VRIJE UNIVERSITEIT VAN AMSTERDAM

KNOWLEDGE ENGINEERING

Botanical Collection Management

Marc Went, 2507013, mwt680

Gary Mikoen, 2508918, gmn270

January 29, 2015

Contents

Chapter 1

Introduction

Starting a botanical collection is not as easy as it sounds. Many factors weigh in for collecting and maintaining your plants and flowers. For instance, you want to know if your plants are compatible with each other and if they can live under the provided circumstances. For instance, a collection containing only cactus plants will not be very suitable for pines. A collector working for a Hortus, a museum for plants, will therefore have a hard time in finding new botanical additions. He always needs to keep the habitation of his collection in mind and also the collection itself. This can be a very knowledge intensive task.

A system that automatically gives suggestions for additional plants would be of great help for the collector. Especially if the suggestions are based on your current collection. Therefore, this project aims to develop such a system. Interviews with experts were held and worksheets and models were made to realize a botanical collection system. The eventual outcome can be found in the following link:

`https://mwent.info/ke/`

Part I

Worksheets

Chapter 2

Worksheets

2.1 OM 1

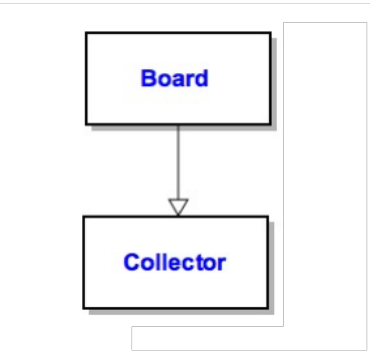
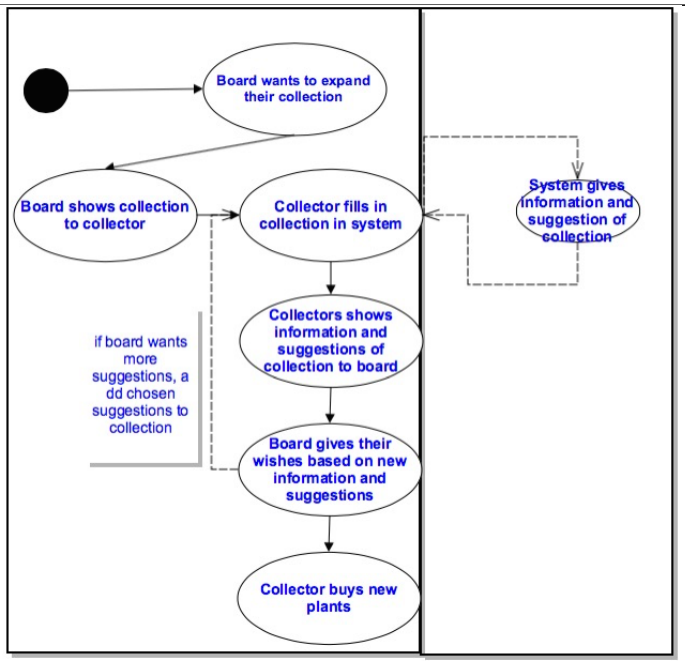
Identifying knowledge-oriented problems and opportunities in the organization

Organization Model	Problems and Opportunities Worksheet OM-1
PROBLEMS AND OPPORTUNITIES	**Problems**: 1. Collectors/users need a lot of time to search and find plants to expand their botanical collection. 2. Collectors/users don't have all the knowledge to expand their collection accordingly. Life on earth is dynamic so the flora is constantly changing. It takes an immense amount of time to keep up with all the new information. 3. Knowing the optimal condition for your botanical collection takes time. Imagine all of the encyclopedia that exists revolving botanics. **Opportunities**: 1. Additional flora for your botanical collection can be found quickly and automatically by using external known databases. 2. Specific queries can be used to find flora specifically fitted to your collection. 3. External online databases can give information about the optimal conditions of your current collection immediately and on the spot.
ORGANIZATIONAL CONTEXT	**Mission**: Finding and maintaining your collection of plants. This is a very knowledge intensive task so a system containing botanical knowledge should lower the time and work for finding and maintaining plants in your collection. **Vision**:

	A well used knowledge based system that helps collectors and plant enthusiasts in finding the right plant and knowing how to maintain them. This system should lower work for professional collectors in places like a hortus or should help plant enthusiasts. **Goals**: 1. Lower the time needed in finding botanical plants 2. Find additional plants that collector has not thought of 3. maintain collection of plants
	Important external factors the organization has to deal with: 1. Relies on external databases 2. Should be online accessible 3. Dynamic earth; life on earth is constantly changing. Factors like global warming or the weather on earth influences life on earth heavily.
	Strategy of the organization: The organization will extract information from external databases (known by botanists). This information contains features on which botanical plants can be classified too. The use of experts will determine the queries on which additional plants will be found by the knowledge based system.
	Its value chain and the major value drivers: Immediately give information of plants and suggestions for other plants. This will save a tremendous amount of time in researching and giving information about (compatible) plants. **Major value driver**: Decrease the valuable time a collector spends on finding additional plants and information for his collection. This gives the opportunity for collectors to spend more time in fully optimizing the collection and use his time more efficiently.
SOLUTIONS	1. Build a knowledge based system that automatically gives suggestions of additional plants based on your current collection of plants. 2. The suggestions are based on information extracted from external databases. This information should give the optimal condition for your plants. The knowledge based system should also list this extracted information so collectors can correctly maintain their collection.

