easyAround

Knowledge Engineering project

Claudia Minardi
minardi.claudia@gmail.com

Marco De Nadai
me@marcodena.it

Vrije Universiteit
Amsterdam, The Netherlands
2014

1 Introduction

This document illustrates the process of the development of an information system according to the CommonKADS [2] approach.
The idea at the base of the project is to build a system capable of assisting a Travel Agent in satisfying the customers. The software must be able to exploit the knowledge of the Travel Agent in order to build a customized itinerary that reflect the desire of the client.
To do so, it is necessary to have precise knowledge rules embedded inside the system itself: this can be done by building a series of models that will constitute the core structure of the software.
The final piece of software, namely *easyAround* will be able to:

- Classify customers according to their age and physical capabilities;
- Gather information on each customer's preferences and personal taste;
- Gather specific information on each customer's desire for a specific trip;
- Propose to each customer an ideal trip, based on the gathered information;
- Let the customers revise and personalize their own itinerary;

The target domain of the software resides inside one single city: the final itineary will be composed of locations to be visited inside that particular city, according to a standard timetable possessed by the Travel Agency.
The target user of the software is the Travel Agent appointed with the task of creating cusomized itineraries for clients who travel alone or accompanied by children.

2 Context Knowledge

OM-1

Identifying knowledge-oriented problems and opportunities in the organization

Organization Model	**Problems and Opportunities Worksheet OM-1**
Problems and opportunities	Difficulty for the travel agent in designing pesonalized itineraries, due to customers lack of knowledge on the subject and great variety of points of interest in a location. The process of building personalized itinerary is time-consuming for the agent, and could be subjected to multiple revisions or discarded altogether from the client.
Organizational context	**Mission, vision, goals**: efficient itinerary design, customer satisfaction, improving time schedule of the travel agent, increasing the number of satisfied requests; **External factors**: requirements of the client, client profile (age, interests), set up of the destination, geographical topology of the location; **Strategy:** given a list of possible locations, assemble an itinerary that best suits the customer's requirements; 4. Its value chain and the major value drivers
Solutions	Automatization of the selection process for the locations and the revision of compiled itinearies, leaving to the travel agent the task of interacting with the client and proposing the drafts.

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	See Figure 1
Process	See Figure 2
People	Single-customer Travel Agent
Resources	**Database** of locations containing all the available infomation. **Database** of customers containing personal features and preferences. **Designing software** capable of assembling the itinerary.
Knowledge	**Requirement rules**: knowledge to choose a set of locations based on the client features; **Preference rules**: knowledge to favour a some location more than others based on client expressed preferences; **Constraint rules**: knowledge to exclude or include specific locations based on client explicit directives.
Culture & power	The opinion of the client is highly prioritized. Being a small agency no particular power influence is noticeable between co-workers: the hierarchical structure is vertical, with the president occupying the highest position and in charge of all important decisions.

