycelium on the roots: GAUMANNOMYCES-GRAMINIS
- Mycelium on the roots occurs in the shape of Hyphae: GAUMANNOMYCES-GRAMINIS
- Spots on the leaves: SEPTORIA-NODORUM, SEPTORIA-TRITICI
- Observation of Yellow spots on the leaves: SEPTORIA-NODORUM, SEPTORIA-TRITICI
- Observation of spores on the leaves: SEPTORIA-NODORUM, SEPTORIA-TRITICI

Figure 25
Since the inferences are shallow and fuzzy and take a lot of context into account, the written explanations on the report should provide the support of each observation for each disease and any further supporting material per disease which can be retrieved from the history of the PARCEL and the WEATHER. This extra supporting material is not included in the current version of FAD. For each observation the possible diseases it supports are given, so that the user can see which observation is not used for any disease. Finally, for each observation not made but decisive for a disease the expected value is given, so that the user can take the report to the parcel and make the final decision himself. We think this is the most valuable aspect of the report.
6.1 CURRENT IMPLEMENTATION

In this chapter we describe the final implementation of FAD in Babylon. The emphasis in this implementation has been on the possibilities of a report writer and the use of dynamic instantiation.

6.1.1 The Disease Cause Process

Every time the system is consulted, a state transition of the DISEASE-CAUSE process takes place. This transition is only reported upon, future implementations should also take care of chronicles which are stored in a database. The instruction part of the system implemented in Babylon looks as follows:

```
(SETQ LEAF-MYCELIUM NIL)
(SETQ FOOT-STRIPES NIL)
(SETQ FOOT-DISCOLORISATION NIL)
(SETQ FOOT-SYMPTOMS NIL)
(SETQ LEAF-BUGS NIL)
(DEFINSTANCE SYMPTOMS-OCURRENCE OF SYMPTOMS-OCURRENCE-FRAME)
(<- DISEASE-OCURRENCE :INSTANTIATE)
(DEFVAR F1)
(DEFVAR F2)
(SETQ F2 (OPEN "HIPPO:KB3>UIT.DAT" ':DIRECTION ':OUTPUT))
(SETQ F1 (MAKE-BROADCAST-STREAM STANDARD-OUTPUT F2))
(<- DISEASE-OCURRENCE :REPORT)
(<- SYMPTOMS-OCURRENCE :REPORT)
(CLOSE F2)
(CLOSE F1))
```
In this instruction part of FAD we see three message-sending Babylon actions. The other instructions are lisp functions used to influence the LISP world. The first five instructions set dynamically instantiated frame instances to NIL. This is necessary, because once created, instances cannot be destroyed. Dynamic instantiation is done by simply putting the DEFINSTANCE macro where we need it. F2 is the file the report is written to, F1 is the output "stream" connected with this file and the terminal.

DISEASE-OCURRENCE is an instance of the DISEASE-OCURRENCE-FRAME which is defined as follows:

(DEFFRAME DISEASE-OCURRENCE-FRAME
  (SLOTS (DAY -) ; time stamp
          (MONTH -) ; time stamp
          (YEAR -) ; time stamp
          (PARCEL-ID -
            :POSSIBLE-VALUES #A(- :POSSIBLE-PARCEL NIL)
            :ASK ("-%Which parcel do you want to consult about? ")
          (GROWTH-STAGE 0) ; not yet implemented
          (POSSIBLE-DISEASE-CAUSES #A(NIL NIL :ADD-DISEASE-CAUSE-TO-LIST))
          (PLANT-SYMPTOMS-SLOT -
            :POSSIBLE-VALUES (:INSTANCE-OF
                               PLANTS-SYMPTOMS-FRAME))))

There is only one instance of this frame, called DISEASE-OCURRENCE. The slot POSSIBLE-DISEASE-CAUSES is implemented as a list and will be used to store instances of possible diseases. The slot has a put behavior :ADD-DISEASE-CAUSE-TO-LIST which adds the instances of the possible diseases to the list. The instances of the diseases are created when the observed and given symptoms give reason for it. In the frames of the diseases we find the symptoms specific for a certain disease. We have as many frames as we have different diseases, for instance the frame describing Septoria tritici looks as follows:

(DEFFRAME SEPTORIA-TRITICI-FRAME
  (SUPERS DISEASE-CAUSE-FRAME)
  (SLOTS (LEAF-SPORES-COLOR #A((LEAF-SPORES COLOR) :GET-INDIRECT NIL)
               :POSSIBLE-VALUES (:ONE-OF "Yellow" "Ochre"
                                  "Brown" "Grey" unknown)
               :EXPECTED-VALUE "Yellow")
  (LEAF-SPORES #A((LEAF-SPORES SPORES) :GET-INDIRECT NIL)
               :POSSIBLE-VALUES (:ONE-OF "Present" "Absent"
                                  unknown)
               :EXPECTED-VALUE "Present")
  - 122 -
The PLANT-SYMPTOMS-SLOT of the instance DISEASE-OCURRENCE is used to store the information about all the symptoms that are collected during a session. The PLANT-SYMPTOMS frame is defined as a structure hierarchy, see figure 22. For all the slots we see :GET-INDIRECT specified as a get behavior, this behavior gets the value of the defined slot of a certain instance if the actual slot is undefined:

(LEAF-SPOTS-COLOR #A((LEAF-SPOTS SPORES-COLOR)
:GET-INDIRECT NIL)
:POSSIBLE-VALUES (:ONE-OF "Brown" "Black")
:EXPECTED-VALUE "Black")

LEAF-SPOTS-COLOR will get the value of slot COLOR defined in the instance LEAF-SPOTS, when a get is performed and when LEAF-SPOTS-COLOR is undefined. The property :EXPECTED-VALUE is used by the report writer and will be described in one of the following sessions. The top node in this hierarchy is defined as:

(DEFFRAME PLANT-SYMPTOMS-FRAME
(SLOTS (SPECIALISATIONS #A(- nil :PUT-SPECIALISATIONS)
 :POSSIBLE-VALUES (:SOME-OF
 "Symptoms on the root"
 "Symptoms on the foot"
 "Symptoms on the leaf")

(ROOT-SYMPTOMS-SLOT unknown
 :POSSIBLE-VALUES (:INSTANCE-OF ROOT-SYMPTOMS-FRAME))

(FOOT-SYMPTOMS-SLOT unknown
 :POSSIBLE-VALUES (:INSTANCE-OF FOOT-SYMPTOMS-FRAME))

(LEAF-SYMPTOMS-SLOT unknown
 :POSSIBLE-VALUES (:INSTANCE-OF LEAF-SYMPTOMS-FRAME))))

In this version of FAD all the slots of the PLANT-SYMPTOMS-FRAME can be used. We will not describe the whole tree in detail, see figure 22, but we will elaborate more on the root symptoms. The ROOT-SYMPTOMS-FRAME is described as:

(DEFFRAME ROOT-SYMPTOMS-FRAME
(SLOTS (MYCELIUM #A(- nil :PUT-MYCELIUM)
 :POSSIBLE-VALUES (:ONE-OF
 yes
 no
 unknown)
 :ASK ("Did you observe mycelium on the roots ? ")

(ROOT-MYCELIUM-SLOT Unknown
 :POSSIBLE-VALUES (:INSTANCE-OF ROOT-MYCELIUM-FRAME))

- 123 -
The slots MYCELIUM and BREAK-OFF are leaves in the tree, but ROOT-MYCELIUM-SLOT is filled with an aggregate described as:

(DEFFRAME ROOT-MYCELIUM-FRAME
(SUPERS DISEASE-LIST-HANDLER)
(SLOTS (COLOR -
:POSSIBLE-VALUES (:SOME-OF
"Brown"
"Black"
unknown)
:ASK ("%What is the color of the mycelium on the roots ? ")
:TEXT "%20TObservation of a mycelium on the roots: 
:SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-SLOT-LIST))
(SHAPE -
:POSSIBLE-VALUES (:ONE-OF "hyphae" "Patches" unknown)
:ASK ("%What is the shape of the mycelium ?")
:TEXT "%20TMycelium on the roots occurs"+
in the shape of Hyphae:
:SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-SLOT-LIST))
(LENGTH -
:POSSIBLE-VALUES (:ONE-OF "normal" "abnormal" unknown)
:ASK ("%What can you say about the length of the roots ?")
:TEXT "%20TObservation of abnormal length of the roots: 
:SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-SLOT-LIST))
(SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-INSTANCE-LIST))
(TEXT "%20TMycelium on the roots: ")

The slots TEXT and SUPPORTED-DISEASES and also the properties :TEXT and :SUPPORTED-DISEASES are used for the report writer and will be discussed in a later section. Described in chapter 4 as decision trees, the integrity constraints on disease causes and symptoms are implemented as lisp functions. For every disease or pest a Lisp function has been defined. And after the system has collected the symptoms by means of a dialogue as described in the next section, all the diseases lisp functions will be called. Every function uses the stored information about the symptoms to decide if a certain disease is possible. The Lisp function for GAUMANNOMYCES GRAMINIS is described as:
(DEFUN GAUMANNOMYCES GRAMINIS
   (TAGBODY
    0 (IF (IS-INSTANCE ROOT-MYCELIUM)
        ;; THEN
        (PROGN
          ( <- SYMPTOMS-OCURRENCE:PUT 'SYMPTOMS-LIST ROOT-MYCELIUM)
          ( <- ROOT-MYCELIUM:PUT 'SUPPORTED-DISEASES GAUMANNOMYCES GRAMINIS)
          (GO 1)
        ;; ELSE
        (GO FAIL))
    1 (IF (MEMBER ( <- ROOT-MYCELIUM:GET 'COLOR)
                   (LIST "Brown" "Black"))
        ;; THEN
        (PROGN
          ( <- ROOT-MYCELIUM:PUT 'COLOR GAUMANNOMYCES-GRAMINIS
                                :SUPPORTED-DISEASES)
          (GO 2)))
        ;; ELSE
        (IF (EQ ( <- ROOT-MYCELIUM:GET 'COLOR) 'unknown)
             ;; THEN
             (GO 2)
             ;; ELSE
             (GO FAIL))
    2 (IF (EQUAL ( <- ROOT-MYCELIUM:GET 'SHAPE) "hyphae")
             ;; THEN
             (PROGN
               ( <- ROOT-MYCELIUM:PUT 'SHAPE GAUMANNOMYCES-GRAMINIS
                               :SUPPORTED-DISEASES)
               (GO 3)))
             ;; ELSE
             (IF (EQ ( <- ROOT-MYCELIUM:GET 'SHAPE) 'unknown)
              ;; THEN
              (GO 3)
             ;; ELSE
             (GO FAIL))
    3 ( <- DISEASE-OCURRENCE:PUT 'POSSIBLE-DISEASE-CAUSES
                      GAUMANNOMYCES-GRAMINIS)
       (FAIL)))

In current version of FAD all assignments to POSSIBLE-DISEASE-CAUSES are possibilistic, i.e. the assigned values are added to a set of possible diseases. The only instance in this function that has not been mentioned before is SYMPTOMS-OCURRENCE. This instance is used to collect all the observed symptoms in a list which can be used by the report writer.
6.1.2 The User Interface

The data used in the decision function are collected by means of a dialogue. The first question asked concerns the identification of the parcel. The dialogue has been implemented by a rule-set named DIALOGUE:

(DEFRULE-SET :DIALOGUE
 (SYMPTOMS ($TRUE)
  ($ASK (PLANT-SYMPTOMS SPECIALISATIONS))
  ($ASK (ROOT-SYMPTOMS MYCELIUM)
   (ROOT-SYMPTOMS BREAK-OFF))
  ($ASK (ROOT-MYCELIUM COLOR)
    (ROOT-MYCELIUM SHAPE)
    (ROOT-MYCELIUM LENGTH)))

We only show here the rules related to the root symptoms. There are additional rules for the foot and leaf symptoms. The rule-set is called when a DISEASE-OCURRENCE is instantiated. The first rule asks to specify the sort of symptoms that are observed. Depending on the answer, instances of the different symptoms will be dynamically instantiated.

By putting the dialogue into a rule-set, the order of the questions in the dialogue is determined dynamically by this rule-set. The actual text of the questions is defined as the :ASK property in the specific slots.

After the instance PLANTS-SYMPTOMS has been filled with information, the value of the POSSIBLE-DISEASE-CAUSES slot can be inferred. All the lisp functions for the diseases are called as a part of the INSTANTIATE behavior of DISEASE-OCURRENCE, so the diagnosis can be set, but at the same time these function organize information to be used by the report writer.
6.1.3 The Report Writer

The user of a diagnostic expert system will not be satisfied by just a plain answer. He or she will want to know how the answer was reached. A written report is a way to give the user something to take to the parcels and add to his records.