2.2 OM 2

Description of organizational aspects that have an impact on and/or are affected by chosen knowledge solutions

Organization Model	Variant Aspects Worksheet OM-2
STRUCTURE	
PROCESS	
PEOPLE	**Board:** The board wants their collection of plants to be updated. This can be the board of a hortus but also the board of a zoo or safari park. **Collectors:** The collectors are people working at a Hortus, an employee. These people maintain and expand the collection of a Hortus

RESOURCES	1. **Collection of plants** The collection of plants is needed for the knowledge based system to give suggestions of new plants. 2. **Knowledge Based System** The knowledge based system is the system that links to a database. With active queries within the system, the system gives automatic and quick suggestions on how to expand your collection 3. **External Database** External database with information of botanical and esthetic's features of plants. The system needs this database to look up information and give suggestions.
KNOWLEDGE	**Features:** Knowledge about the abiotic and esthetic's features of plants is necessary to classify which plants are compatible with the current collection. **Classification:** Knowledge on how to classify plants, knowing which queries to use for the knowledge based system.
CULTURE & POWER	**Culture:** The highest placed member in the hierarchy is the board member that decides if a collection needs to be expanded. The collector uses the system to comply to the board member. **Power:** The collector knows the culture of the company by checking out the collection of the Hortus and the wishes of the board.

2.3 OM 4

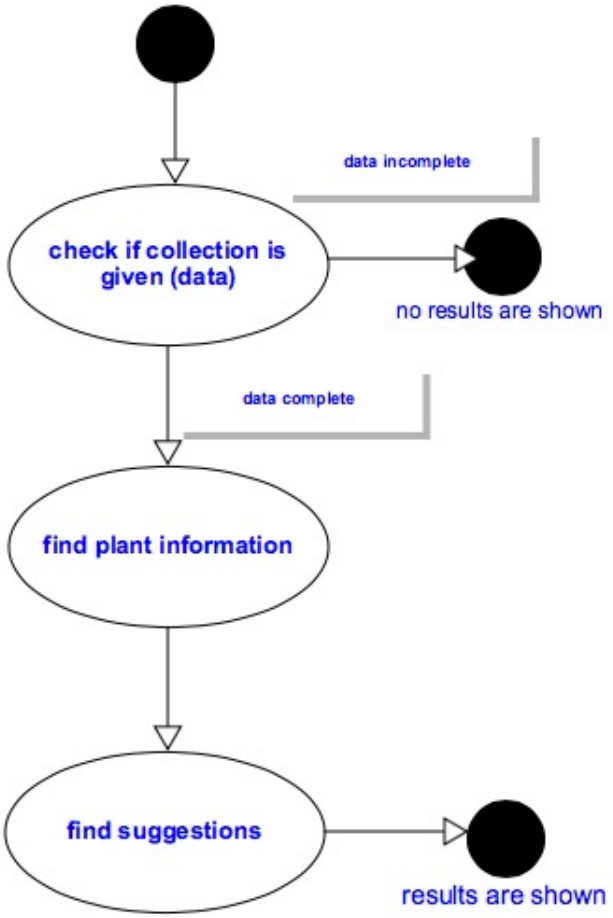
Description of the Knowledge component of the organization model and its major characteristics

Organization Model		**Knowledge Assets Worksheet OM-4**				
KNOWLEDGE ASSET	POSSESSED BY	USED IN	RIGHT FORM?	RIGHT PLACE?	RIGHT TIME?	RIGHT QUALITY?
Classification suggestion of plants	Collector	Classification	Yes	Yes	Yes	Yes
Information of plants	Collector	Monitoring	Yes	Yes	Yes	Yes

2.4 TM-1

Refined description of the tasks within the target process

Task Model	**Task Analysis Worksheet TM-1**
TASK	Collection expander
ORGANIZATION	Collector
GOAL AND VALUE	Classifying which plants are compatible with the current collection
DEPENDENCY AND FLOW	**Preceding Tasks**: Put current collection in database **Follow-up Tasks**: Add chosen suggestions to current collection
OBJECTS HANDLED	**Input objects**: Abiotic Features 1. Soil 2. pH 3. Hardiness 4. Aspect Aesthetic Features 1. Color 2. Area **Output objects**: Information of collection 1. Abiotic Features Suggestion of additional plants **Internal objects** Queries on which features needs to extracted

TIMING AND CONTROL	**Follow-up Tasks**: The system can be used any time the collection needs to be expanded and takes little time **Control**:  **Constraints & Conditions**: (i) **pré-conditions** collection must be given (ii) **post-conditions** find information and suggestions of plants task; *constraints* queries needs to be well executed
AGENTS	Collector
KNOWLEDGE AND COMPETENCE	Knowledge of collector and board to successfully decide if suggestions are good enough
RESOURCES	• Current botanical collection • External databases that the knowledge based system uses
QUALITY AND PERFORMANCE	Board decides if suggestions are right and fitting for their collection.

2.5 TM-2

Specification of the knowledge employed for a task, and possible bottlenecks and areas for improvement

Task Model	**Knowledge Item Worksheet TM-2**
NAME	Collection Expansion
POSSESSED BY	Collector
USED IN	Classification Task
DOMAIN	Botanica, Flora maintenance
	discipline, branch of science or engineering, professional community)

Nature of the knowledge		**Bottleneck / to be improved?**
Formal, rigorous	x	
Empirical, quantitative	x	
Heuristic, rules of thumb		
Highly specialized, domain-specific	x	
Experience-based		
Action-based	x	
Incomplete	x	x
Uncertain, may be incorrect		
Quickly changing		
Hard to verify		
Tacit, hard to transfer		

Form of the knowledge		
Mind		
Paper		
Electronic	x	
Action skill	x	x (Collection needs to be manually filled in)
Other		

Availability of knowledge		
Limitations in time		
Limitations in space		
Limitations in access		
Limitations in quality	x	x
Limitations in form	x	x

2.6 AM-1

Agent specification according to the CommonKADS agent model

Agent Model	Agent Worksheet AM-1
NAME	*Collector*
ORGANIZATION	Collector takes order from board members
INVOLVED IN	Expanding botanical collection
COMMUNICATES WITH	Board
KNOWLEDGE	Botanical knowledge
OTHER COMPETENCES	Knowledge if classification is done right
RESPONSIBILITIES AND CONSTRAINTS	**Responsibilities:** Know what plants there are in the collection. Take care of and maintain the collection **Constraints** Collector takes order from board to add or buy plants.