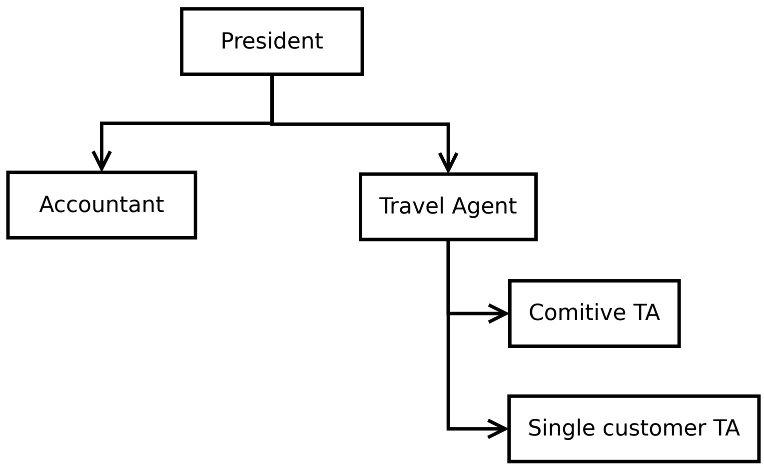

Figure 1: Organization structure

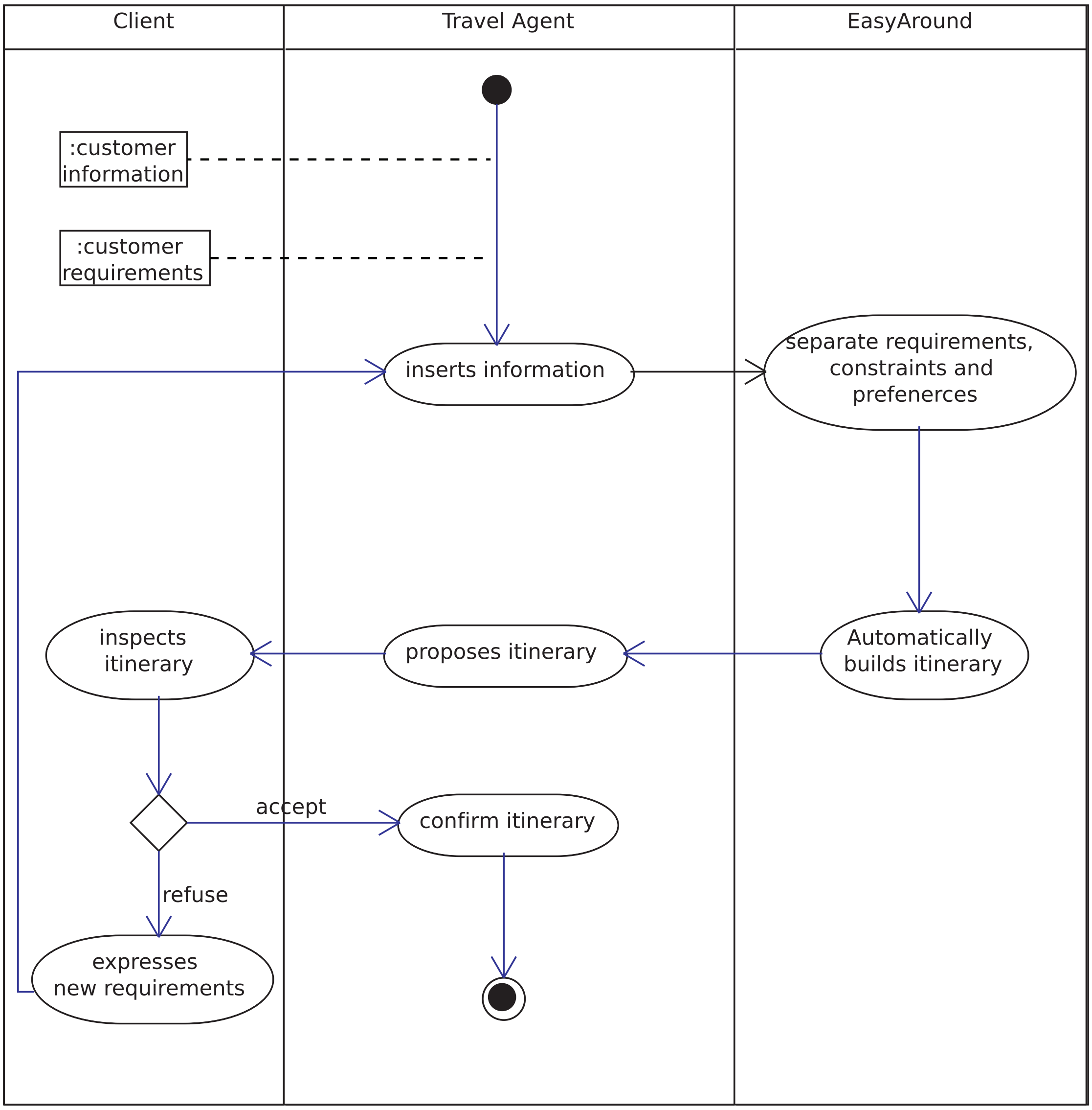

Figure 2: Organization process

OM-5
Checklist for the feasibility decision document

Organization Model	**Checklist for Feasibility Decision Document: Worksheet OM-5**
Business feasibility	**Benefits**: the itinerary process is quicker, the client is more satisfied, travel agents can schedule their work time on a higher number of customers; **Added value**: the speed up should be quite significant, it is expected that the TA can satisfy a client surplus of 30% with the time he saved in building and reviewing degigns. **Costs**: the costs are a summation of the salary of the employees working on building the software (programmers, experts) and the time spent in integrating the licenced content into the automated system; **Organizational Changes**: the system is built to avoid organizational changes. **Risks**: the system could have difficulties in selecting the right locations based on customer's requests, not posing as an advantage to the Travel Agent. In this case the workload would not decrease.
Technical feasibility	**Complexity**: the complexity level of the required reasoning is high, because it need the integration of a lot of informal knowledge into a formal system, and the handling of many constraints; **Critical aspects**: the solution must be developed correctly, otherwise the risk of losing clients grows. Furthermore, if the results are not as expected, the software could not be accepted or used inside the acency. **Success Measures**: if the design is coherent with the requirements, if there are no constraint violations, if it corresponds to the preferences of the client, and it is at least the same or better than a manual design done by the TA, then it is a success. **User Interface**: the UI can be constructed to be very simple and intuitive, requiring no additional knowledge about IT systems from the user. **Additional Interactions**: the only extern interaction is with the structured database of locations, which basic structure is fully impemented and documented in many shapes and programming languages. **Further technological risks**: there are no further risks;