The form of the report was described in a previous chapter. In this implementation of FAD we introduced a new object SYMPTOMS-OCURRENCE. This object has a slot where we store a list of observed symptoms. The list will be created by the diagnostic lisp functions:

\[
(<- \text{SYMPTOMS-OCURRENCE} : \text{PUT} \ '\text{SYMPTOMS-LIST} \ \text{ROOT-MYCELIUM})
\]

SYMPTOMS-LIST is the slot name and ROOT-MYCELIUM is the name of a symptom. Symptoms are implemented as instances or as slots depending upon their place in the structure hierarchy of figure 22. When the symptom is defined as a instance it has a slot named SUPPORTED-DISEASES, when a symptom is defined as a slot it has a property named :SUPPORTED-DISEASES. These slots or properties are used to store the list of diseases that are supported by the particular symptom. These lists will also be updated by the diagnostic lisp functions:

\[
(<- \text{ROOT-MYCELIUM} : \text{PUT} \ '\text{SUPPORTED-DISEASES} \ \text{GAUMANNOMYCES-GRAMINIS})
\]

ROOT-MYCELIUM is the name of a symptom and GAUMANNOMYCES-GRAMINIS is one of the diseases.

After all the diagnostic functions have been called we have a list of symptoms and a list of diseases. In the INSTRUCTIONS part of the system we find the two instructions:

\[
(<- \text{DISEASE-OCURRENCE} : \text{REPORT})
\]
\[
(<- \text{SYMPTOMS-OCURRENCE} : \text{REPORT})
\]

These two messages use the two lists to create the final report. Every disease in the list of possible diseases will be send a :REPORT message, these :REPORT messages activate a certain behavior, for instance the REPORT behavior of GAUMANNOMYCES-GRAMINIS has been defined as follows:

\[
\text{(DEFBEHAVIOR (GAUMANNOMYCES-GRAMINIS-FRAME :REPORT)})()
\]
\[
(\text{FORMAT F1 "-\%"})
\]
\[
(\text{FORMAT F1 "-\% "10TSymptoms of Gaumannomyces graminis: "})
\]
\[
(\text{FORMAT F1 "-\%"})
\]
\[
(\text{FORMAT F1 "-\% "10T "42<OBERVED";VALUE";EXPECTED VALUE">"})
\]

- 127 -
CURRENT IMPLEMENTATION

This behavior, when activated, will generate the report about GAUMANNOMYCES--GRAMINIS shown in figure 25.

The texts necessary for the generation of the symptoms part of the report are stored as values of TEXT slots if the symptoms are instances and values of :TEXT properties if the symptoms are values of slots:

(LENGTH -
  :POSSIBLE-VALUES (:ONE-OF "normal" "abnormal" unknown)
  :ASK ("%What can you say about the length of the roots ?")
  :TEXT "%20'The observation of abnormal length of the roots: "
  :SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-SLOT-LIST))
  (SUPPORTED-DISEASES #A(nil nil :ADD-DISEASE-TO-INSTANCE-LIST))
  (TEXT "%20'The mycelium on the roots: ")
)

The strings defined in the property :EXPECTED-VALUE can be used to generate information about a disease that has not been observed by the user. In those cases the most expected value of a certain symptom will be put in the report:

(LEAF-SPORES #A((LEAF-SPOTS SPORES) :GET-INDIRECT NIL)
  :POSSIBLE-VALUES (:ONE-OF "Present" "Absent" unknown)
  :EXPECTED-VALUE "Present")
CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 EVALUATION OF BABYLON

In the second report we evaluated the use of Babylon as an implementation mechanism for the model of crop diseases and for the dialogue with the farmer. The major conclusions were that dynamic frame instantiation and deletion of instances should be possible, and that functions for searching instances are needed. These facilities are routinely provided by database management systems, where a frame corresponds to a table or relation and an instance to a tuple or row in the table. A database query is an arbitrary Boolean condition on an arbitrary number of tables, i.e. on the sets of instances of one or more frames, and delivers the instances or instance combinations satisfying the condition.

Another conclusion was that the dialogue would be more user-friendly if form management would be provided.

These facilities can be programmed by the user of Babylon himself, but then everything can be programmed by the user, and we do not want to re-invent the wheels which have been invented in the history of DBMS's since the 60's.

In this chapter we described two experiments, one with dynamic instantiation and one with report writers. Dynamic instantiation proved possible in a clumsy way. A major problem is that instances must be bound to names. In our second report we mentioned surrogates as the basic naming mechanism of an information system. Surrogates are generated by the system itself and visible to the user, but cannot be modified by him. A surrogates must be identified by the values of the attributes it has, and this can be uniquely done if a combination of attributes is uniquely identifying, like the X,Y coordinate pair of PARCEL. If there is no uniquely identifying description, then the query requesting the instance will in general deliver a set of
instances. This requires the search function of which the need is mentioned above. In this way, instances are identified by the value of their attributes; addressing is associative, i.e. by content, not by location in memory. This is a fundamental difference between relational databases on the one hand and older schemes like hierarchical and network databases on the other, which use pointers to locate data in memory. The use of pointers slows updating but speeds up queries, and in those schemes the pointer structure, motivated by implementation considerations, is not separated from the conceptual structure. Associative addressing also contrasts with frame systems, where frame instances are named. An instance name is just a synonym for its address, so named frame instances are addressed by location. Thus, search functions are not needed, because each instance is directly pointed at by its name. This scheme is not workable for the large number of instances typically stored in a database and there, search functions are needed.

A second problem with dynamic instantiation in Babylon is that instances cannot be destroyed. Binding a name to an instance only happens when the name has no instance as its value. Once it has a value, a DEFINE will have no effect, even though we would like to forget the old instance and create a new one. In the current version of FAD this is solved simply by setting the instance name to NIL when we want to delete an instance. The instance is not destroyed, but unreachable, and will be eventually destroyed by the garbage collector. In DBMS's, tuple deletion is commonly done by setting a delete flag for a record and cleaning up the file and its indexes once in a while. The work of the garbage collector is then done by the database administrator.

Implementing the report writer in a flexible way was greatly facilitated by the possibility to add programmer-defined properties to slots, e.g. :EXPECTED-VALUE and :SUPPORTED-DISEASES for symptoms slots. We have no criteria yet by which we can evaluate report writers.

We wind up the evaluation of Babylon with a discussion of the differences in design philosophy between a DBMS like Oracle running on a machine like a VAX and an expert system shell like Babylon running on a machine like a Symbolics. On a Symbolics, the Shell, the application developed in it, and the information communicated to the system during a user session are all part of a huge collection of machine code called "the world." There is no separation between the operating system code of the Symbolics, the code for the Shell, for the application, and for the data entered by the user. On the VAX under VMS there is a layered structure of virtual machines as described by Tanenbaum [1976]: The bottom layer is the hardware, the
next layer is the conventional machine level defined by the microcode, which in turn is the base for the operating system level, which is the base for several alternative language levels which serve as a base for different applications. One virtual machine is built on top of another such that it can be replaced by another without affecting the virtual machine on which it runs. The reverse independence, replacability of a lower level virtual machine without affecting the machines which run on it, is harder to guarantee but is usually provided. On a Symbolics, there is hardly a separation between the operating system level, the language level, and the application level. Objects on all three levels are accessible to all users and an awkward package system is needed to keep the names used on different levels and by different applications apart. But though the package system can prevent name clashes, it does not prevent users from accessing all objects in the operating system, language level and application level. The concept of world as used on a Symbolics goes against the concept of a layered structure as described by Tanenbaum. This has two disadvantages. In the first place, the human mind, finite as it is, follows a divide-and-conquer strategy to master complexity and the layered machine structure aids the user greatly to divide the world into that part in which he is interested, which is his application, and that part which he keeps out of his mind, which is the hierarchy of virtual machines on which his application runs. Symbolics users use the same strategy, but are not protected from inadvertently affecting the operating system or language level and this hinders application development. Secondly, in each edit-compile-run cycle on a Symbolics, executable code and data are added to the world. If the garbage collector is switched off, this is a memory-greedy way of working. In contrast, on the VAX/VMS, in an edit-compile-run cycle executable code is added to disk, not to virtual memory, and code generated during running is destroyed after the run. When the garbage collector of a Symbolics is switched on, space-inefficiency is traded for time-inefficiency, while a still sizeable amount of virtual memory must be unoccupied if the garbage collector is to be able to do its work. Compilation on the Symbolics can be done on disk but to run the compiled code, it must be loaded into virtual memory and it then stays there until it becomes inaccessible and is destroyed by the garbage collector or the machine is cold-booted. The conclusion is that both conceptually and implementation-wise the Symbolics design philosophy is inefficient.

Oracle follows the layered machine design philosophy and Babylon follows the Symbolics design philosophy. Because in a layered machine code and data generated during a run in virtual memory is destroyed after the run, Oracle saves all changes made by the user on disk. Any creation, deletion, or updating of an instance of a frame is a change. To this end it provides the user with a COMMIT operation, which saves
all changes made since the start of the Oracle run or since the last COMMIT. When a run is exited without a COMMIT, Oracle asks the user if changes should be COMMITed. By default, in interactive mode every change is COMMITed immediately. This is in sharp contrast to Babylon, which fixes the number of instances at the beginning of a run and provides a RESET operation which undoes the changes made during one run. The RESET absolves one from re-compiling the application code. This situation would not be improved if dynamic instantiation is provided, since it is hard to see how RESET can avoid destroying created instances, which are bound to names during program execution. But even if they are not destroyed during RESETing, they are not saved by Babylon either. The result is that in Oracle changes survive over sessions while in Babylon no change survives sessions, unless the Symbolics world is saved or the programmer takes explicit action to write instances to disk. Any version of the world can be saved, i.e. dumped to disk, and restored, i.e. restored from disk. One would not want to do this to save only a few changes since this would be a huge waste of disk space. It would be equivalent to saving the machine code for VMS, Oracle and the application along with the instances one wants to save on disk. It is, of course possible to write and read instances to a file, but this is the start of a road which is likely to re-invent the weels which were invented in the last 20 or 30 years of database management. Following the layered machine design philosophy, we would like those facilities to be provided by the Shell or DBMS, so that we can concentrate on application development.

From these considerations we conclude that Babylon is not suitable as a database but can serve as a front-end to a database. We think the project also shows that this is the only way Babylon, or any other expert system shell, can be of use. Data retrieved for a consult should be retrieved from a database, and relevant results computed during a consult should be stored in a database. (We leave other applications, e.g. process control, out of consideration. The non-determinism of production rules may be exploited for some control tasks.) The consult itself can concern diagnosis or planning or any other decision-support function. Viewed in this way, Babylon should stand on an equal footing with front-ends like IAF (a screen form management facility), RPT (a report writer utility) and HLI (a host language interface) of Oracle. In other words, it should provide the facilities to define user-friendly screen-forms to view and update data in a database, compute new information from the data on those forms, and report to the user on this new information.
The conclusions of the project which follow are a summary and extension of the conclusions listed in section 3.2 of Wieringa & Curwiel [1986]. We divide the conclusions into four groups, concerning expert systems in general, the use of Babylon, the feasibility of the expert systems approach in agriculture, and the automation of crop disease diagnosis.

1. EXPERT SYSTEMS

1. What is usually referred to as uncertainty management in expert systems are often inadequate applications of probability theory and statistics. Where no inconsistencies are committed, nothing is added to Bayesian updating. This is argued in Wieringa & Curwiel [1986] section 1.3.

2. Second generation expert systems follow information systems on the road to the three-level ANSI/SPARC structure. They distinguish model level from implementation level. These two levels have as task the accurate representation of reality and the efficient execution of algorithms. What is still missing in expert systems is attention to the user level as something more than a program to produce smart explanations. Heuristics are relevant at the implementation level and at the user level but not at the model level. Expert systems, like any other information system, need a model which accurately describes the logical structure of reality but often they use empirical models as well, which enable one to empirical predict which of the logically possible events will happen next. These models give the expert information system the power of a decision support system which analyses data and advises on future developments.

3. Knowledge acquisition is but one part of system development. There is hardly a body of established practice and principles of knowledge acquisition but there is a large pool of methods, principles and experience with information system development methods. It would be expedient in the development of an expert system to start with a promising information system development method and extend it with whatever is known at the moment about knowledge acquisition.
4. Production rules are to be seen as an extensively tried technique for the implementation of some of the functions that distinguish expert systems from other types of information systems, like diagnosis, classification, planning support and design. They are an implementation mechanism for certain functions on the user level, not for the model on the conceptual level. Production rules are just dynamic decision trees or decision tables and the programming techniques for production systems resemble those for decision tables very much.