Agent Model	Agent Worksheet AM-1
NAME	*Board*
ORGANIZATION	Highest in the hierarchy of the Hortus
INVOLVED IN	Expanding botanical collection
COMMUNICATES WITH	Collector
KNOWLEDGE	Organization knowledge of the company
OTHER COMPETENCES	Leadership Decisiveness
RESPONSIBILITIES AND CONSTRAINTS	**Responsibilities:** Leading the organization. Makes sure everyone at the company does his job or has something to do. **Constraints** Dependent on the availability of his employees.

Part II

Models

Chapter 3

Knowledge model

3.1 Task Knowledge

3.1.1 Task template

Nature of output: Suggestion of additional classified plants and information of their abiotic features. Nature of input: Common names of plants in your current collection.

While developing the knowledge based system, it seemed that the botanical collection system mostly resembled the classification task template. However, to fully fit the botanical collection system, the classification task template has been modified to fully meet the needs of this project. The original unaltered classification task template shows a candidate that needs to be classified. This doesn't line up with the botanical collection system. In this system, the candidate is a class where other plants (suggestions) are be classified to. The botanical collection system wants to find additional plants based on your current collection of flora. In other words, the candidate or current collection of botanical plants is the class where additional plants found in your database are classified to. So, the suggestions are more or less candidates. A more profound explanation can be found in the inference model where the process is completely seen.

3.1.2 Task decomposition diagram

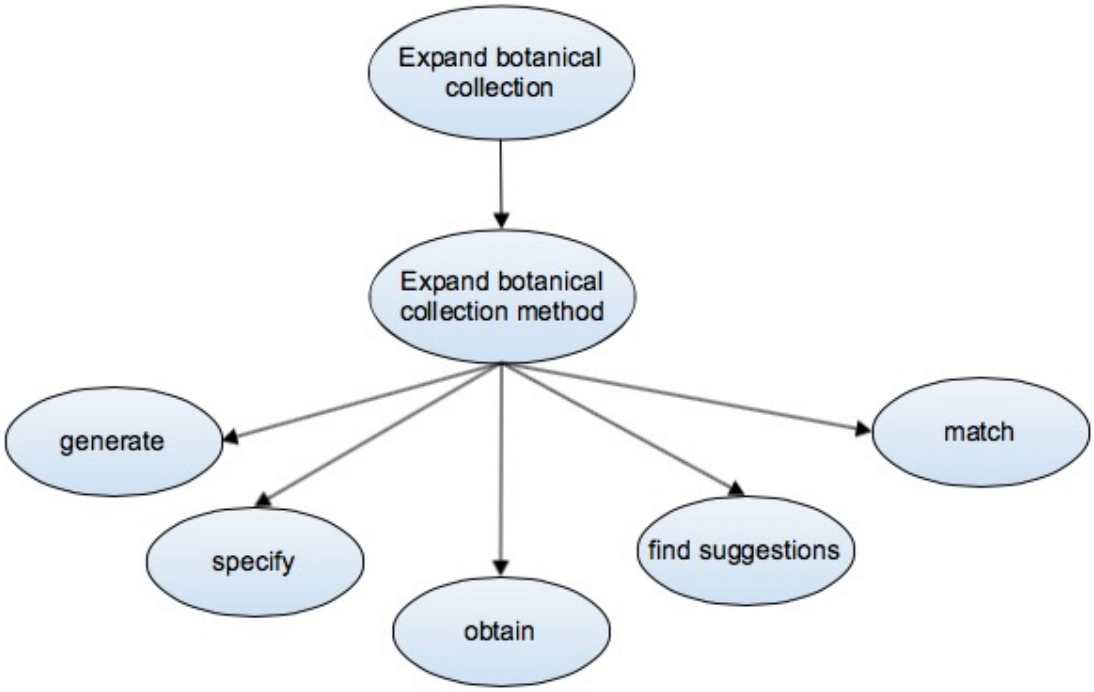


Figure 3.1: Tasks decomposed

3.1.3 Task and Task method description

```
TASK find-relevant-plant
    GOAL:
        "Formulate weighted plants based on plant factors";
    ROLES:
        INPUT:
            factors: "A set of (a)biotic factors for a specific plant";
        OUTPUT:
            plants: "The 10 most relevant plants based on the input factors";
    SPECIFICATION: "Find what plants could go well with the input plant,
                    based on given factors";
END TASK find-relevant-plant

TASK-METHOD find-relevant-plant-method
    REALIZED: find-relevant-plant;
    DECOMPOSITION:
        INFERENCES: select, specify, verify, filter;
        TANSFER-FUNCTION: obtain;
    ROLES:
        INTERMEDIATE:
            hypotheses: "List of a-biotic factors"
            hypothesis: "A a-biotic factor";
            observable: "A plant in the plant database";
```

```
            finding: "A value to what extent the plant is similar to the
                      desired plant";
            result: "List of similar plants";
            filtered_result: "List of top 10 weighted and filtered results";
    CONTROL-STRUCTURE:
        FORALL hypotheses as hypothesis:
            select(hypothesis -> observable);
            specify(observable -> finding);
            verify(finding -> result);
            filter(result -> filtered_result);
        END-FORALL
END TASK-METHOD;

TASK compare-plant-factors:
    GOAL:
        "To find the best factors and compare them to the current factors
         of a plant";
    ROLES:
        INPUT:
            user-specific-factors: "The factors a plant has with the
                                    current user";
            best-plant-factors: "Factors plant grow best in, given
                                 through a database";
        OUTPUT:
            comparison: "Comparison between user-specific-factors and
                         best-plant-factors";
    SPECIFICATION: "The comparison between the user his plant and the
                    optimal factors of that plant";
END-TASK compare-plant-factors;

TASK-METHOD compare-plant-factors-method:
    REALIZED: compare-plant-factors;
    DECOMPOSITION:
        INFERENCES: match;
        TRANSFER-FUNCTION: null;
    CONTROL-STRUCTURE:
        match(user-specific-factors -> best-plant-factors);
END-TASK-METHOD;
```