PROJECT FEASIBILITY	**Commitment**: the TAs are interested in a mechanism that allows them to save time for single-customer itinerary design, the president is interested in employing new technologies to increment profit. **Resources**: since the expertise is provided by the agency itself, the necessary resources left are the ones needed for the programmers. Being freelancers, their cost is relatively limited by the absence of an organization that coordinates the work. **Knowledge**: the knowledge is available since it's provided by the agency itself, and it's largely available on public means such as the web; **Expectations**: the expectation are realistic; **Communication**: the communication is efficient, both between the programmers who have worked with each other previously, and between the expert consultant and the team since they are acquaintances.
PROPOSED ACTIONS	1. *Focus:* speed-up of the design process, increased number of customers; 2. *Target solution:* Automatization of the design and revision process; 3. *Results, costs, and benefits:* satisfaction of the client, saved workload and working time for the TA; 4. *Project actions*: building the Knowledge Model, create the Design Model, create the Communication Model, implement the system, embed the knowledge in the software, test the software and collect results; 5. *Risks:* the system could have difficulties in selecting the right locations based on customer's requests, not posing as an advantage to the Travel Agent. In this case the workload would not decrease

TM-1
Refined description of the tasks within the target process

Task Model	**Task Analysis Worksheet TM-1**
Task	Automated Design
Organization	Task is controlled by the Travel Agent and executed by the appointed software. It is the product of non-human intervention.
Goal and value	The goal is the design of an itinerary composed of multiple locations, based on the preferences and the requirements set by the customer.
Dependency and flow	*Input tasks*: Evaluate Request *Output tasks*: Propose Itinerary
Objects handled	*Input objects*: requirements, preferences and constraints from the customer. *Output objects*: itinerary. *Internal objects*: database of locations.
Timing and control	**Frequency and duration**: whenever a client asks for a custom-made itinerary, arbitrarily short duration. **Control relation**: (I) *Preconditions*: the request from the client must be organized in a set of requirements, constraints and preferences; (II) *Postconditions*: the itinerary must satisfy the request of the client.
Agents	Travel Agent
Knowledge and competence	Requirement rules, preference rules, constraint rules.
Resources	Database of exsting locations, automated software for itinerary design, Travel Agent for customer interaction; The duration of the interaction depends on the satisfaction of the client and he number of reviews requested on the itinerary. It should be in every occasion shorter than the duration of an interaction that does not include the automated system.
Quality and performance	If the design is coherent with the requirements, if there are no constraint violations, if it corresponds to the preferences of the client, and it is at least the same or better than a manual design done by the TA, then it is of good quality.

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	Requirement Rules	
POSSESSED BY	Travel Agent	
USED IN	Automated Design.	
DOMAIN	Travel Planning	
Nature of the knowledge		**Bottleneck / to be improved?**
Formal, rigorous		
Empirical, quantitative	X	X
Heuristic, rules of thumb	X	X
Highly specialized, domain-specific	X	
Experience-based	X	
Action-based		
Incomplete		
Uncertain, may be incorrect	X	X
Quickly changing		
Hard to verify	X	X
Tacit, hard to transfer	X	X
Form of the knowledge		
Mind	X	
Paper		
Electronic		
Action skill		
Other		
Availability of knowledge		
Limitations in time		
Limitations in space		
Limitations in access		
Limitations in quality	X	X
Limitations in form		

Task Model	Knowledge Item Worksheet TM-2
NAME	Preference Rules
POSSESSED BY	Travel Agent
USED IN	Automated Design.
DOMAIN	Travel Planning

Nature of the knowledge		**Bottleneck / to be improved?**
Formal, rigorous		
Empirical, quantitative	X	X
Heuristic, rules of thumb	X	X
Highly specialized, domain-specific	X	
Experience-based		
Action-based		
Incomplete		
Uncertain, may be incorrect	X	X
Quickly changing	X	X
Hard to verify	X	X
Tacit, hard to transfer	X	X
Form of the knowledge		
Mind	X	
Paper		
Electronic		
Action skill		
Other		
Availability of knowledge		
Limitations in time	X	X
Limitations in space		
Limitations in access		
Limitations in quality	X	X
Limitations in form		

Task Model	Knowledge Item Worksheet TM-2
NAME	Constraint Rules
POSSESSED BY	Travel Agent
USED IN	Automated Design.
DOMAIN	Travel Planning