5. The modelling techniques of aggregation, specialisation, instantiation and the use of surrogates and of the JSD analysis of the domain in communicating sequential processes can be integrated into a very powerful structure. We have barely scratched the surface of the possibilities of this structure and in the recommendations we will suggest that a follow-up project be initiated to investigate these techniques in-depth.

6. The modelling techniques we explored contain a lot of leads for an efficient implementation. For example, surrogates translate naturally in database keys, chronicles can be implemented as sequential linked lists, events as frame behaviors, aggregation and structured attribute values by referential keys with referential integrity demanded for aggregation and optional for structured values, the specialisation hierarchy can be used for efficient indexing of often-used groups of attributes, and uniquely identifying descriptions can be used for indexing as well. Moreover, the model structure is compatible with the use of content-addressable storage or location addressable storage. Flat files are useful for flexible storage but efficient retrieval can be gained by the use of pointers. Both are compatible with the proposed model structure. This is a great advantage above the relational model structure, which is a simplification of the network and hierarchical structures but does not leave the implementation level very far behind. Relational insert, update, and delete instructions can too easily be viewed as operations on files and the relational algebra does not reflect the queries which are natural to a domain. By putting more of the conceptual structure of the domain in the model, more information is available to create an efficient implementation.
2. BABYLON

Babylon is a very modular system with easy access to the representation languages Lisp, Prolog, and Frames and production rules dialects. It was noted above that production rules are useful on the level of system functions and user interface and not on the model level. Frames are useful on the model level but Babylon does not provide the facilities required from an information system, storage of data between sessions, and query and update of data during sessions. It can therefore only function as a front-end to an information system, providing services as a module to compute decisions with production rules or to compute possible scenarios with its frames processor. Lisp and Prolog are implementation languages with dubious standardisation and portability. The difference between Prolog and a backward chaining production rules set is that Prolog unifies its expressions while the production rule set matches its variables against a working memory, which in this particular case is structured as a collection of frame instances. It is not clear what this difference amounts to in terms of speed of execution, memory requirements, and conceptual clarity.

3. FEASABILITY OF EXPERT SYSTEMS APPROACH IN AGRICULTURE

An expert system is only feasible when it is seen as an information system and developed using the methods and techniques for information system development, possibly extended with methods and techniques to handle datalogical complexity. The virtues of such a system in agriculture would be immediate and continuous availability, interactive advice on the initiative of the farmer, and support for the decisions made in the form of explanations of output, the ability to compute different future scenarios, and the multiple use of data for crop protection, crop management, and farm management.

4. DIAGNOSIS OF PLANT DISEASES

1. The inferences from observations to diseases are implicit, short, and fuzzy. Extension officers find it very hard to explicate these inferences.

2. The concepts used in the description of symptoms are inherently fuzzy. Extension officers find it hard to describe the symptoms without naming the disease. We should expect a similar difficulty with the farmers.

- 135 -
3. The greatest problem in crop disease diagnosis is not knowledge of diagnosis rules but reliable observation. Farmers often have an idea of the type of disease present in a crop but do not know what to look for to make the decision.

4. These points suggest that the function of a diagnosis system should not be the assignment of possible diseases to a collection of reported symptoms, but the verification of a suggested disease class with the data available to the system and the suggestion of further observations to make the decision.

5. The important observations to be made are visual and tactual, while the system only has a linguistic interface. The greatest problem here is what the meaning of observation terms is, e.g. what is "watery green," or "orange," or "brown," etc? Part of the solution to this is the provision of a good collection of photographs of diseases in an early stage of development and from a distance at which the farmer is acquainted with the symptoms. The collection should be integrated in the system in that the dialogue as well as the report should refer to the photographs.

6. Another part of the solution of the observation problem is the design of a good observation discipline which is reflected in the structure of the dialogue as well as the report. The crop protection officers advising the farmer should be of great help here. Diagnosis rules should be drawn up with the help of phytopathologists.

7. A system for crop disease diagnosis is only of use in combination with a system which can provide a spraying advice. The use of the system would not lie in its knowledge of one particular disease, but in the number of diseases it knows about and in the up-to-dateness of its knowledge of current pathogen population in the country and of other phytopathological data. This would also have an educational effect on the farmer.
7.3 RECOMMENDATIONS

The recommendations which are based upon the conclusions of this project concern one research and one development project. The development project is to build a useable system for plant disease diagnosis; the research project is to design a model structure for object-oriented databases.

To place a diagnosis system in a larger framework, there is a need for a decision support system for crop management on farms which supports all decisions which a farmer needs to make during one cultivation cycle of a parcel. One group of decisions in this cycle concerns crop protection, e.g. is there a disease in the crop, what disease is it, is it important, should I spray against the disease before or after it show its symptoms, when and how much should I spray, etc. To support these decisions the system should be able to provide information on the current state, history, and possible futures of the crop and parcel under different scenarios. When the decision to spray against a disease can be made after the disease is observed, the following phases in decision-making can be discerned.

1. Diagnose the observed symptoms. This is carried out by the farmer or a crop protection specialist. The purpose of the FAD project is to automate part of this.

2. Quantify the disease. This must be carried out by the farmer in a systematic way.

3. Predict the actual crop yield and loss starting from the current situation as quantified. This can only be done with the aid of quantitative models.

4. Optimise yield by an optimal spraying advise. This can only be done with the aid of quantitative models.

Work is being done on each of these stages by several departments of the agricultural university. The department of phytopathology is involved in studies of aspects of all stages of crop protection, the department of computer science has been working on an expert system for disease diagnosis, the department of theoretical production ecology works on models for the estimation of crop yield and loss, and the department of mathematics studies the optimisation of spraying advises. These efforts should be integrated by defining a project to build a crop protection subsystem of a crop management system. The project proposal should state how tight or loose the coupling between the different groups of people working on these different aspects of crop protection is, as well as what the relation is with other project
for crop management or farm management systems. Currently, a software company called FytOconsult is constructing a system for farm management. The farm management and crop protection systems need similar data and the farmer should be spared the obligation to enter the same piece of information once for every system he uses.

There is no reason why the development of a crop protection system should not benefit from the body of principles and practical advice present in the branch of computer science studying information system development. There are a number of stages in the development of any information system, including the identification of interest groups, user information needs, interface ergonomics, hardware requirements, and infrastructure for system use and maintenance. These aspects do not receive any attention at the moment and to reach the long-range goal of a crop protection system for real use on the farm they should be attended to.

We therefore recommend the creation of an interdisciplinary team which monitors work currently being done on the automisation of crop protection advice and which possibly initiates new work with a view to implementing a crop protection system. Proposal for a precise form and function for this team is outside the scope of this report, but at least the following points are relevant for the formation of such a team.

1. To be workable, the team should be small, but to have a chance on success, all relevant groups of people should at least express a willingness to provide part of their knowledge and time for the development of a crop protection system. The project definition should strike a balance between these two extremes with which all relevant groups of people agree.

2. Relevant groups of people who should be willing to provide knowledge or dedicate time to the project include representatives from the departments of phytopathology, theoretical production ecology, mathematics and computer science, as well as representatives from the extension service, and the future users.

3. Subtasks for system development include not only the construction of a diagnosis system, models for crop yield and loss estimation, and a module for yield optimisation, but also the identification of user requirements, the design of an infrastructure for system use and maintenance, and the definition and execution of performance tests.
4. The relevance of the different groups of people to the project is:

1. Agricultural scientists and mathematicians provide knowledge of disease diagnosis, yield and loss estimation, and yield optimisation,

2. Representatives of the extension service provide knowledge of the dialects and observation discipline relevant for the user interface as well as the organisational structure for the use and maintenance of the system,

3. The future users provide the knowledge of the dialects and the information requirements which the system should fulfil, and

4. Computer scientists provide the knowledge of development theory and practice of a workable information system.

Exclusion of one of these groups of people from system development could lead to failure.

The first major decision to be made by the team is whether system development and/or system exploitation be funded by government or by a commercial firm. Other decisions, which can be relegated to working groups of one or more people who report to the team, include at least the following.

1. The users of the system can be farmers, extension officers, or schools. Each of these groups have different user requirements.

2. The data available to the system should at least include crop and parcel data but can also include available chemicals, prices, crop prices, present susceptibilities of crops for diseases, current spread of pathogens in the country, etc. The availability of these data increases the quality of the advice given but also increases the complexity of the infrastructure around the system.

3. The user interface includes a simple dialogue and report but these can vary from simple requests and summaries of data to intelligent explanations of advice and the possibility to compute different scenarios for the future under different assumptions.
4. The hardware can consist of microcomputers of different prices and sizes, hand-held terminals, videotex systems, videodisk, books with photographs, etc. These choices should be made in the light of user requirements.

5. The software can consist of a database system, and there is a large choice of programming languages. These choices affect portability, compatibility, price, and hardware requirements of the system.

6. The infrastructure for system marketing, maintenance, and updating should be designed. The farmer will have to follow certain procedures if the system is to be of any use and the benefits of using the system will have to outweigh these efforts on the part of the farmer.

7. Testing procedures will have to be designed. Options are testing by farmers, with attendant agreements about liabilities for crop loss, and testing by schools which teach crop disease diagnosis.

Our second recommendation is to start a research project on the model structure for object-oriented databases. This project could investigate one or more of the following related topics:

1. The integration of the static modelling techniques of aggregation, specialisation and instantiation with a dynamic technique like JSD;

2. The formalisation of this approach in the theory of communicating sequential processes;

3. The design of a formal model description language which allows the designer to express integrity constraints on the data in a consistent manner;

4. The development of an implementation language as an extension of the model description language which exploits the logical structure of the data and the expected use patterns to define an efficient implementation;

5. The design of a user-friendly query language which does justice to the model structure;

6. The design and implementation of a data manipulation language which maintains integrity;
7. The design of a logical structure for an integrated data dictionary which represents the logical structure of the data, their current extent, and their implementation.

This is surely enough to keep one person busy for more than one lifetime, and a definition of a follow-up research project will have to limit itself carefully to what can be achieved in a few years per person. An exact proposal for this project is outside the scope of this report but will follow in due time.
APPENDIX A
REFERENCES

Blokker K.J. [1984]

Boehm C. & G. Jacopini [1966]

Bruyn, L. de [1986]
"De Bruikbaarheid van JSD als Ontwikkelingsmethode van een Expert Systeem voor het Diagnostiseren van Tarweziekten," Computer Science Department, Agricultural University, April 1986.

Butler E.J. & S.G. Jones [1946]

Date C.J. [1981]

Dekker J. [1984]
Fytopathologie (in Dutch), Dept. of Phytopathology, Agricultural University Wageningen, September 1984.

Griethuysen, J.J. van [1982]

Jackson M. [1983]

- 142 -
Oracle [1983]

Oracle [1984]

Simons J.L., R.J. Wieringa [1986]

Smith J.M. & D.C.P. Smith [1977a]

Smith J.M. & D.C.P. Smith [1977b]


Stol W. [1985]

Tanenbaum A.S. [1976]

Wieringa R.J., P.H. Curwiel [1985]

Wieringa R.J., P.H. Curwiel [1986]

Wiese M.V. [1977]
"EPIPRE, A Computer-Based Scheme for Pest and Disease Control
REFERENCES


APPENDIX B
SOURCE LISTING

;;; -*- mode:lisp; syntax:zetalisp -*-

;;; ******************************************************************************
;;; \hspace*{1cm} Lisp functions
;;; ******************************************************************************

(defun multiple-value-string-equal (multiple-value-string string)
  (if (listp multiple-value-string)
    (member string multiple-value-string)
    (string-equal string multiple-value-string)))

(defun retrieve (cultivation-list parcel-id month year)
  ;; Retrieve the cultivation of the parcel PARCEL-ID in MONTH and YEAR from
  ;; CULTIVATION-LIST. If not found, create and store a new one by asking
  ;; SOWING-YEAR, SOWING-DAY and CROP and VARIETY. In both cases, infer the
  ;; growth stage of the crop, ask the user for confirmation, and set the answer
  ;; in DISEASE-OCCURRENCE. The behavior should provide help on the growth
  ;; stage, perhaps with pictorial information and with information on
  ;; the relation between the decimal and the Feekes scale.
  ;; Confirmation from the user is necessary because ageing can occur at
  ;; different and even variable speeds.
  )

;;; ******************************************************************************

(knowledge-base fad3)

;;; ******************************************************************************

- 145 -
;;; parcel aggregation
;;; ************************************************************
(defframe parcel-frame
  (slots (x-coordinate -)
         (y-coordinate -)
         (soil-type -
            :possible-values (:one-of "sandy" "Clay-like"))))

;;; These instantiations should be done once, when the database is initialized,
;;; and then whenever a mutation should be modelled by the database.
(definstance parcel-1 of parcel-frame
  with
  x-coordinate = 20
  y-coordinate = 30
  soil-type = "Sandy")

(definstance parcel-2 of parcel-frame
  with
  x-coordinate = 39
  y-coordinate = 40
  soil-type = "Clay-like")

(definstance parcel-3 of parcel-frame
  with
  x-coordinate = 333
  y-coordinate = 666
  soil-type = "Sandy")

(definstance parcel-4 of parcel-frame
  with
  x-coordinate = 0
  y-coordinate = 0
  soil-type = "Sandy")

;;; ***************************************************************
;;; crop specialisations
;;; ***************************************************************
(defframe crop-frame
  (slots (not-yet-implemented -))) ; inherited slots

(defframe crop-specialisation-frame
  (slots (type "Wheat"
          :possible-values (:one-of "Potatoes"
                            "Wheat"))
        - 146 -)
"Sugarbeet"
"Wheat")

(defun instance crop-specialisation of crop-specialisation-frame)

(defun frame wheat-frame
  (supers crop-frame)
  (slots (not-yet-implemented -))) ; inherited slots

(defun frame wheat-specialisation-frame
  (slots (subtype "Winter wheat"
    :possible-values (:one-of
      "Winter wheat"
      "Summer wheat")))))

(defun frame winter-wheat-frame
  (supers wheat-frame)
  (slots (not-yet-implemented -))) ; inherited slots

(defun frame winter-wheat-specialisation-frame
  (slots (variety -
    :possible-values
      (:one-of
        "Nautica"
        "Citadel"
        "Saiga"
        "Okapi"
        "Marksman"
        "Granada")
    :ask ("%Which variety of winter wheat are you growing on this field? ")))

(defun instance winter-wheat of winter-wheat-frame)

(defun instance winter-wheat-specialisation of winter-wheat-specialisation-frame)

(defun frame cultivation-frame
  (slots (month -)
    (year -)) ; time stamp
    ; time stamp
;;; These instantiations are supposed to be the result of previous cultivations.