3.2 Inference knowledge

As said in the section "Task knowledge", the classification task template has been altered to fit the botanical collection system. In the modified classification task template, two chains have been added to the process. These two links are "find suggestions" and "suggestion". Without these two links, only the abiotic features are matched and compared to the candidate. In essence, the suggestions are based on the abiotic features but the suggestions are not equal to the abiotic features.

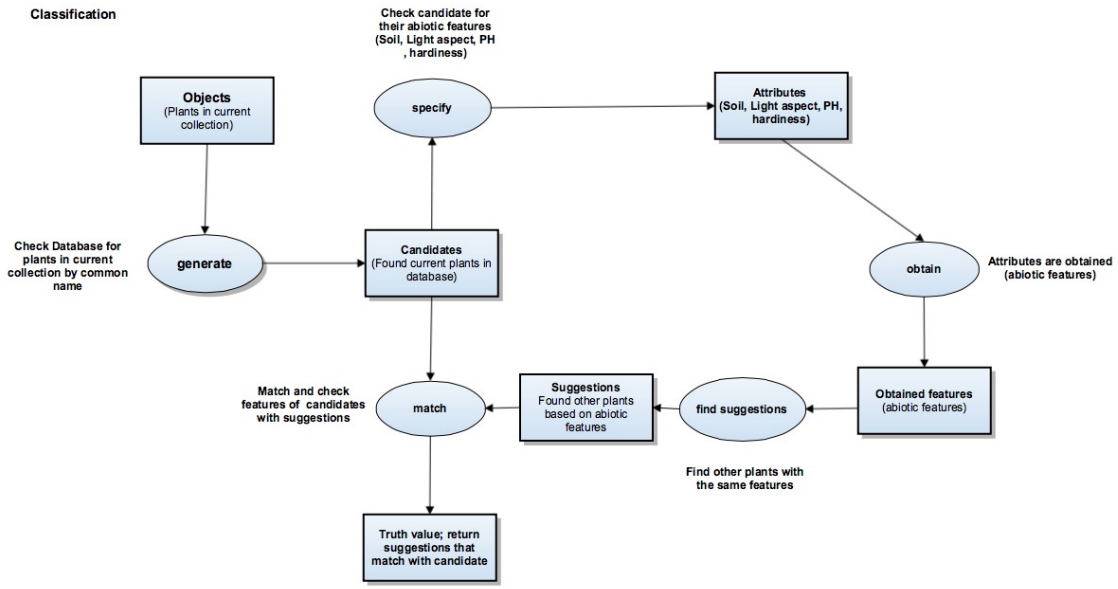


Figure 3.2: Classification Task Model

Step	Action	Description
1	Object	The current botanical collection of plants
2	Generate	Check database if the current collection can be found
3	Candidate	Found plants are candidates
4	Specify	Check candidates for their attributes
5	Attributes	Attributes are found as abiotic features
6	Obtain	Obtain the abiotic features of the current collection
7	Obtained features	Features of current collection are found
8	Find suggestion	Find other plants based on the features
9	Suggestions	Additional plants are found
10	Match	Match suggestions with the candidate
11	Truth value	Return the suggestions that match + the abiotic features

3.2.1 Inference

Inference	Input	Output	Specification
GENERATE	Object	Candidate	Candidates are selected here
SPECIFY	Candidate	Attributes	Attributes are selected here
OBTAIN	Attributes	Obtained features	Features are obtained here
FIND SUGGESTION	Obtained features	Suggestions	Suggestions are made here
MATCH	Suggestions	Truth value	Suggestions + information are given here

3.2.2 Knowledge role

Knowledge role	Type	Domain mapping
OBJECT	Static	Plants in current collection
CANDIDATE	Dynamic	Found current plants in Database
ATTRIBUTES	Dynamic	Soil, light aspect, pH, hardiness
OBTAINED FEATURES	Dynamic	Abiotic features
SUGGESTIONS	Dynamic	Found plants
TRUTH VALUE	Dynamic	Suggestion + information