Nature of the knowledge		Bottleneck / to be improved?
Formal, rigorous	X	
Empirical, quantitative		
Heuristic, rules of thumb		
Highly specialized, domain-specific	X	
Experience-based		
Action-based		
Incomplete		
Uncertain, may be incorrect		
Quickly changing	X	X
Hard to verify		
Tacit, hard to transfer		
Form of the knowledge		
Mind	X	
Paper		
Electronic		
Action skill		
Other		
Availability of knowledge		
Limitations in time	X	X
Limitations in space		
Limitations in access		
Limitations in quality		
Limitations in form		

AM-1
Agent specification according to the CommonKADS agent model

Agent Model	**Agent Worksheet AM-1**
NAME	*Single-customer Travel Agent*
ORGANIZATION	Human, sub-category of the Travel Agent
INVOLVED IN	Automated Design
COMMUNICATES WITH	Customer
KNOWLEDGE	Requirement rules, Preference rules, Constraint rules
OTHER COMPETENCES	Social skills to interact with a customer
RESPONSIBILITIES AND CONSTRAINTS	Collect the request from the client, and provide the customer's personal featues to the software; supervise the automated process of design and propose the itinerary to the customer; modify the request in case of review of the proposed itinerary.

3 Task Knowledge

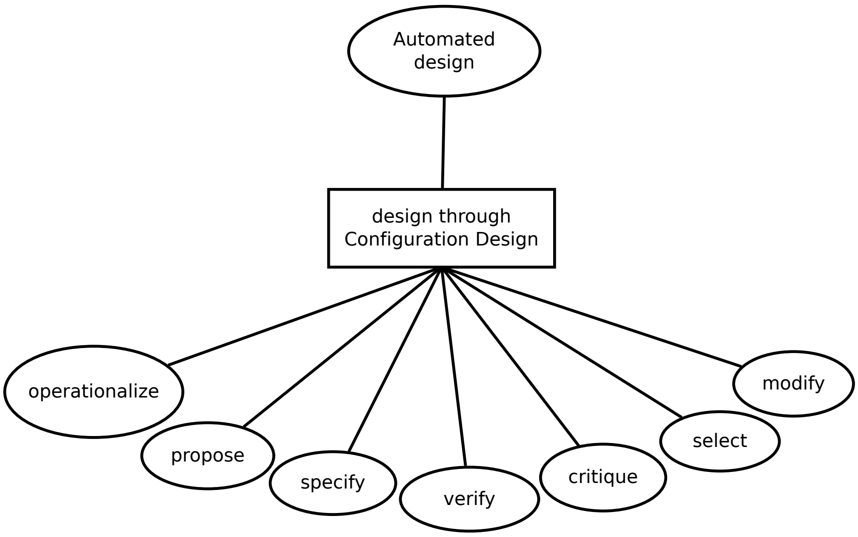

Figure 3: Task knowledge

The “propose and revise” method for configuration design presented in its original form in the textbook for the course has been slightly modified to obtain a method that reflects the needs of our software. The *WHILE* loop to revise the the design has been postponed from a state of *propose* to a state of *verify*, and the internal *REPEAT UNTIL* loop to select the actions has been integrated in the outer cycle. This way the method reflects exactly the intended steps to be realized in the software.

Listing 1: Task and task method description

```
TASK automated-design;
  ROLES:
    INPUT: request: "request for the design";
    OUTPUT: itinerary: "the resulting design";
END TASK configuration-design;

TASK-METHOD propose-and-revise;
  REALIZES: automated-design;
  DECOMPOSITION:
    INFERENCES: operationalize, propose, specify, verify,
        critique, select, modify;
  ROLES:
    INTERMEDIATE:
      preferences-and-requirements: "requirements and
         preferences to be preferably fulfilled";
```

```
      constraints: "requirements that have to be fulfilled";
      skeletal-design: "set of slots to be filled";
      proposal: "a possible compilation of the skeletal-design
         ";
      customer-input: "set of new requirements or constraints";
      violation: "new constraints violated by the current
         design";
      truth-value: "boolean indicating the result of the
         verification";
      action-list: "ordered list of possible repair (fix)
         actions";
      action: "a single repair action";
      itinerary: "a new possible compilation of the skeletal-
         design";
  CONTROL-STRUCTURE:
    operationalize(request -> preferences-and-requirements +
       constraints);
    specify(request -> skeletal-design);
    propose(constraints + preferences-and-requirements +
       skeletal-design -> proposal);
    itinerary := proposal ADD itinerary;
    WHILE verify(customer-input + itinerary -> truth-value +
       violation) IS truth-value == false DO
      critique(violation + itinerary -> action-list)
      select(action-list -> action)
      modify(itinerary + action -> itinerary)
      verify(itinerary + customer-input -> truth-value +
         violation);
    END WHILE
END TASK-METHOD propose-and-revise;
```