(definstance cultivation-1 of cultivation-frame
  with
  parcel-id = parcel-1
  sowing-day = 300
  sowing-year = 1985
  crop = winter-wheat)

(definstance cultivation-2 of cultivation-frame
  with
  parcel-id = parcel-1
  sowing-day = 300
  sowing-year = 1984
  crop = winter-wheat)

(definstance cultivation-3 of cultivation-frame
  with
  parcel-id = parcel-1
  sowing-day = 300
  sowing-year = 1983
  crop = winter-wheat)

;;; **************************************************************************
;;;  weather aggregation
;;; **************************************************************************

;;; This should be a weather chronicle in the database.
;;; The tuples should be dynamically instantiated by a perpetual
;;; process connected to a local weather station.

(defframe weather-frame
  (slots (day -) ; time stamp
         (month -) ; time stamp
         (year -) ; time stamp
         (temperature -) ; degrees centigrade
  )
  - 148 -
)
(radiation -) ; hours sunshine
(rainfall -)) ; mm / 24 hrs

(defbehavior (weather-frame :after :initialize) (with-specification)
  (multiple-value-bind (sec min hours day month year day-of-the-week)
      (time:get-time)
        (<- self :set 'day day)
        (<- self :set 'month month)
        (<- self :set 'year year))
        (<- self :set 'temperature 'temp-measurement-from-weather-station)
        (<- self :set 'rainfall 'rainfall-measurement-from-weather-station)
        (<- self :set 'radiation 'radiation-measurement-from-weather-station))

;;; This instantiation should be done whenever a new measurement from the
;;; weather station is available.

(defun weather of weather-frame)

;;; ############################################################################
;;; disease-cause specialisations
;;; ############################################################################

(defframe disease-cause-frame
  (slots (systematic-name -)
         (vernacular -)))

;;; ############################################################################

(defframe septoria-tritici-frame
  (supers disease-cause-frame)
  (slots (leaf-spots-color #A((leaf-spots color) :get-indirect nil)
      :possible-values (:one-of "yellow" "Ochre" "Brown"
                        "Grey" unknown)
      :expected-value "Yellow")
  (leaf-spores #A((leaf-spots spores) :get-indirect nil)
      :possible-values (:one-of "Present" "Absent" unknown)
      :expected-value "Present")
  (leaf-spores-color #A((leaf-spots spores-color) :get-indirect nil)
      :possible-values (:one-of "Brown" "Black")
      :expected-value "Black"))

(defbehavior (septoria-tritici-frame :report)()
  (format fl "-%")
  (format fl "% -lOTSymptoms of Septoria tritici: ")
  (format fl "-%")

- 149 -
(format fl "-% ~10T                             ~42<Observed";Value";Expected value">")
(if (eq ($value leaf-spots-color) 'unknown)
  ;; then
  (progn
    (format fl "-% ~10TColor of leaf spots                     ~35<no"; "A">"
      (<- self :get 'leaf-spots-color :expected-value))
  ;; else
  (progn
    (format fl "-% ~10TColor of leaf spots                     ~35<yes"; "A"; ~>"
      (<- self :get 'leaf-spots-color))))
(if (eq ($value leaf-spores) 'unknown)
  ;; then
  (progn
    (format fl "-% ~10TPresence of spores on the leaves         ~35<no">")
  ;; else
  (progn
    (format fl "-% ~10TPresence of spores on the leave          ~35<yes"; "A"; ~>"))(<- self :get 'leaf-spores))
(if (eq ($value leaf-spores-color) 'unknown)
  ;; then
  (progn
    (format fl "-% ~10TColor of spores on the leave             ~35<no"; "A">"
      (<- self :get 'leaf-spores-color :expected-value))
  ;; else
  (progn
    (format fl "-% ~10TColor of spores on the leaves            ~35<yes"; "A"; ~>")(<- self :get 'leaf-spores-color)))))))

(defunseptoria-tritici of septoria-tritici-frame
  with
  systematic-name = "Septoria tritici"
  vernacular = "Septoria leaf spot")

;;; ******************************
defframe septoria-nodorum-frame
  (supers disease-cause-frame)
  (slots (leaf-spots-color #A((leaf-spots color) :get-indirect nil)
    :possible-values (:one-of "yellow" "Ochre" "Brown" "Grey" unknown)
    :expected-value "Yellow")
  (leaf-spores #A((leaf-spots spores) :get-indirect nil)
    :possible-values (:one-of "Present" "Absent" unknown)
    - 150 -
(leaf-spores-color #A((leaf-spots spores-color) :get-indirect nil)
 :possible-values (:one-of "Brown" "Black" unknown)
 :expected-value "Brown"))

(defbehavior (septoria-nodorum-frame :report)()

  (format fl "-%
  (format fl "-% "10TSymptoms of Septoria nodorum: ")
  (format fl "-% "10T "42<Observed”;Value”;Expected value”>")
  (if (eq ($value leaf-spots-color) 'unknown)
       ;; then
       (progn
         (format fl "-% "10T Color of leaf spots
                    "35<no”; ";”A”; ”>
                    (<- self :get 'leaf-spots-color :expected-value)))
       ;; else
       (progn
         (format fl "-% "10T Color of leaf spots
                    "35<yes”; ”A”; ”>
                    (<- self :get 'leaf-spots-color))))
  (if (eq ($value leaf-spores) 'unknown)
       ;; then
       (progn
         (format fl "-% "10T Presence of spores on the leaves
                    "35<no”))
       ;; else
       (progn
         (format fl "-% "10T Presence of spores on the leaves
                    "35<yes”; ”A”; ”>
                    (<- self :get 'leaf-spores))
  (if (eq ($value leaf-spores-color) 'unknown)
       ;; then
       (progn
         (format fl "-% "10T Color of spores on the leaves
                    "35<no”; ”;”A”; ”>
                    (<- self :get 'leaf-spores-color :expected-value)))
       ;; else
       (progn
         (format fl "-% "10T Color of spores on the leaves
                    "35<yes”; ”A”; ”>
                    (<- self :get 'leaf-spores-color))))))

(definst septoria-nodorum of septoria-nodorum-frame
  with
  systematic-name = "Septoria nodorum"
  vernacular = "blotch Glume")

- 151 -
(defframe gaumannomyces-graminis-frame
  (supers disease-cause-frame)
  (slots (root-mycelium-color #A((root-mycelium color) :get-indirect nil)
    :possible-values (:one-of "Brown" "Black" unknown)
    :expected-value "Brown")
  (root-mycelium-shape #A((root-mycelium shape) :get-indirect nil)
    :possible-values (:one-of "Hyphae" unknown)
    :expected-value "Hyphae"))

(defbehavior (gaumannomyces-graminis-frame :report)()
  (format fl
    "-%
    -lOTSymptoms of Gaumannomyces graminis: 
    -lOT
    -l42<0bserved-;Value-;Expected value->")
  (if (eq ($value root-mycelium-color) 'unknown)
    ;; then
    (progn
      (format fl
        "-% -lOTColor of root mycelium -35<no-;A->"
      )
    ;; else
    (progn
      (format fl
        "-% -lOTColor of root mycelium -35<yes-;A-;->"
      )
    )
  )
  (if (eq ($value root-mycelium-shape) 'unknown)
    ;; then
    (progn
      (format fl
        "-% -lOTShape of the mycelium on the roots -35@<no->")
    )
  ;; else
  (progn
    (format fl
      "-% -lOTShape of the mycelium on the roots -35<yes-;A-;->"
    )
  )))

(definstance gaumannomyces-graminis of gaumannomyces-graminis-frame
  with
    systematic-name = "Gaumannomyces Graminis"
    vernacular = "Take-all")

(defframe erysiphe-graminis-frame
  (supers disease-cause-frame)
  - 152 -
(slots (leaf-mycelium-shape #A((leaf-mycelium shape) :get-indirect nil)
 :possible-values (:one-of "Hyphae" "Stripes" "Patches" unknown)
 :expected-value "Patches")
(leaf-mycelium-color #A((leaf-mycelium color) :get-indirect nil)
 :expected-value "White,Brown,Black")
(leaf-mycelium-spores #A((leaf-mycelium spores) :get-indirect nil)
 :expected-value "n/a")
(leaf-mycelium-spores-color #A((leaf-mycelium spores-color)
 :get-indirect nil)
 :expected-value "Brown,Black"))

(defbehavior (erysiphe-graminis-frame :report)()
 (format fl ""%")
 (format fl ""10T Symptoms of Erysiphe graminis: ")
 (format fl "")
 (format fl ""10T Observed;Value;Expected value"")
 (if (eq ($value leaf-mycelium-color) 'unknown)
     ;; then
     (progn
       (format fl ""10T Color of leaf mycelium
          "no"; "A""
       (<- self :get 'leaf-mycelium-color :expected-value)))
     ;; else
     (progn
       (format fl ""10T Color of leaf mycelium
          "yes"; "A""
       (<- self :get 'leaf-mycelium-color)))
   (if (eq ($value leaf-mycelium-shape) 'unknown)
       ;; then
       (progn
         (format fl ""10T Shape of the mycelium on the leaves
            "no"
         (<- self :get 'leaf-mycelium-shape)))
       ;; else
       (progn
         (format fl ""10T Shape of the mycelium on the leaves
            "yes"
         (<- self :get 'leaf-mycelium-shape)))
   (if (eq ($value leaf-mycelium-spores) 'unknown)
       ;; then
       (progn
         (format fl ""10T Presence of spores in the mycelium
            "no"
       (<- self :get 'leaf-mycelium-spores)))
       ;; else
       (progn
         (format fl ""10T Presence of spores in the mycelium
            "yes"
       (<- self :get 'leaf-mycelium-spores)))
   (if (eq ($value leaf-mycelium-spores-color) 'unknown)
       ;; then
       (progn
         (format fl ""10T Color of spores in the mycelium
            "no"
       (<- self :get 'leaf-mycelium-spores-color)))
       ;; else
       (progn
         (format fl ""10T Color of spores in the mycelium
            "yes"
       (<- self :get 'leaf-mycelium-spores-color))))

((- self :get 'leaf-mycelium-spores-color :expected-value)))
;; else
(progn
(format fl "-% ~10TYColor of spores in the mycelium ~35<yes~;"A~; ~>" 
((- self :get 'leaf-mycelium-spores-color)))))))

(definstance erysiphe-graminis of erysiphe-graminis-frame 
with
  systematic-name = "Erysiphe graminis F.sp. tritici" 
  vernacular = "Powdery mildew")

;;; *****************************/
fusarium-culmorum-frame
  (supers disease-cause-frame)
  (slots (foot-stripes-color #A((foot-stripes color) :get-indirect nil) 
      :possible-values (:one-of "Orange" "Brown" 
      "Black" unknown) 
      :expected-value "Brown,Black") 
  (foot-dicolorisation-color #A((foot-dicolorisation color) 
      :get-indirect nil) 
      :possible-values (:one-of "Orange" "Brown" 
      "Black" unknown) 
      :expected-value "Brown"))