3.3 Inference domain

3.3.1 Domain schema

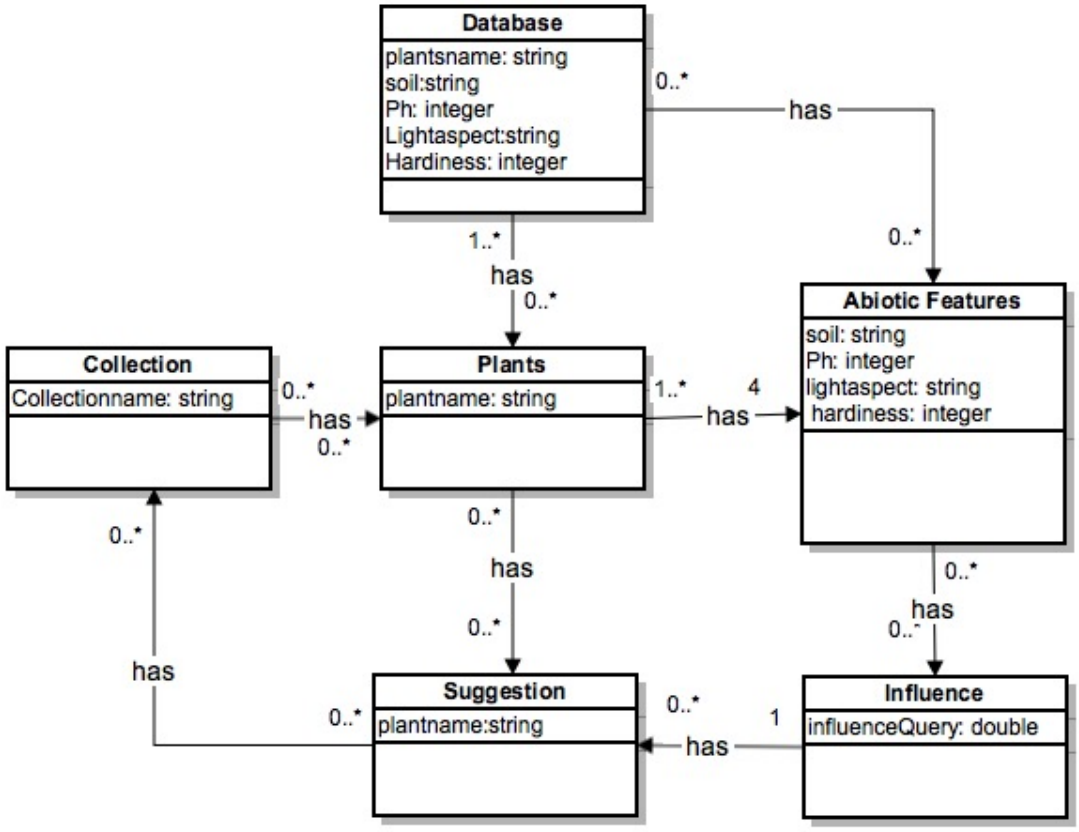


Figure 3.3: domain model

3.3.2 Rule types

```
RULE-TYPE Factor-influence-rule;
    DESCRIPTION: "Rule stating the relation between the plants and a factor";
        ANTECEDANT: Factor;
            CARDINALITY: 1+;
```

```
        CONSEQUENT: Plant;
            CARDINALITY: 1+;
        CONNECTION-SYMBOL: causes;
END RULE-TYPE;
```

3.3.3 Knowledge base

Because in logic $(P \rightarrow Q, P \vdash Q)$ but $(P \rightarrow Q, Q \nvdash P)$. Since nothing is said about what happens when P is false; Q could be either true or false The same goes for our knowledge base. When a pH.value = alkaline, it means that any suggested plant needs at least the pH.value = alkaline, but it doesn't say whether neutral or acid should be available or not.

The example here show we cannot explain it using the causes rule-set:

```
pH.value = alkaline
causes
show acid = false

pH.value = acid
causes
show alkaline = false

pH.value  = neutral
causes
show acid = true
show alkaline = true

aspect.value = north-facing
causes
show south-facing = false
```

3.4 Scenario's

3.4.1 Finding suggestions

The knowledge based system can handle two different known scenarios. One of those scenarios is expanding your botanical collection by submitting your current collection in the knowledge based system. The result of submitting of your current collection is a suggestion of new plants that match the abiotic features of your current collection.

3.4.2 Showing optimal conditions for your current collection

The second scenario works essentially the same but the output is different. Like the first scenario, the current collection of plants is submitted into the knowledge based system. The knowledge based system responds by showing the abiotic features of your current collection.

It can be noticed that the proceedings of both scenario 1 and 2 are the same. This means that the output for both scenarios is the same. Both scenario 1 and 2 show a list of suggestions with the abiotic features of the current collection.