4 Inference Knowledge

As inference model we use a modified version of the Configuration design template, because given predefined components we need to find and assembly that satisfies the requirements. The inference model deriving from this task can be found in Figure 4. The standard inference model for the configuration design template (propose and revise) has been modified to better express the needs of our software, in a way that the system interacts directly with the client a second time right after the proposal phase:

- Requirements have been transformed in Request
- Soft and hard requirements have been transformed in "preferences and requirements" and "constraints" respectively
- Extension has been changed into "proposal"
- Verify requires the direct input of the customer, since it's a verification of subjective correctness more than a verification of constraint violation.
- Design has been changed int "itinerary" for coherency purposes.

It has to be noted that the subsystem building the proposal, in the implementation of the system, does not permit a constraint violation, so the verification of the constrait has been removed because redundant.

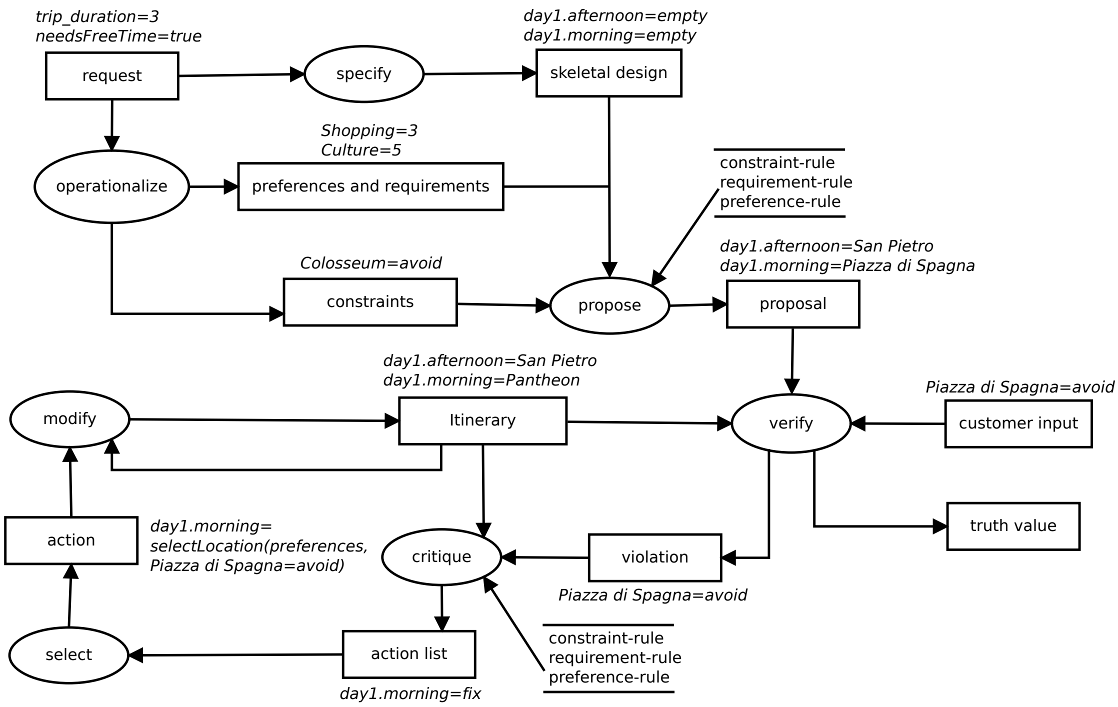

Figure 4: Inference structure

Inference	Input	Output	Description
specify	request	skeletal design	the function builds the default skeletal design: the basic structure of a trip where each day is composed of: morning, afternoon, evening and meal.
operationalize	needs of the customers	preferences, requirements, constraints	the needs and desires are translated into preferences ("I would like to have time for shopping and visit many cultural places. I am not interested so much in food places"), requirements ("I want a quiet trip") and contraints ("In Rome I want to visit the *Colosseum* and avoid *Piazza di Spagna*").
propose	preferences and requirements, skeletal design and constraints	filled skeletal design	fill the slots of the skeletal design with locations that fits the preferences and requirements.
verify	proposal and customer input	-	it submits the proposal to the TA allowing them to gather the client's critiques.
select	fix actions list	fix action	It simply selects an action from the fix actions list generated by the critique function.
modify	itinerary design, fix action	itinerary	it applies the fix actions to the proposal.
critique	itinerary and violations	fix actions list	it creates a series of actions which will adjust the itinerary according to the new set of contraints, contained into the violations.