(defbehavior (fusarium-culmorum-frame :report)() 
  (format fl ""m") 
  (format fl ""m ~10TSymptoms of Fusarium culmorum : ") 
  (format fl ""m") 
  (format fl ""m ~42<Observed~;Value~;Expected value~>"") 
  (if (eq ($value foot-stripes-color) 'unknown) 
      ;; then
      (progn
      (format fl ""m ~10TColor of foot stripes ~39<no~; ~;"A~; ~>
      ( (- self :get 'foot-stripes-color :expected-value)) 
      ;; else
      (progn
      - 154 -
(format fl "~% "10TColor of foot stripes
    (~35<yes";"A"; ~")
    (~<self :get 'foot-stripes-color)))
(if (eq ($value foot-discolorisation-color) 'unknown)
    ;; then
    (progn
        (format fl "~% "10TColor of the foot discolorisation
            (~35@<no">")
    ;; else
    (progn
        (format fl "~% "10TColor of the foot discolorisation
            (~35<yes";"A"; ~")
    ))
)
(definstance fusarium-culmorum of fusarium-culmorum-frame
    with
    systematic-name = "Fusarium culmorum"
    vernacular = "Foot rot")

;;; *****************************************************************************

(defframe sitobion-avenae-frame
    (supers disease-cause-frame)
    (slots (leaf-bugs-color #A((leaf-bugs color) :get-indirect nil)
        :possible-values (:one-of "Green" "Dark green"
            "Light green" "Brown" unknown)
        :expected-value "Green, Dark green, Light green")
    (leaf-bugs-shape #A((leaf-bugs shape) :get-indirect nil)
        :possible-values (:one-of "Squat" "Oblong" unknown)
        :expected-value "Squat")
    (leaf-bugs-siphons #A((leaf-bugs siphons) :get-indirect nil)
        :possible-values (:one-of "White" "Green"
            "Light-green" "Dark-green" "Black" unknown)
        :expected-value "Black")
    ))

(defbehavior (sitobion-avenae-frame :report)()
    (format fl "~%")
    (format fl "~% "10TSymptoms of Sitobion avenae:")
    (format fl "~%")
    (format fl "~% "10T"42<Observed”;Value”;Expected value”>")
    (if (eq ($value leaf-bugs-color) 'unknown)
        ;; then
        (progn
            (format fl "~% "10TColor of leaf bugs
                (~54<no"; "A";")
                (~<self :get 'leaf-bugs-color :expected-value)))
        ;; else

- 155 -
(progn
  (format f1 "%-10TColor of leaf bugs -35<yes_;_A_;_">"
  (<- self :get 'leaf-bugs-color)))
  (if (eq ($value leaf-bugs-shape) 'unknown)
    ;; then
    (progn
      (format f1 "%-10TShape of the bugs on the leaves -35<no~>")
    );; else
    (progn
      (format f1 "%-10TShape of the bugs on the leaves -35<yes_;_A_;_">"
      (<- self :get 'leaf-bugs-shape)))
  ));
  (if (eq ($value leaf-bugs-siphons) 'unknown)
    ;; then
    (progn
      (format f1 "%-10TColor of siphons of the bugs -35<no-;_;A-> 11
      (<- self :get 'leaf-bugs-siphons :expected-value)))
    );; else
    (progn
      (format f1 "%-10TColor of siphons of the bugs -35<yes_;_A_;_">"
      (<- self :get 'leaf-bugs-color))))

(definistance sitobion-avenae of sitobion-avenae-frame
  with
  systematic-name = "Sitobion avenae"
  vernacular = "English grain aphid")

;;;;; ******************************************************************************

(defframe metopolophium-dirhodum-frame
  (supers disease-cause-frame)
  (slots (leaf-bugs-color #A((leaf-bugs color) :get-indirect nil)
    :possible-values (:one-of "Green" "Dark green"
    "Light green" "Brown" unknown)
    :expected-value "Green,Lightgreen")
  (leaf-bugs-shape #A((leaf-bugs shape) :get-indirect nil)
    :possible-values (:one-of "Squat" "Oblong" unknown)
    :expected-value "Oblong")
  (leaf-bugs-siphons #A((leaf-bugs siphons) :get-indirect nil)
    :possible-values (:one-of "White" "Green"
    "Light-green" "Dark-green" "Black" unknown)
    :expected-value "White,Green")
  )

(defbehavior (metopolophium-dirhodum-frame :report)())
(format fl "-%")
(format fl "-% "10TSymptoms of Metopolophium dirhodum: ")
(format fl "-%")
(format fl "-% "42<Observed~;Value~;Expected value~">")
(if (eq ($value leaf-bugs-color) 'unknown)
  ;; then
  (progn
    (format fl "-% "10TColor of leaf bugs
                ~44<no~; "~;"A~">"
                (<- self :get 'leaf-bugs-color :expected-value)))
  ;; else
  (progn
    (format fl "-% "10TColor of leaf bugs
                ~35<yes~;"A~; ~>"
                (<- self :get 'leaf-bugs-color)))
  (if (eq ($value leaf-bugs-shape) 'unknown)
      ;; then
      (progn
        (format fl "-% "10TShape of the bugs on the leaves
                   ~35<no~; "~;"A~">")
        ;; else
        (progn
          (format fl "-% "10TShape of the bugs on the leaves
                     ~35<yes~;"A~; ~>"
                     (<- self :get 'leaf-bugs-shape)))
      (if (eq ($value leaf-bugs-siphons) 'unknown)
          ;; then
          (progn
            (format fl "-% "10TColor of siphons of the bugs
                       ~39<no~; "~;"A~">"
                       (<- self :get 'leaf-bugs-siphons :expected-value)))
          ;; else
          (progn
            (format fl "-% "10TColor of siphons of the bugs
                       ~35<yes~;"A~; ~>"
                       (<- self :get 'leaf-bugs-color))))
    (definstance metopolophium-dirhodum of metopolophium-dirhodum-frame
     with
     systematic-name = "Metopolophium dirhodum"
     vernacular = "Rosegrass aphid")

    ;;;****************************************************************************
    (defframe pseudocercosporella-herpotrichoides-frame
     (supers disease-cause-frame)
     (slots (foot-spots-borders #A((foot-spots well-defined-borders)
                                  :get-indirect nil)
            :possible-values (:one-of yes no unknown)
            :expected-value no)

     - 157 -
(foot-spots-mycelium #A((foot-spots mycelium) :get-indirect nil)
 :possible-values (:one-of yes no unknown)
 :expected-value "n/a")

(foot-spots-mycelium-color #A((foot-spots mycelium-color)
 :get-indirect nil)
 :possible-values (:one-of "Brown" "Black" "Purple"
 unknown)
 :expected-value "Black"))

(defbehavior ( pseudocercosporella-herpotrichoides-frame :report)()
 (format fl "")
 (if (eq ($value foot-spots-borders) 'unknown)
 (progn
 (format fl "~10TSharp borders of foot spots ~35<no>; ~;"A~>"
 (<-self :get 'foot-spots-borders :expected-value)))
 (progn
 (format fl "~10TSharp borders of foot spots ~35<yes>; "A; ~>"
 (<-self :get 'foot-spots-borders)))
 (if (eq ($value foot-spots-mycelium) 'unknown)
 (progn
 (format fl "~10TPresence of mycelium at the foot ~35<no>-"
 (<-self :get 'foot-spots-mycelium)))
 (progn
 (format fl "~10TPresence of mycelium at the foot ~35<yes>";"A","~>"
 (<-self :get 'foot-spots-mycelium))
 (if (eq ($value foot-spots-mycelium-color) 'unknown)
 (progn
 (format fl "~10TColor of mycelium at the foot ~35<no>";","A","~>"
 (<-self :get 'foot-spots-mycelium-color))
 (progn
 (format fl "~10TColor of mycelium at the foot ~35<yes>";"A","~>"
 (<-self :get 'foot-spots-mycelium-color))))))))

(defvar pseudocercosporella-herpotrichoides-frame)

(definstance pseudocercosporella-herpotrichoides of
 pseudocercosporella-herpotrichoides-frame
 with
 systematic-name = "Pseudocercosporella herpotrichoides"
- 158 -
vernacular = "Eyespot")

;;; ******************************************************************************
(defun frame corticium-solani-frame
  (supers disease-cause-frame)
  (slots (foot-spots-borders #A((foot-spots well-defined-borders)
     :get-indirect nil)
     :possible-values (:one-of yes no unknown)
     :expected-value yes)
  (foot-spots-mycelium #A((foot-spots mycelium) :get-indirect nil)
     :possible-values (:one-of yes no unknown)
     :expected-value "n/a")
  (foot-spots-mycelium-color #A((foot-spots mycelium-color)
     :get-indirect nil)
     :possible-values (:one-of "Brown" "Black" "Purple"
      unknown)
     :expected-value "Black,Purple"))

(defunbehavior (corticium-solani-frame :report)()
  (format fl "-% OtSymptoms of Corticium solani: ")
  (format fl "-% Ot
  (if (eq ($value foot-spots-borders) 'unknown)
    ;; then
    (progn
      (format fl "-% OtSharp borders of foot spots ~35<no"; ~;"A~")
      (<-self :get 'foot-spots-borders :expected-value)))
    ;; else
    (progn
      (format fl "-% OtSharp borders of foot spots ~35<yes"; "A"; ~")
      (<-self :get 'foot-spots-borders)))
  (if (eq ($value foot-spots-mycelium) 'unknown)
    ;; then
    (progn
      (format fl "-% OtPresence of mycelium at the foot ~35<no"))
    ;; else
    (progn
      (format fl "-% OtPresence of mycelium at the foot ~35<yes"; "A"; ~")
      (<-self :get 'foot-spots-mycelium))
  (if (eq ($value foot-spots-mycelium-color) 'unknown)
    ;; then
    (progn
      (format fl "-% OtColor of mycelium at the foot ~40<no"; ~;"A~")
      (<-self :get 'foot-spots-mycelium-color :expected-value)))
  ;;;;

- 159 -
;;; else
(progn
  (format fl ""% ~10TColor of mycelium at the foot ~35<yes”;”~A”; “~>
  (<- self :get 'foot-spots-mycelium-color)))
))

(defun instance cortici-solani of cortici-solani-frame
  with
  systematic-name = "Corticium solani"
  vernacular = "Sharp Eyespot")

;;;******************************************************************************
(defun frame pucinia-striiformis-frame
  (supers disease-cause-frame)
  (slots (leaf-spores-shape #A((leaf-spores shape) :get-indirect nil)
    :possible-values (:one-of "Stripes" "Patches"
    unknown)
    :expected-value "Stripes")
  (leaf-spores-color #A((leaf-spores color) :get-indirect nil)
    :possible-values (:some-of
      "Yellow"
      "Pink"
      "Yellow-brown"
      "Brown"
      "Dark-brown"
      "Orange"
      "Orange-brown"
      "Purple"
      "Black"
      unknown)
    :expected-value "Yellow,Yellow-brown,Brown")
))

(defun behavior (pucinia-striiformis-frame :report)()
  (format fl "%")
  (format fl "% ~10TSymptoms of Pucinia striiformis: ~")
  (format fl "%")
  (format fl "% ~10T ~42<Observed”;”Value”;”Expected value”">")
  (if (eq ($value leaf-spores-shape) 'unknown)
    ;; then
    (progn

(format f1 "%-10TShape of leaf spots ~35<no"; ~;"A">"
   (<- self :get 'leaf-spores-shape :expected-value)))

;; else
(progn
  (format f1 "%-10TShape of leaf spots ~35<yes"; "A"; ~""
        (<- self :get 'leaf-spores-shape)))

(if (eq ($value leaf-spores-color) 'unknown)
  ;; then
  (progn
    (format f1 "%-10TColor of the spots on the leaves ~53<no"; ~;"A">"
          (<- self :get 'leaf-spores-color :expected-value)))
  ;; else
  (progn
    (format f1 "%-10TColor of the spots on the leaves ~35<yes"; "A"; ~""
          (<- self :get 'leaf-spores-color))))

(defvar pucinia-striiformis-frame
  (definstance pucinia-striiformis-of-pucinia-striiformis-frame
    with
      systematic-name = "Pucinia striiformis"
      vernacular = "Yellow rust")