Chapter 4

Communication model

4.1 Communication plan

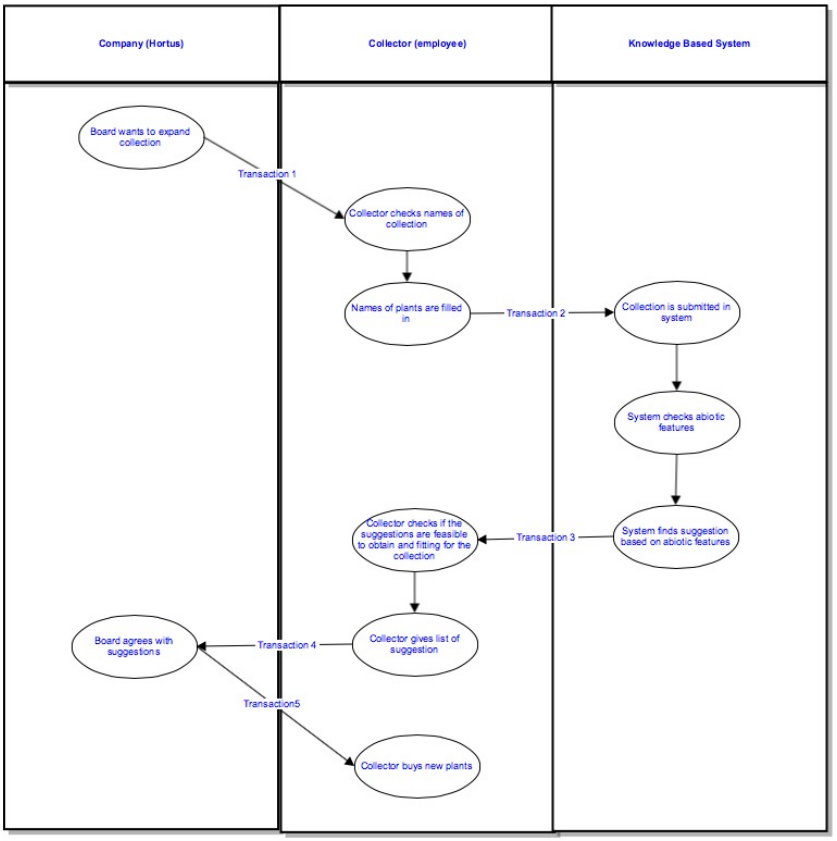


Figure 4.1: Classification Task

4.2 Transactions

4.2.1 Transaction 1

Communication model	Transaction description worksheet CM-1
TRANSACTION IDENTIFIER/NAME	Request collection expansion
INFORMATION OBJECT	Transaction between the initial need (expand collection) and the starting point for a collector (check current collection)
AGENTS INVOLVED	Sender: Board Receiver: Collector
COMMUNICATION PLAN	Figure 4.1 communication plan
CONSTRAINT	:- (collector must be available)

4.2.2 Transaction 2

Communication model	Transaction description worksheet CM-1
TRANSACTION IDENTIFIER/NAME	Submit current collection
INFORMATION OBJECT	Transaction between the filling in of the common names and submitting the current collection in the system
AGENTS INVOLVED	Sender: Collector Receiver: System
COMMUNICATION PLAN	Figure 4.1 communication plan
CONSTRAINT	:- (Collection must be complete and accurately named)

4.2.3 Transaction 3

Communication model	Transaction description worksheet CM-1
TRANSACTION IDENTIFIER/NAME	Return suggestions
INFORMATION OBJECT	Transaction between getting new suggestions by the system and checking the new suggestions by the collector
AGENTS INVOLVED	Sender: System Receiver: Collector
COMMUNICATION PLAN	Figure 4.1 communication plan
CONSTRAINT	:- (Suggestions must be found)

4.2.4 Transaction 4

Communication model	Transaction description worksheet CM-1
TRANSACTION IDENTIFIER/NAME	Show suggestions
INFORMATION OBJECT	Transaction between checking the suggestions and agreeing to expand the collection
AGENTS INVOLVED	Sender: Collector Receiver: Board
COMMUNICATION PLAN	Figure 4.1 communication plan
CONSTRAINT	:- (Collector must agree with the system to handover list of suggestions)

4.2.5 Transaction 5

Communication model	Transaction description worksheet CM-1
TRANSACTION IDENTIFIER/NAME	Agreeing to expand
INFORMATION OBJECT	Transaction agreeing to expand and buying the suggestion
AGENTS INVOLVED	Sender: Board Receiver: Collector
COMMUNICATION PLAN	Figure 4.1 communication plan
CONSTRAINT	:- (Board must give a go to buy)

Chapter 5

Design model

5.1 Worksheet DM 1

System architecture

Design Model	**Worksheet DM-1: System Architecture**
Architecture decision	**Format**
SUBSYSTEM STRUCTURE	The system is based on the MVC Model
CONTROL MODEL	Call-return model is used, which handles the call for information from the user and returns said information
SUB-SYSTEM DECOMPOSITION	The subsystem 'application model' decomposes into queries. The decomposition therefore follows query-oriented principles.

5.2 Worksheet DM 2

Target implementation platform

Design Model	Worksheet DM-2: Target Implementation Platform
SOFTWARE PACKAGE	Fuseki & SPARQL
POTENTIAL HARDWARE	Hardware that is able to run Fuseki server with SPARQL database
TARGET HARDWARE	Personal computers
VISUALIZATION LIBRARY	YASUI & web browser
LANGUAGE TYPING	SPARQL for entering queries
KNOWLEDGE REPRESENTATION	Triple pattern (subject, predicate, object)
INTERACTION PROTOCOLS	SPARQL libraries
CONTROL FLOW	Not needed
COMMONKADS SUPPORT	No

Chapter 6

Reflection

For the comparison of the plants we used the following attributes:

Abiotic features

- Soil
- pH
- Hardiness
- Light facing aspect

Sadly we weren't able to use the attributes that denote the plant's color as well as the area where they grow in. This was because of some constraints with the web crawler used to gather data for the database.

Also we wanted to be able to query multiple plants at once to compare them to each other, yet we did not manage to make that work. Neither did the comparison of the plant's specific situation to the plant's normal situation work, this has to be done manually.

6.1 Future work

Unfortunately, the amount of time allowed us to create a system where it is only possible to submit 1 plant. So querying multiple plants at once so we can make a more specific selection of plants would be a huge improvement. It would also be nice to be able to compare the plants situation and give advice based on input data. Now, suggestions based on the features of a certain plant. It would be great if you could also submit the features immediately and separately from your current collection.