5 Domain knowledge

5.1 Domain schema

The domain schema can be found in Figure 5

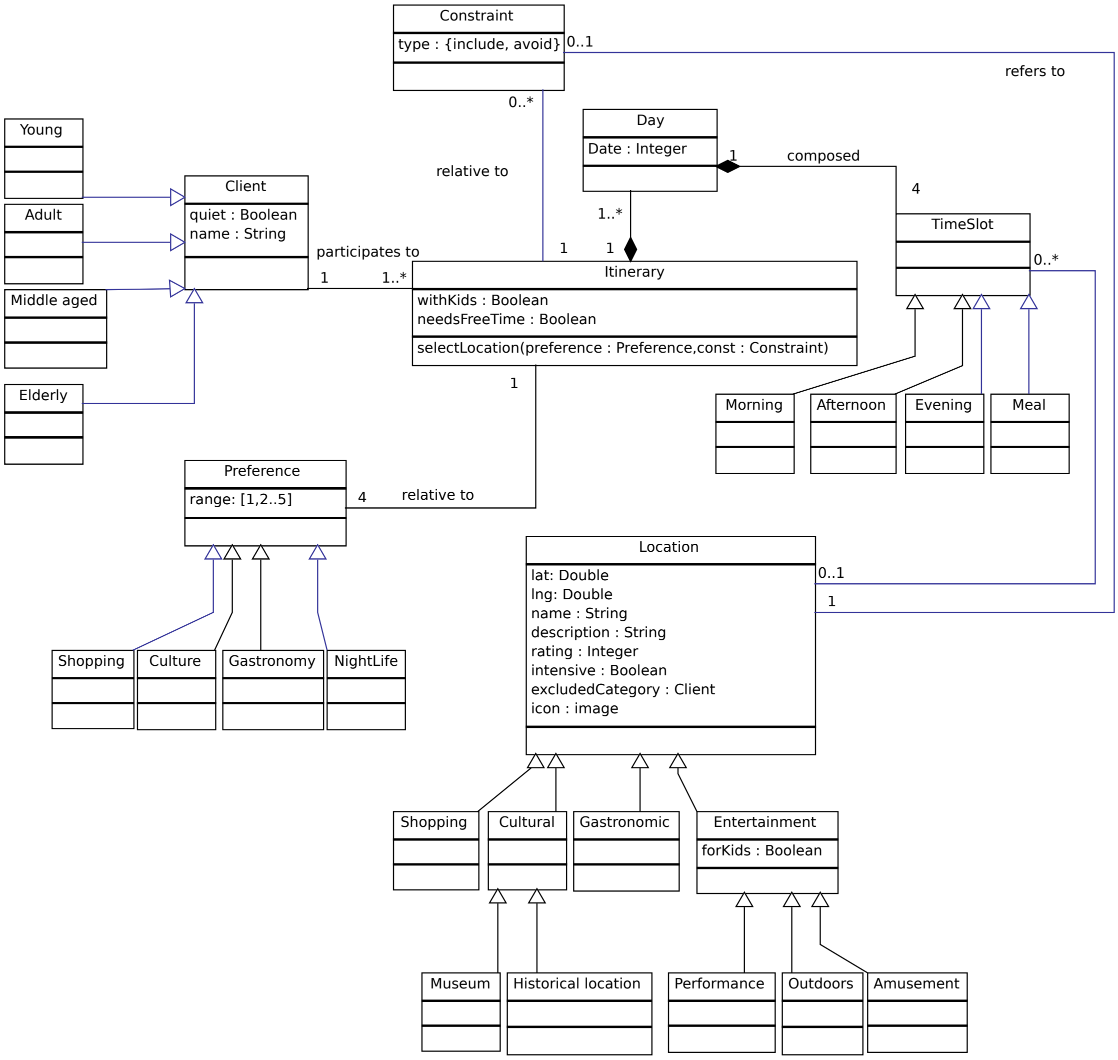

Figure 5: Domain schema

This schema seems complicated, for this reason every model is explained in the following list:

Client

The client who goes to the travel agency. He can express preferences for a quiet enviroment. The clients are categorized by their age to avoid unsuitable locations (ex: elderly people in a climbing location).

Preference

Each client needs to specify a list of preferences, valued from 1 to 5, where 1 is "I'm not so interested" and 5 is "I'd love to do it!". These preferences are related to the itinerary we want to create, consequently if the same clients wants to create another itinerary, it will specify again all the preferences he wants in this second trip.