;;; ****************************************************************************

(defvar pucinia-recondita-frame
  (defframe pucinia-recondita-frame
    (supers disease-cause-frame)
    (slots (leaf-spores-shape #A((leaf-spores shape) :get-indirect nil)
      :possible-values (:one-of
        "Stripes"
        "Patches"
        unknown)
      :expected-value "patches")
    (leaf-spores-epidermis #A((leaf-spores epidermis) :get-indirect nil)
      :possible-values (:one-of yes no unknown)
      :expected-value yes)
    (leaf-spores-color #A((leaf-spores color) :get-indirect nil)
      :possible-values (:one-of
        "Yellow"
        "Pink"
        "Yellow-brown"
        "Brown"
        "Dark-brown"
        "Orange"
        "Orange-brown"
        "Purple"
        "Black"
(defbehavior (pucinia-recondita-frame :report)()
  (format fl ""%""
  (format fl ""%"" "10TSymptoms of Pucinia recondita: ")
  (format fl ""%"" "~42<Observed"';Value"';Expected value">"'))
(if (eq ($value leaf-spores-shape) 'unknown)
;; then
  (progn
    (format fl ""%"" "~35<no"'; ";'A"'>
      (<-self :get 'leaf-spores-shape :expected-value)))
;;; else
  (progn
    (format fl ""%"" "~35<yes"'; ";'A"'; >")
    (if (eq ($value leaf-spores-epidermis) 'unknown)
      ;; then
        (progn
          (format fl ""%"" "~35@<no">")
          (progn
            (format fl ""%"" "~35<yes';"';A'");
            (<-self :get 'leaf-spores-epidermis))))
      ;; else
        (progn
          (format fl ""%"" "~35<yes"'; ";'A"'; >")
          (if (eq ($value leaf-spores-color) 'unknown)
            ;; then
              (progn
                (format fl ""%"" "~63<no"'; ";'A"'>
                  (<-self :get 'leaf-spores-color :expected-value)))
            ;; else
              (progn
                (format fl ""%"" "~35<yes"'; ";'A"'; >")
                (<-self :get 'leaf-spores-color))))))

(deffunction pucinia-recondita-frame of pucinia-recondita-frame
  with
    systematic-name = "Pucinia recondita"
    vernacular = "Brown rust")

;;; ********************************************************************************
;;; symptoms-occurrence aggregation
;;; - 162 -
(defframe symptoms-ocurrence-frame
  (slots (symptoms-list #A(nil nil :add-symptoms-to-list))))

(deffbehavior (symptoms-ocurrence-frame :add-symptoms-to-list)
  (new-value active-value prop slot)
  (if (not (member new-value ($value symptoms-list)))
   (cons new-value ($value symptoms-list))
   ;;else
   ($value symptoms-list)))

(deffbehavior (symptoms-ocurrence-frame :report)()
  (if (eq ($value symptoms-list) nil)
    ;; then
    (format f1 "~% ~10TNo symptoms observed. ")
    ;; else
    (format f1 "~%")
    (format f1 "~% ~10TThe symptoms are : ")
    (format f1 "~% ~10T------------------- ")
    (format f1 "~%")
    (loop for symptom1 in ($value symptoms-list)
      do
      (format fl (<- symptom1 :get 'text))
      (format:print-list fl "-A" (<- symptom1 :get 'supported-diseases))
      (loop for symptom2 in (<- symptom1 :slots)
        do
        (if (and (not (eq (<- symptom1 :get symptom2) 'unknown))
          (not (eq (<- symptom1 :get symptom2 :text) nil)))
          ;; then
          (progn
            (format fl (<- symptom1 :get symptom2 :text)
              (<- symptom1 :get symptom2))
            (format:print-list fl "-A" (<- symptom1 :get symptom2
              :supported-diseases))))))))

;;; ****************************
;;; disease-occurrence aggregation
;;; ****************************

(defframe disease-occurrence-frame
  (slots (day -) (month -)
    ; time stamp ; time stamp
    - 163 -
  )
(year -); time stamp
(parcel-id -)
  :possible-values #A(- :possible-parcel nil)
  :ask "What parcel do you want to consult about?"
(growth-stage 0); not yet implemented
(possible-disease-causes #A(nil nil :add-disease-cause-to-list))
(plant-symptoms-slot -
  :possible-values (:instance-of plant-symptoms-frame)))

;;; (defbehavior (disease-occurrence-frame :possible-parcel)
;;;   (new-value active-value prop slot)
;;;   (cons :one-of
;;;     (delq nil (mapcar #'cadr
;;;     (mapcar #'car parcel-aggregate)))))

(defbehavior (disease-occurrence-frame :possible-parcel)
  (new-value active-value prop slot)
  (cons :one-of
    (delq nil (get-instance-list 'parcel-frame)))))

(defbehavior (disease-occurrence-frame :add-disease-cause-to-list)
  (new-value active-value prop slot)
  (cons new-value ($value possible-disease-causes)))

(defbehavior (disease-occurrence-frame :instantiate)()
  ;; Get current DAY, MONTH, YEAR.
  (multiple-value-bind (sec min hours day month year day-of-the-week)
    (time:get-time)
      (<- self :set 'day day)
      (<- self :set 'month month)
      (<- self :set 'year year))

  ;; Get PARCEL-ID. An ask-helping should be used which retrieves the list of
  ;; parcel-id’s from the database and lets the user choose from it.
  (<- self :ask 'parcel-id)

  ;; Infer cultivation and create one if necessary.
  (retrieve (get-instance-list 'cultivation-frame) parcel-id month year)

  ;; Fill SYMPTOMS.
  (defineinstance plant-symptoms of plant-symptoms-frame)
  (<- self :put 'plant-symptoms-slot plant-symptoms)
  ;; Dialogue to fill all slots in SYMPTOMS and its components.
  (find-implications :dialogue :do-all)

  ;; Apply integrity constraints to determine POSSIBLE-DISEASE-CAUSES
(septoria-tritici)
(septoria-nodorum)
(gaumannomyces-graminis)
(erysiphe-graminis)
(fusarium-culmorum)
(sitobion-avenae)
(metopolophium-dirhodum)
(pseudocercosporella-herpotrichoides)
(corticium-solani)
(pucinia-striiformis)
(pucinia-recondita))

(defun behavior (disease-occurrence-frame :report)()
  (if (eq ($value possible-disease-causes) nil)
      ;; then
      (progn
        (format t "-%")
        (format t "-% -10TNo disease cause inferred."))
      ;; else
      (format t "-% -lOTPossible disease causes are :")
      (loop for disease in ($value possible-disease-causes)
        do
        (<- disease :report))))

(defun instance disease-occurrence of disease-occurrence-frame)

;;; ****************************************************************************
;;; symptom specialisation
;;; ****************************************************************************

;;; within the instances and slots of the symptoms we need behaviors
;;; to put a supported disease in lists, which will be printed by the report
;;; generator. These behaviors will be defined for a frame DISEASE-LIST-HANDLER.
;;; This frame will be a super for all the symptoms frames.

(defun frame disease-list-handler
  (slots (dummy -))

(defun behavior (disease-list-handler :add-disease-to-slot-list)
  (new-value active-value property slot)
  (cons new-value (<- self :get slot property)))

(defun behavior (disease-list-handler :add-disease-to-instance-list)
  (new-value active-value property slot)
(cons new-value ($)value supported-diseases))

;;; ****************************
;;; plant symptoms specialisation
;;; ****************************

(defframe plant-symptoms-frame
  (slots (specialisations #A(- nil :put-specialisations)
    :possible-values (:some-of
      "Symptoms on the root"
      "Symptoms on the foot"
      "Symptoms on the leaf"
    )
    :ask("%On which plant part(s) are the symptoms located? ")
  )
  (root-symptoms-slot unknown
    :possible-values (:instance-of root-symptoms-frame)
  )
  (foot-symptoms-slot unknown
    :possible-values (:instance-of foot-symptoms-frame)
  )
  (leaf-symptoms-slot unknown
    :possible-values (:instance-of leaf-symptoms-frame)
  )
)

(defbehavior (plant-symptoms-frame :put-specialisations)
  (new-value active-value prop slot)
  ;; Take care of dialogue structure. OBSERVED is used by later rules
  ;; in rule-set :DIALOGUE.
  (cond-every
    ((multiple-value-string-equal new-value "Symptoms on the root")
      (definstance root-symptoms of root-symptoms-frame)
      (<-self :put 'root-symptoms-slot root-symptoms))
    ((multiple-value-string-equal new-value "Symptoms on the foot")
      (definstance foot-symptoms of foot-symptoms-frame)
      (<-self :put 'foot-symptoms-slot foot-symptoms))
    ((multiple-value-string-equal new-value "Symptoms on the leaf")
      (definstance leaf-symptoms of leaf-symptoms-frame)
      (<-self :put 'leaf-symptoms-slot leaf-symptoms)))
  new-value)

;;; ****************************
;;; Root symptoms observation
;;; ****************************

(defframe root-symptoms-frame

- 166 -
(slots (mycelium #A(- nil :put-mycelium)
    :possible-values (:one-of
       yes
       no
       unknown)
    :ask ("%%Did you observe mycelium on the roots ? "))
(root-mycelium-slot unknown
    :possible-values (:instance-of
       root-mycelium-frame))
(break-off -
    :possible-values (:one-of yes no unknown)
    :ask ("%%Do the roots break off easily ?")))

(defun behavior (root-symptoms-frame :put-mycelium)
  (new-value active-value prop slot)
  (cond-every
    ((multiple-value-string-equal new-value 'yes)
     (definstance root-mycelium of
       root-myceli~frame)
     (<-self :put 'root-mycelium-slot root-mycelium)))
  new-value)

;;; ***************************************************
;;; Root mycelium observation
;;; ***************************************************

(defun frame root-mycelium-frame
  (supers disease-list-handler)
  (slots (color -
    :possible-values (:some-of
       "Brown"
       "Black"
       unknown)
    :ask ("%%What is the color of the mycelium on the roots ? ")
    :text "%%%20TObservation of "A mycelium on the roots: 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
(shape -
    :possible-values (:one-of "hyphae" "Patches" unknown)
    :ask ("%%What is the shape of the mycelium? ")
    :text "%%%20TMycelium on the roots occurs in the shape of
      Hyphae: 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
(length -
    :possible-values (:one-of "normal" "abnormal" unknown)
    :ask ("%%What can you say about the length of the roots ?"))
    - 167 -
(supported-diseases #A(nil nil :add-disease-to-instance-list))
(text """201Mycelium on the roots: ")

;;; ********************
;;; Foot symptoms observation
;;; ********************

(defframe foot-symptoms-frame
  (slots (specialisations #A(- nil :put-specialisations)
    :possible-values (:some-of
                      "Stripes"
                      "Discolorisation"
                      "Spots"
                      unknown)
    :ask ("""What kind of foot symptoms did you observe? ")
    (foot-stripes-slot unknown :possible-values (:instance-of foot-stripes-frame))
    (foot-discolorisation-slot unknown :possible-values (:instance-of foot-discolorisation-frame))
    (foot-spots-slot unknown :possible-values (:instance-of foot-spots-frame))
  ))

(defbehavior (foot-symptoms-frame :put-specialisations)
  (new-value active-value prop slot)
  (cond-every
    ((multiple-value-string-equal new-value "Stripes")
      (definstance foot-stripes of foot-stripes-frame)
      (<- self :put 'foot-stripes-slot foot-stripes))
    ((multiple-value-string-equal new-value "Discolorisation")
      (definstance foot-discolorisation of foot-discolorisation-frame)
      (<- self :put 'foot-discolorisation-slot foot-discolorisation))
    ((multiple-value-string-equal new-value "Spots")
      (definstance foot-spots of foot-spots-frame)
      (<- self :put 'foot-spots-slot foot-spots))
    new-value)

;;; ********************
;;; Foot stripes observation
;;; ********************

(defframe foot-stripes-frame
  - 168 -
(supers disease-list-handler)
(slots (color -
  :possible-values (some-of
  "Orange"
  "Brown"
  "Black"
  unknown)
  :ask ("%What is the color of the stripes at the foot of the plant? ")
  :text "%-%Observation of %A stripes on the foot: 
  :supported-diseases #A(nil nil :add-disease-to-slot-list))
)
:possible-values (:one-of yes no unknown)
:ask ("% Do you see mycelium patches in the centre of the spots ?")
:after-asking ((if (= yes) ($ask (foot-spots mycelium-color))))
:text "% ~20T Presence of mycelium patches in the spots at the foot:"
:supported-diseases #A(nil nil :add-disease-to-slot-list))
(mycelium-color -
 :possible-values (:one-of "Black"
 "Brown"
 "Purple"
 unknown)
:ask ("% What is the color of the foot-mycelium ?")
:text "% ~20T Observation of ~A mycelium patches in the spots at the foot:"
:supported-diseases #A(nil nil :add-disease-to-slot-list))
(supported-diseases #A(nil nil :add-disease-to-instance-list))
(text "% ~20T Spots at the foot: ")))

;;; ****************************
;;; Leaf symptoms observation
;;; ****************************

;;; This could contain an indication whether the symptoms are observed on the
;;; upper or lower leaves. For the time being, we disregard this topological
;;; information.