Part III

Appendix

Chapter 7

JavaScript code

```
if (!String.prototype.format) {
  String.prototype.format = function() {
  var args = arguments;
  return this.replace(/{(\d+)}/g, function(match, number) {
    return typeof args[number] != 'undefined' ? args[number] : match
    ;
  });
  };
}
if(!Array.prototype.diff){
  Array.prototype.diff = function(a) {
    return this.filter(function(i) {return a.indexOf(i) < 0;});
  };
}
function merge() {
  var obj, name, copy,
    target = arguments[0] || {},
    i = 1,
    length = arguments.length;

  for (; i < length; i++) {
    if ((obj = arguments[i]) !== null) {
      for (name in obj) {
        copy = obj[name];

        if (target === copy) {
          continue;
        }
        else if (copy !== undefined) {
          target[name] = copy;
        }
      }
    }
  }

  return target;
}


var DEFAULT_ENDPOINT = "https://mwent.info/fuseki/ke/query";
```

```
var DEFAULT_PREFIX =   "PREFIX rdf: <http://www.w3.org/1999/02/22
    -rdf-syntax-ns#> " + //
        "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> " + //
        "PREFIX {0} {1} ".format("plant:", "<http://mwent.info/ke/fake/
            plant#>");

$(document).ready(function() {

});

var getPlantNames = function(values){
  values = merge({
      succes: function(json){var res = new ResultReturner(json, {}).toVar()
          ; console.log(res);}
    }, values || {});
  var literal = {
    name: "name",
    common: "common_name"
  };
  var query = "SELECT ?{0} (group_concat(distinct ?{1} ; separator = \",
      \") AS ?{1}s) " + //
        "WHERE {" +
        "  ?plant plant:common_name ?{1}; " +
        "      plant:name ?{0} " +
        "} GROUP BY ?plant ?{0} " +
        "ORDER BY ?{1} ";
  query = DEFAULT_PREFIX + query.format(literal.name, literal.common);

  queryFunc(DEFAULT_ENDPOINT, query, {
    callback : {
      success : values.succes
    }
  });
};

var getPlantDetails = function(values){
  values = merge({
      succes: function(json){
        var res = new ResultReturner(json, {}).toVar();
        res = res[0];
        for(key in res){
          $("#data").append("{0}: {1}".format(key, res[key].value));
          $("#data").append("<br />");
        }
        getCompatiblePlants(res.aspect.value, res.soil.value, res.hardiness
            .value, res.ph.value);
      }
    }, values || {});
  var literal = {
    name: "name",
    common: "common_name",
    plant: "plant",
    area: "area",
    hardiness: "hardiness",
    ph: "ph",
    soil: "soil",
    aspect: "aspect"
  };
  var query = "SELECT ?{1} ?{2} ?{3} ?{4} " + //
```

```
        "WHERE {" +
        "  ?{5}      plant:aspect     ?{1}; " +
        "            plant:soil      ?{2}; " +
        "            plant:hardiness  ?{3}; " +
        "            plant:ph        ?{4}. " +
        "  {?{5} plant:common_name \"{0}\". } " +
        "  UNION { ?{5} plant:name \"?{0}\" .} " +
        "} GROUP BY ?{1} ?{2} ?{3} ?{4} " +
        "LIMIT 1";
  query = DEFAULT_PREFIX + query.format(values.plantNames, literal.aspect,
      literal.soil, literal.hardiness, literal.ph, literal.plant);

  queryFunc(DEFAULT_ENDPOINT, query, {
    callback : {
      success : values.succes
    }
  });
};

var getCompatiblePlants = function(values){
  values = merge({
      succes: function(json){
        var res = new ResultReturner(json, {}).toVar();
        for(var i = 0; i < res.length; i++){
          $("#suggestions").append(res[i].common_name.value);
          $("#suggestions").append("<br />");
        }
      }
    }, values || {});


  var literal = {
    name: "name",
    common: "common_name",
    plant: "plant",
    area: "area",
    hardiness: "hardiness",
    ph: "ph",
    soil: "soil",
    aspect: "aspect"
  };

  var query = "SELECT ?{0} ?{1} ?{2} ?{3} (group_concat(distinct ?{4}s ;
      separator = \", \") AS ?{4}) (group_concat(distinct ?{5}s ; separator
       = \", \") AS ?{5}) (group_concat(distinct ?{6}s ; separator = \",
      \") AS ?{6}) (group_concat(distinct ?{7}s ; separator = \", \") AS
      ?{7}) " +
  "WHERE " +
  "{ " +
  "{  ?{0} plant:name ?{1} ; " +
  "      plant:common_name ?{6}s ; " +
  "      plant:aspect ?{7}s ; " +
  "      plant:area ?{2}; " +
  "      plant:hardiness ?{3}; " +
  getMultiVal(values.ph, "ph") +
  getMultiVal(values.soil, "soil") +
  getMultiVal(values.aspects, "aspect") +
  ".  } " +
  //getNegAspects(aspects) + //
```

```
  //getNegPh(ph) + //
  "  FILTER regex(?{2}, \"{0}\"). ".format((values.area || "")) + //
  "  FILTER regex(?{3}, \"H[{8}-9]\"). " + //
  "} GROUP BY ?{0} ?{1} ?{2} ?{3} " + //
  "LIMIT 30 ";
  query = DEFAULT_PREFIX + query.format(literal.plant, literal.name,
      literal.area, literal.hardiness, literal.ph, literal.soil,  literal.
      common, literal.aspect, parseInt(values.hardiness[1]));

  var success = function(json){
    var res = new ResultReturner(json, {}).toVar();
    //console.log(res);
    for(var i = 0; i < res.length; i++){
      $("#suggestions").append(res[i].common_name.value);
      $("#suggestions").append("<br />");
    }
  };

  queryFunc(DEFAULT_ENDPOINT, query, {
    callback : {
      success : values.succes,
      error: function(err){console.log(err);}
    }
  });
};
var getMultiVal = function(arr, subject){
  if(!(arr instanceof Array)){
    var tmp = [];
    tmp.push(arr);
    arr = tmp;
  }
  var val = "";
  for(var i = 0; i < arr.length; i++){
    val += " plant:{0} \"{1}\"; ".format(subject, arr[i]);
  }
  return val;
};

var getNegAspects = function(aspects){
  if(!(aspects instanceof Array)){
    var tmp = [];
    tmp.push(aspects);
    aspects = tmp;
  }
  var negAspects = ["North-facing", "South-facing", "East-facing", "
      West-facing"].diff(aspects);
  var val = "";
  for(var i = 0; i < negAspects.length; i++){
    val += " MINUS {?plant plant:aspect \"{0}\".} ".format(negAspects[i]);
  }
  return val;
};
var getNegPh = function(ph){
  if(!(ph instanceof Array)){
    var tmp = [];
    tmp.push(ph);
    ph = tmp;
  }
  var negPh = ["Acid", "Alkaline", "Neutral"].diff(ph);
```

```
205   var val = "";
206   for(var i = 0; i < negPh.length; i++){
207     val += " MINUS {?plant plant:ph \"{0}\".} ".format(negPh[i]);
208   }
209   return val;
210 };
211 function queryFunc(endpoint, query, options) {
212   ajaxQueryFunc(endpoint, query, options);
213 }
214
215
216 function shuffle(o) {
217   for (var j, x, i = o.length; i; j = Math.floor(Math.random() * i), x = o[
          --i], o[i] = o[j], o[j] = x);
218   return o;
219 }
```