Constraint

Each client needs to specify a list of contraints that have to be fulfilled. As well as the *Preference*, they are related to the single itinerary.

Itinerary

This represents the itinerary we want to create. It is composed of a fixed number of *Day* and it is related to a *Client* who has specified his own list of *Constraint* and *Preference*. If the itinerary includes kids, the system needs to select some *Location* that could entertain them. This is a requirement as the *needsFreeTime* attribute, which specifies that the clients needs to have some time not allocated by the TA. The method *selectLocation* takes a list of *Preference* and produces a list of *Location* that could fit this preferences.

Day

This describes a day of the itinerary.

Timeslot

A timeslot is a fixed part of a day. The division of the day came from the expert interview.

Location

This model represents the point of interests that a customer could visit. The attribute *rating* describes the quality of this place, *intensive* describes if the place is not for quite people and *excludedCategory* specifies if a client category is not suitable for the location (ex: elderly people in a climbing location).

5.2 Domain mapping

Knowledge Role	Type	Domain Mapping
request	dynamic	Client
skeletal design	static	Timeslot
preferences and requirements	dynamic	Client, Itinerary, Preference
constraints	dynamic	Constraint
customer input	dynamic	Constraint
proposal	dynamic	Itinerary
itinerary	dynamic	Itinerary
violation	dynamic	Constraint, Location
constraint-rule	static	Constraint
preference-rule	static	Preference
requirement-rule	static	Client, Itinerary

5.3 Rule types

Listing 2: Rules

```
RULE TYPE constraint-rule;
    DESCRIPTION: "rule stating the relation between client and
        the choice for a location in the itinerary, by means of
        defining strict boundaries that must be respected.";
ANTECEDENT: Client;
CONSEQUENT: Itinerary;
CONNECTION-SYMBOL: restricts;
END RULE-TYPE constraint-rule;

RULE TYPE requirement-rule;
    DESCRIPTION: "rule stating the relation between the client
        and the choice for a location in the itinerary, by means
         of defining boundaries that should be respected.";
ANTECEDENT: Client;
CONSEQUENT: Itinerary;
CONNECTION-SYMBOL: requires;
END RULE-TYPE requirement-rule;

RULE TYPE preference-rule;
    DESCRIPTION: "rule stating the relation between the client
        and the choice for a location in the itinerary, by means
         of defining preferences that could be satisfied with
        probability X (calculated on the input values) .";
```

```
ANTECEDENT: Client;
CONSEQUENT: Itinerary;
CONNECTION-SYMBOL: prefers-with-probability;
END RULE-TYPE preference-rule;
```

Here are presented also some example in order to better understand all the rule types.

Listing 3: The client wants to include a destination into the itinerary.

```
client.constraint.location.name=A AND client.constraint.type=
    include
RESTRICTS
∃itinerary.day.timeslot, timeslot.location.name=A;
```

Listing 4: The client is a quite person

```
client.quiet=true, client.needsFreeTime=true
REQUIRES
itinerary.day.timeslot.location, location.intensive=false;
itinerary.day.timeslot, timeslot.location=NULL;
```

Listing 5: The client expresses four preferences with four ranges (from 1 to 5). The method selectLocation will compose the itinerary selecting the locations that fits the preferences. For example it could select 3 shopping 1 gastronomy and 1 cultural locations.

```
Var A, B, C, D: client.preference;
Var E: client.constraint;
A.type=shopping AND A.range=x
B.type=cultural AND B.range=y
C.type=gastronomy AND C.range=w
D.type=nightlife AND D.range=z
E.type = avoid AND E.location = Colusseum
PREFERS-WITH-PROBABILITY
∀itinerary.day.timeslot, timeslot.location=selectLocation(A, B,
    C, D, E);
```

5.4 Knowledge Base

The Knowledge base can be seen in Figure 6. In the model it is shown the relation between instances of the types specified in the domain schema, according to the rules used to build the system.

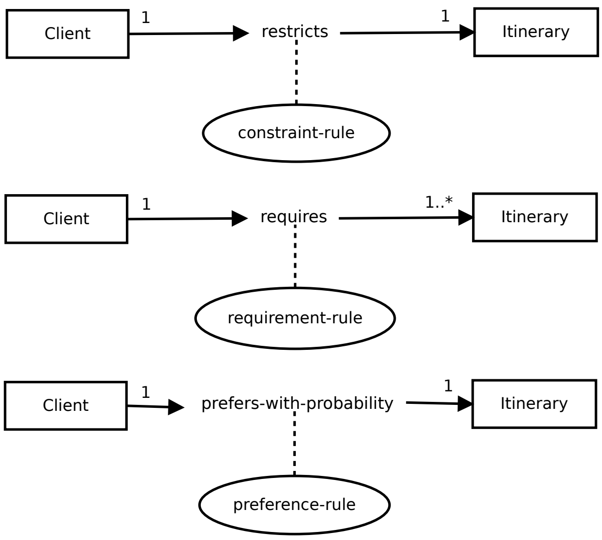

Figure 6: Knowledge base

6 Scenarios

6.1 Scenario 1

Rose, a 76 years old lady would like to visit Rome for three days with her nephew John who is ten years old. She would like to have her trip planned but with the possibility to explore the city on her own.