;;; The SPECIALISATIONS slot is shared by all symptoms specialisation frames but
;;; should not be inherited. It cannot be inherited because it is active.

(defframe leaf-symptoms-frame
(slots (specialisations #A(- nil :put-specialisations)
 :possible-values (:some-of "Leaf spots"
 "Leaf mycelium"
 "Leaf spores"
 "Bugs"
 unknown)
 :ask ("% What kind of leaf symptoms did you observe? ")))
(leaf-spots-slot unknown
 :possible-values (:instance-of leaf-spots-frame))
(leaf-mycelium-slot unknown
 :possible-values (:instance-of leaf-mycelium-frame))
(leaf-spores-slot unknown
 :possible-values (:instance-of leaf-spores-frame))
(leaf-bugs-slot unknown
 :possible-values (:instance-of leaf-bugs-frame))
(defbehavior (leaf-symptoms-frame :put-specialisations)
  (new-value active-value prop slot)
  (cond-every
   ((multiple-value-string-equal new-value "Leaf spots")
    (definstance leaf-spots of leaf-spots-frame)
    (<-self :put 'leaf-spots-slot leaf-spots))
   ((multiple-value-string-equal new-value "Leaf mycelium")
    (definstance leaf-mycelium of leaf-mycelium-frame)
    (<-self :put 'leaf-mycelium-slot leaf-mycelium))
   ((multiple-value-string-equal new-value "Leaf spores")
    (definstance leaf-spores of leaf-spores-frame)
    (<-self :put 'leaf-spores-slot leaf-spores))
   ((multiple-value-string-equal new-value "Bugs")
    (definstance leaf-bugs of leaf-bugs-frame)
    (<-self :put 'leaf-bugs-slot leaf-bugs))))
)

(defframe leaf-spots-frame
  (supers disease-list-handler)
  (slots (color -
    :possible-values (:some-of "Yellow" "Ochre" "Brown" "Grey" "Black" unknown)
    :ask ("-% What is the color of the leaf spots? ")
    :text ("-% 20TObservation of 20T spots on the leaves: ")
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
  (spores -
    :possible-values (:one-of "Present" "Absent" unknown)
    :ask ("-% Are spores present in the leaf spots ? ")
    :after-asking ((if (= "present")(#A(leaf-spots spores-color)))))
    :text ("-% 20TObservation of spores on the leaves: ")
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
  (spores-color -
    :possible-values (:one-of "Brown" "Black" unknown)
    :ask ("-% What is the color of the leaf spots? ")
    :text ("-% 20TObservation of 20T spots on the leaves: ")
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
)

--- 171 ---
We have to use the name of the instance in the :AFTER-ASKING
Use of SELF in the after-asking gives error message.
Unknown value still triggers the :AFTER-ASKING question.
The aging constraint between GREY and other colors is not represented, as is
the topological constraint between the size and form of the spots on upper
and lower leaves. There are also constraints between the size of the spots
and the growth-stage of the plant.

********************
Leaf mycelium observation
********************

(defframe leaf-mycelium-frame
  (supers disease-list-handler)
  (slots (shape -
    :possible-values (:some-of
                       "Hyphae"
                       "Stripes"
                       "Patches"
                       unknown)
    :ask ("%-What is the shape of the mycelium on the leaves? ")
    :text "%-Observation of "A on the leaves: 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
  (color -
    :possible-values (:some-of
                       "White"
                       "Grey"
                       "Reddish-brown"
                       "Yellow-brown"
                       "Brown"
                       "Black"
                       "Orange"
                       unknown)
    :ask ("%-What is the color of the mycelium on the leaves? ")
    :text "%-Observation of "A mycelium on the leaves: 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
  (spores -
    :possible-values (:one-of "Present" "Absent" unknown)
    :ask ("%-Are spores present on the leaves? ")
    :after-asking ((if (= "present")(ask
                        (leaf-mycelium spores-color))))

- 172 -
Observation of spores on the leaves:
(supported-diseases #A(nil nil :add-disease-to-slot-list))

(spores-color -
:possible-values (:one-of
"Brown"
"Black"
unknown)
:ask ("%What color? ")
:text "-% "20TObservation of "A spores on the leaves:
(supported-diseases #A(nil nil :add-disease-to-slot-list))

(text "-% "20TMycelium on the leaves:")

Leaf bugs observation

(defframe leaf-bugs-frame
(supers disease-list-handler)
(slots (color -
:possible-values (:one-of
"Green"
"Dark green"
"Light green"
"Brown"
unknown)
:ask ("%What is the color of the bugs on the leaves?")
:text "-% "20TObservation of "A bugs on the leaves:
(supported-diseases #A(nil nil :add-disease-to-slot-list))

(shape -
:possible-values (:one-of
"Squat"
"Oblong"
unknown)
:ask ("%What is the shape of the bugs on the leaves?")
:text "-% "20T"A shape of the bugs:
(supported-diseases #A(nil nil :add-disease-to-slot-list))

(Siphons -
:possible-values (:one-of
"White"
"Green"
"Light-green"
"Dark-green"
"Black"
unknown)
:ask ("What is the color of the syphons (antennae)
of the bugs on the leaves?\)
:text "A siphons of the bugs: 
:supported-diseases #A(nil nil :add-disease-to-slot-list))
(supported-diseases #A(nil nil :add-disease-to-instance-list))
(text "Bugs on the leaves:\))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Leaf spores observation
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defframe leaf-spores-frame
 (supers disease-list-handler)
 (slots (color -
    :possible-values (:some-of
        "Yellow"
        "Pink"
        "Yellow-brown"
        "Brown"
        "Dark-brown"
        "Orange"
        "Orange-brown"
        "Purple"
        "Black"
        unknown)
    :ask ("What color do the spores on the leaves have?\)
    :text "Observation of the spores on the leaves: " 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
(support -
    :possible-values (:one-of
        "Stripes"
        "Patches"
        unknown)
    :ask ("How do the spores occur on the leaves?\)
    :text "The spores occur as: " 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
(epidermis -
    :possible-values (:one-of yes no unknown)
    :ask ("Is the epidermis broken around the spores?\)
    :text "Observation of a broken epidermis: " 
    :supported-diseases #A(nil nil :add-disease-to-slot-list))
(supported-diseases #A(nil nil :add-disease-to-instance-list))
(text "Spores on the leaves:\))

- 174 -
(deffrule-set :dialogue

;; The dialogue mirrors the specialisation hierarchy for symptoms and
;; cooperates with the :PUT-SPECIALISATIONS active values to build up the
;; SYMPTOMS-SLOT aggregate of the DISEASE-OCURRENCE instance.

(symptoms ($true)
  ($ask (plant-symptoms specialisations)))

(root-symptoms-1 ($and (not(plant-symptoms root-symptoms-slot = unknown)))
  ($ask (root-symptoms mycelium)
    (root-symptoms break-off)
  ))

(root-symptoms-2 ($and (not(root-symptoms root-mycelium-slot = unknown)))
  ($ask (root-mycelium color)
    (root-mycelium shape)
    (root-mycelium length))

(foot-symptoms-1 ($and (not(plant-symptoms foot-symptoms-slot = unknown)))
  ($ask (foot-symptoms specialisations)))

(foot-symptoms-2 ($and (not(foot-symptoms foot-stripes-slot = unknown)))
  ($ask (foot-stripes color)))

(foot-symptoms-3 ($and (not(foot-symptoms foot-discolorisation-slot = unknown)))
  ($ask (foot-discolorisation color)))

(foot-symptoms-4 ($and (not(foot-symptoms foot-spots-slot = unknown)))
  ($ask (foot-spots well-defined-borders)
    (foot-spots mycelium)))

(leaf-symptoms-1 ($and (not(plant-symptoms leaf-symptoms-slot = unknown)))
  ($ask (leaf-symptoms specialisations)))

(leaf-symptoms-2 ($and (not(leaf-symptoms leaf-spots-slot = unknown)))
  ($ask (leaf-spots color)
    (leaf-spots spores))

(leaf-symptoms-3 ($and (not(leaf-symptoms leaf-mycelium-slot = unknown)))
  ($ask (leaf-mycelium color)
    (leaf-mycelium shape)
    (leaf-mycelium length))

)
($ask (leaf-mycelium shape)
    (leaf-mycelium color)
    (leaf-mycelium spores)))

(leaf-symptoms-4 ($and (not(leaf-symptoms leaf-bugs-slot = unknown)))
    ($ask (leaf-bugs color)
        (leaf-bugs shape)
        (leaf-bugs siphons)))

(leaf-symptoms-5 ($and (not(leaf-symptoms leaf-spores-slot = unknown)))
    ($ask (leaf-spores color)
        (leaf-spores shape)
        (leaf-spores epidermis)))

;;;;; **************************************************************************
;;;;; decision graphs for diagnosis rules
;;;;; **************************************************************************
;;;;; The nodes are implemented either as actions or as tests, not as both.
;;;;; There is no difference between nodes 4 and 5, which should implement
;;;;; deterministic and possibilistic assignment, respectively.

(defun septoria-tritici ()
  (tagbody
   0 (if (is-instance leaf-spots)
       ;; then
       (progn
        (<- symptoms-occurrence :put 'symptoms-list leaf-spots)
        (<- leaf-spots :put 'supported-diseases septoria-tritici)
        (go 1))
       ;; else
       (go fail))
   1 (if (member (<- leaf-spots :get 'color)
        
      (list "Yellow" "Ochre" "Brown" "Grey"))
       ;; then
       (progn
        (<- leaf-spots :put 'color septoria-tritici :supported-diseases )
        (go 2)))
       ;; else
       (if (eq (<- leaf-spots :get 'color) 'unknown)
           ;; then
           (go 2)
           ;; else
           (go fail))
    2 (if (equal (<- leaf-spots :get 'spores) "Present")
       ;; then
       (progn

- 176 -
(<- leaf-spots :put 'spores septoria-tritici :supported-diseases )
(go 3)))
;; else
(if (eq (<- leaf-spots :get 'spores) 'unknown)
  ;; then
  (go 4)
  ;; else
  (go fail))
3 (if (equal (<- leaf-spots :get 'spores-color) "Black")
  ;; then
  (progn
    (<- leaf-spots :put 'spores-color septoria-tritici :supported-diseases )
    (go 4)))
(if (eq (<- leaf-spots :get 'spores-color) 'unknown)
  ;; then
  (go 4)
  ;; else
  (go fail))
4 (<- disease-occurrence :put 'possible-disease-causes septoria-tritici)
fail))

;;; ******************************************************************************
(defun septoria-nodorum ()
  (tagbody
    0 (if (is-instance leaf-spots)
     ;; then
     (progn
       (<- symptoms-occurrence :put 'symptoms-list leaf-spots)
       (<- leaf-spots :put 'supported-diseases septoria-nodorum)
       (go 1))
     ;; else
     (go fail))
1 (if (member (<- leaf-spots :get 'color)
              (list "Yellow" "Ochre" "Brown" "Grey"))
     ;; then
     (progn
       (<- leaf-spots :put 'color septoria-nodorum :supported-diseases)
       (go 2)))
     ;; else
     (if (eq (<- leaf-spots :get 'color) 'unknown)
      ;; then
      (go 2)
      ;; else
      (go fail))
2 (if (equal (<- leaf-spots :get 'spores) "Present")

- 177 -
;;; then
(progn
  (<- leaf-spots :put 'spores septoria-nodorum :supported-diseases)
  (go 3))
;; else
(if (equal (<- leaf-spots :get 'spores) "Absent")
  ;; then
  (go 4)
  ;; else (unknown)
  (go 4))
3 (if (equal (<- leaf-spots :get 'spores-color) "Brown")
  ;; then
  (progn
    (<- leaf-spots :put 'spores-color septoria-nodorum :supported-diseases)
    (go 4))
  ;; else
  (if (eq (<- leaf-spots :get 'spores-color) 'unknown)
    ;; then
    (go 4)
    ;; else
    (go fail))
4 (<- disease-occurrence :put 'possible-disease-causes septoria-nodorum)
fail))

;;;  ****************************************************************************

(defun gaumannomyces-graminis ()
  (tagbody
    0
    (if (is-instance root-mycelium)
      ;; then
      (progn
        (<- symptoms-occurrence :put 'symptoms-list root-mycelium)
        (<- root-mycelium :put 'supported-diseases gaumannomyces-graminis)
        (go 1))
      ;; else
      (go fail))
    1 (if (member (<- root-mycelium :get 'color)
                  (list "brown" "Black"))
        ;; then
        (progn
          (<- root-mycelium :put 'color gaumannomyces-graminis :supported-diseases)
          (go 2))
        ;; else
        (if (eq (<- root-mycelium :get 'color) 'unknown)
            ;; then
            )
(go 2)
;; else
(go fail))