Chapter 8

PHP Code

```php
<?php

require_once ("sparqllib.php");
error_reporting(-1);
ini_set('display_errors', 'On');
switch($_SERVER['REQUEST_METHOD'])
{
case 'GET': $the_request = &$_GET; break;
case 'POST': $the_request = &$_POST; break;
default:
}

$query = $the_request['query'];
$endpoint = $the_request['endpoint'];
$type = $the_request['dataType'];


$db = sparql_connect($endpoint);
if (!$db) { print $db -> errno() . ": " . $db -> error() . "\n";
  exit ;
}


$result = $db -> query($query, null, $type);

if (!$result) { echo json_encode(array("error" => $db -> errno() . ": " .
    $db -> error() . "\n"));
  exit ;
}

switch ($type) {
  case 'json' :
    echo json_encode($result);
    break;
  default :
    break;
}
?>
```

Chapter 9

HTML Code

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
    <meta charset="utf-8">
    <title>Knowledge Engineering</title>

    <link rel='shortcut icon' href='img/favicon.ico' />
    <link rel="stylesheet" href="//code.jquery.com/ui/1.10.3/themes/
        smoothness/jquery-ui.css" />
    <link href='//fonts.googleapis.com/css?family=Roboto:300' rel='
        stylesheet' type='text/css'>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.
        js"></script>
    <script src="//code.jquery.com/ui/1.11.2/jquery-ui.js"></script>
    <!-- <script src="js/sparql.js"></script>
    <script src="js/snorql.js"></script> -->
    <script src="js/test.js"></script>
    <script src="js/index.js"></script>
    <script>
      var compPlantFunc = function(json){
        var res = new ResultReturner(json, {}).toVar();
        for(var i = 0; i < res.length; i++){
          var suggestion = $('<span />').addClass('plant_suggestions').html
              (res[i].common_name.value);
          //$(suggestion).click(function(){
          //  loadPlant($(this).val());
          //});
          $("#suggestions").append(suggestion);
          $("#suggestions").append("<br />");
        }
      };
      var plantDetailFunc = function(json){
        var res = new ResultReturner(json, {}).toVar();
        res = res[0];
        for(key in res){
          $("#data").append("{0}: {1}".format(key, res[key].value));
          $("#data").append("<br />");
        }
        getCompatiblePlants({
          aspects: res.aspect.value,
```

```
          soil: res.soil.value,
          hardiness: res.hardiness.value,
          ph: res.ph.value,
          succes: compPlantFunc
        });
      };
      var loadPlant = function(name){
        $("#suggestions").empty();
        $("#data").empty();
        getPlantDetails({
          plantNames: name.split(",")[0],
          succes:  plantDetailFunc
        });
      };
      var plantNameFunc =  function(json){
        var res = new ResultReturner(json, {}).toVar();
        var tags = [];
        for(var i = 0; i < res.length; i++){
          tags.push(res[i].common_names.value);
        }
        $( "#names" ).autocomplete({
          source: tags,
          select: function( event, ui ) {
            loadPlant(ui.item.value);
          }
        });
      };

      $(document).ready(function(){
        getPlantNames({
          succes: plantNameFunc
        });
      });

    </script>
    <style>
      label {
        color: #B4886B;
        font-weight: bold;
        display: block;
        width: 150px;
        float: left;
        clear:both;
        }
        label + div{
          clear: both;
          float: left;
          margin-left: 15px;
        }
    </style>
  </head>
  <body>
    <div>
      <label for="names">Names: </label>
      <input id="names"></div>
    <div>
      <label for="data">Data for the plant: </label>
      <div id="data"></div></div>
    <div>
```

```
98        <label for="suggestions">Suggestions: </label>
99        <div id="suggestions"></div></div>
100     <script>
101       (function(i, s, o, g, r, a, m) {
102         i['GoogleAnalyticsObject'] = r;
103         i[r] = i[r] ||
104         function() {
105           (i[r].q = i[r].q || []).push(arguments);
106         }, i[r].l = 1 * new Date();
107         a = s.createElement(o), m = s.getElementsByTagName(o)[0];
108         a.async = 1;
109         a.src = g;
110         m.parentNode.insertBefore(a, m);
111       })(window, document, 'script', '//www.google-analytics.com/analytics.
            js', 'ga');
112
113       ga('create', 'UA-43774734-1', 'mwent.info');
114       ga('send', 'pageview');
115
116     </script>
117   </body>
118 </html>
```