2 (if (equal (<- root-mycelium :get 'shape) "hyphae")
  ;; then
  (progn
    (<- root-mycelium :put 'shape gaumannomyces-graminis :supported-diseases)
    (go 3))
  ;; else
  (if (eq (<- root-mycelium :get 'shape) 'unknown)
    ;; then
    (go 3)
    ;; else
    (go fail))
  (go fail))

3 (<- disease-occurrence :put 'possible-disease-causes gaumannomyces-graminis)

;;;****************************************************************************
(defun erysiphe-graminis ()
  (tagbody
    0 (if (is-instance leaf-mycelium)
      ;; then
      (progn
        (<- symptoms-occurrence :put 'symptoms-list leaf-mycelium)
        (<- leaf-mycelium :put 'supported-diseases erysiphe-graminis)
        (go 1))
      ;; else
      (go fail))
    1 (if (equal (<- leaf-mycelium :get 'shape) "Patches")
      ;; then
      (progn
        (<- leaf-mycelium :put 'shape erysiphe-graminis :supported-diseases)
        (go 2))
      (if (eq (<- leaf-mycelium :get 'shape) 'unknown)
        ;; then
        (go 2)
        ;; else
        (go fail))
    2 (if (equal (<- leaf-mycelium :get 'color) "White")
      ;; then
      (progn
        (<- leaf-mycelium :put 'color erysiphe-graminis :supported-diseases)
        (go 4))
      ;; else
      (go 3))
  )
)
3 (if (member (<- leaf-mycelium :get 'color)
    (list "Brown" "Black"))
    ;; then
    (progn
        (<- leaf-mycelium :put 'color erysiphe-graminis :supported-diseases)
        (go 4))
    (if (eq (<- leaf-mycelium :get 'color) 'unknown)
        ;; then
        (go 4)
        ;; else
        (go fail))
4 (if (equal (<- leaf-mycelium :get 'spores) "Present")
    ;; then
    (progn
        (<- leaf-mycelium :put 'spores erysiphe-graminis :supported-diseases)
        (go 5))
    ;; else
    (go 6))
5 (if (member (<- leaf-mycelium :get 'spores-color)
    (list "Brown" "Black"))
    ;; then
    (progn
        (<- leaf-mycelium :put 'spores-color erysiphe-graminis :supported-diseases)
        (go 6))
    (if (eq (<- leaf-mycelium :get 'spores-color) 'unknown)
        ;; then
        (go 6)
        ;; else
        (go fail))
6 (<- disease-occurrence :put 'possible-disease-causes erysiphe-graminis)
fail))
;;; ******************************************************************************
(defun fusarium-culmorum ()
  (tagbody
    0 (if (is-instance foot-stripes)
        ;; then
        (progn
            (<- symptoms-occurrence :put 'symptoms-list foot-stripes)
            (<- foot-stripes :put 'supported-diseases fusarium-culmorum)
            (go 2))
        ;; else
        (go 1))
    1 (if (is-instance foot-discolorisation)
        ;; then
        (go fail))
    fail))
(progn
  (<- symptoms-occurrence :put 'symptoms-list foot-discolorisation)
  (<- foot-discolorisation :put 'supported-diseases fusarium-culmorum)
  (go 3))
  ;; then
  (progn
    (<- foot-stripcolors :put 'color fusarium-culmorum :supported-diseases)
    (go 5)))
  (if (eq (<- foot-stripcolors :get 'color) 'unknown)
    ;; then
    (go 5)
    ;; else
    (go fail))
  3 (if (is-instance 'root-mycelium)
    ;; then
    (go fail)
    ;; else
    (go 4))
  4 (if (is-instance leaf-bugs)
    ;; then
    (progn
      (<- symptoms-occurrence :put 'symptoms-list leaf-bugs)
      (<- leaf-bugs :put 'supported-diseases sitobion-avenae)
      (go 1))
    ;; else
    (go fail))
  5 (<- disease-occurrence :put 'possible-disease-causes fusarium-culmorum)
  fail)

;;; ******************************************************************************

(defun sitobion-avenae ()
  (tagbody
    0 (if (is-instance leaf-bugs)
      ;; then
      (progn
        (<- symptoms-occurrence :put 'symptoms-list leaf-bugs)
        (<- leaf-bugs :put 'supported-diseases sitobion-avenae)
        (go 1))
      ;; else
      (go fail))

- 181 -
1 (if (member (<-leaf-bugs :get 'color) (list "Green" "Dark green" "Brown"))
 ;; then
 (progn
   (<-leaf-bugs :put 'color sitobion-avenae :supported-diseases)
   (go 2))
 (if (eq (<-leaf-bugs :get 'color) 'unknown)
 ;; then
   (go 2)
 ;; else
   (go fail))
 2 (if (equal (<-leaf-bugs :get 'shape) "Squat")
 ;; then
   (progn
     (<-leaf-bugs :put 'shape sitobion-avenae :supported-diseases)
     (go 3))
 (if (eq (<-leaf-bugs :get 'shape) 'unknown)
 ;; then
   (go 3)
 ;; else
   (go fail))
 3 (if (equal (<-leaf-bugs :get 'siphons) "Black")
 ;; then
   (progn
     (<-leaf-bugs :put 'siphons sitobion-avenae :supported-diseases)
     (go 4))
 (if (eq (<-leaf-bugs :get 'siphons) 'unknown)
 ;; then
   (go 4)
 ;; else
   (go fail))
 4 (<-disease-occurrence :put 'possible-disease-causes sitobion-avenae)
 fail)

;;; **************************************************************************

(defun metopolophium-dirhodum ()
  (tagbody
   0 (if (is-instance leaf-bugs)
 ;; then
   (progn
     (<-symptoms-occurrence :put 'symptoms-list leaf-bugs)
     (<-leaf-bugs :put 'supported-diseases metopolophium-dirhodum)
     (go 1))
 ;; else
   (go fail)))

- 182 -
1 (if (member (<- leaf-bugs :get 'color)  
    (list "Green" "light green"))  
    ;; then  
    (progn  
      (<- leaf-bugs :put 'color metopolophium-dirhodum :supported-diseases)  
      (go 2)))  
    (if (eq (<- leaf-bugs :get 'color) 'unknown)  
      ;; then  
      (go 2)  
      ;; else  
      (go fail))

2 (if (equal (<- leaf-bugs :get 'shape) "Oblong")  
    ;; then  
    (progn  
      (<- leaf-bugs :put 'shape metopolophium-dirhodum :supported-diseases)  
      (go 3)))  
    (if (eq (<- leaf-bugs :get 'shape) 'unknown)  
      ;; then  
      (go 3)  
      ;; else  
      (go fail))

3 (if (equal (<- leaf-bugs :get 'siphons) (list "White" "Green"))  
    ;; then  
    (progn  
      (<- leaf-bugs :put 'siphons metopolophium-dirhodum :supported-diseases)  
      (go 4)))  
    (if (eq (<- leaf-bugs :get 'siphons) 'unknown)  
      ;; then  
      (go 4)  
      ;; else  
      (go fail))

4 (<- disease-occurrence :put 'possible-disease-causes metopolophium-dirhodum)  
  fail)

;;; ******************************************************************************

(defun pseudocercosporella-herpotrichoides ()  
  (tagbody  
    0 (if (is-instance foot-spots)  
      ;; then  
      (progn  
        (<- symptoms-occurrence :put 'symptoms-list foot-spots)  
        (<- foot-spots :put 'supported-diseases  
          pseudocercosporella-herpotrichoides)

- 183 -
(go 1))
;; else
(go fail))
1 (if (eq (< foot-spots :get 'well-defined-borders) 'no)
    ;; then
    (progn
    (< foot-spots :put 'well-defined-borders
    pseudocercosporella-herpotrichoides :supported-diseases)
    (go 2)))
(if (eq (< foot-spots :get 'well-defined-borders) 'unknown)
    ;; then
    (go 2)
    ;; else
    (go fail))
2 (if (eq (< foot-spots :get 'mycelium) 'yes)
    ;; then
    (progn
    (< foot-spots :put 'mycelium
    pseudocercosporella-herpotrichoides :supported-diseases)
    (go 3))
    ;; else
    (go 4))
3 (if (equal (< foot-spots :get 'mycelium-color) "Black")
    ;; then
    (progn
    (< foot-spots :put 'mycelium-color
    pseudocercosporella-herpotrichoides :supported-diseases)
    (go 4)))
(if (eq (< foot-spots :get 'mycelium-color) 'unknown)
    ;; then
    (go 4)
    ;; else
    (go fail))
4 (<-disease-occurrence :put 'possible-disease-causes
pseudocercosporella-herpotrichoides)
fail))
;;; ****************************************************************************
(defun corticium-solani ()
(tagbody
  0 (if (is-instance foot-spots)
    ;; then
    (progn
    (<- symptoms-occurrence :put 'symptoms-list foot-spots)
    (<- foot-spots :put 'supported-diseases corticium-solani)
    (go 1))
- 184 -
(defun pucinia-striiformis ()
  (tagbody
   0 (if (is-instance leaf-spores)
       ;; then
       (progn
         (<- symptoms-occurrence :put 'symptoms-list leaf-spores)
         (<- leaf-spores :put 'supported-diseases pucinia-striiformis)
         (go 1))
       ;; else
       (go fail))
   )
)
(defun pucinia-striiformis ()
  (tagbody
   0 (if (is-instance leaf-spores)
       ;; then
       (progn
         (<- leaf-spores :put 'shape pucinia-striiformis :supported-diseases)
         (go 2))
       (if (eq (<- leaf-spores :get 'shape) 'unknown)
           ;; then
           (go 2)
           ;; else
           (go fail)))
   1 (if (member (<- leaf-spores :get 'color) (list "Yellow" "Yellow-brown" "Brown"))
       ;; then
       (progn
         (<- leaf-spores :put 'color pucinia-striiformis :supported-diseases)
         (go 3))
       (if (eq (<- leaf-spores :get 'color) 'unknown)
           ;; then
           (go 3)
           ;; else
           (go fail)))
   2 (if (equal (<- leaf-spores :get 'shape) "Stripes")
       ;; then
       (progn
         (<- leaf-spores :put 'shape pucinia-striiformis :supported-diseases)
         (go 2))
       (if (eq (<- leaf-spores :get 'shape) 'unknown)
           ;; then
           (go 2)
           ;; else
           (go fail)))
   3 (<- disease-occurrence :put 'possible-disease-causes pucinia-striiformis fail))

;;; ****************************************************************************

(defun pucinia-recondita ()
  (tagbody
   0 (if (is-instance leaf-spores)
       ;; then
       (progn
         (<- symptoms-occurrence :put 'symptoms-list leaf-spores)
         (<- leaf-spores :put 'supported-diseases pucinia-recondita)
         (go 1))
       ;; else
       (go fail)))
   1 (if (equal (<- leaf-spores :get 'shape) "Patches")
       ;; then
       (progn
         (<- leaf-spores :put 'shape pucinia-recondita :supported-diseases)
         (go 2))
       (if (eq (<- leaf-spores :get 'shape) 'unknown)
           ;; then
           (go 2)
           ;; else
           (go fail)))
   - 186 -
2 (if (eq (<- leaf-spores :get 'epidermis) 'yes)
 ;; then
 (progn
   (<- leaf-spores :put 'epidermis pucinia-recondita :supported-diseases)
   (go 3))
 (if (eq (-< leaf-spores :get 'epidermis) 'unknown)
 ;; then
 (go 3)
 ;; else
 (go fail))
)
3. (if (member (-< leaf-spores :get 'color)
 (list "Orange" "Orange-brown" "Brown" "Dark-brown")
 ;; then
 (progn
   (<- leaf-spores :put 'color pucinia-recondita :supported-diseases)
   (go 4))
 (if (eq (-< leaf-spores :get 'color) 'unknown)
 ;; then
 (go 4)
 ;; else
 (go fail))
fail))

;;; ........................................................................................................
;;; instruction part
;;; ........................................................................................................

(instructions
 (setq leaf-spots nil)
 (setq leaf-spores nil)
 (setq foot-spots nil)
 (setq root-mycelium nil)
 (setq leaf-mycelium nil)
 (setq foot-stripes nil)
 (setq foot-dicolorisation nil)
 (setq foot-symptoms nil)
 (setq leaf-bugs nil)
 (definstance symptoms-occurrence of symptoms-occurrence-frame)
 (<- disease-occurrence :instantiate)
 (defvar f1)
 (defvar f2)
 (setq f2 (open "hippo:>kb3>uit.dat" ':direction ':output))
 (setq f1 (make-broadcast-stream standard-output f2))
 (<- disease-occurrence :report)
 (<- symptoms-occurrence :report)

- 187 -