6.1.1 Interview

In a scale from one to five, how do you enjoy shopping?
I really love to do shopping, so five!

In a scale from one to five, how do you enjoy cultural places?
I am going to Rome, so four!

In a scale from one to five, how do you enjoy nightlife?
Have you looked at me? 1!

In a scale from one to five, how much would you like to try new restaurants?
I guess... I don't know, 3?

Is there anything that you'd absolutely like to see?
Yes, I've never seen the Colosseum.

Is there anything that you have already seen or don't want to see?
Not really, everything is fine.

The travel agent while is interviewing the customer, inserts the acquired data into the system through graphic interface.

6.1.2 Operazionalize

The system once it receives the request divides them into three categories: requirements, preferences and constraints.

```
REQUIREMENTS:
  client.quiet = true
  itinerary.withKids = true
  itinerary.needsFreeTime = true
  VAR a,b,c: day;
  a.date = "14/05/2014"
  b.date = "15/05/2014"
  c.date = "16/05/2014"
PREFERENCES:
  VAR a,b,c,d : preference;
  a.type = shopping
  a.range = 5
  b.type = culture
  b.range = 4
  c.type = nightlife
  c.range = 1
  d.type = gastronomy
  d.range = 3
CONSTRAINTS:
  constraint.location.name = Colosseum
  constraint.type = include
```

6.1.3 Specify

The system reads the request and compiles the skeletal design, an empty itinerary containing only the structure of the days.

```
REQUIREMENTS:
  VAR a,b,c: day;
  a.date = "14/05/2014"
  b.date = "15/05/2014"
  c.date = "16/05/2014"
SKELETAL-DESIGN:
  NEW-ITINERARY(a.date, c.date)
```

6.1.4 Propose

The system processes the request using the knwoledge rules and returns a first version of the itinerary.

```
ITINERARY:
 VAR a,b,c: day;
  a.date = "14/05/2014"
  b.date = "15/05/2014"
  c.date = "16/05/2014"
  a.morning = Colosseum
  a.afternoon = Shopping mall "I gladiatori"
  a.meal = Parolaccia
  a.evening = Fontana di Trevi
  b.morning = Villa Borghese
  b.afternoon = Outlet shoes Roma
  b.meal = Pizzeria da Matteo
  b.evening = Piazza di Spagna
  c.morning = EMPTY
  c.afternoon = EMPTY
  c.evening = Piazza del Popolo
```

6.1.5 Verify

The system passes the itinerary to the TA through the GUI. The TA asks the Client for a confirmation or the need for modification.

This is a possible itinerary, do you have any modifications you want to do?

> Yes please, I don't need shoes.

6.1.6 Critique

Based on the Client feedback, the system builds an action list of modifications

```
ACTION-LIST:
  contraint.location = Outlet shoes Roma
  constraint.type = avoid
```

6.1.7 Select

The system chooses one action at the time for the itinerary to be modified

```
ACTION:
  contraint.location = Outlet shoes Roma
  constraint.type = avoid
```

6.1.8 Modify

The system modifies the itinerary accordingly to the selected action.

```
ITINERARY:
  VAR a,b,c: day;
   a.date = "14/05/2014"
   b.date = "15/05/2014"
   c.date = "16/05/2014"
   a.morning = Colosseum
   a.afternoon = Shopping mall "I gladiatori"
   a.meal = Parolaccia
   a.evening = Fontana di Trevi
   b.morning = Villa Borghese
   b.afternoon = Le Piramidi
   b.meal = Pizzeria da Nando
   b.evening = Piazza di Spagna
   c.morning = EMPTY
   c.afternoon = EMPTY
   c.meal = EMPTY
   c.evening = Piazza del Popolo
```

6.1.9 Verify

The system passes the itinerary to the TA through the GUI. The TA asks the Client for a confirmation or the need for modification.

This is a possible itinerary, do you have any modifications you want to do?

No, the itinerary is fine.

6.2 Scenario 2

Richard a 30 years old guy, would like to visit Rome for two days alone.

6.2.1 Interview

Would you consider yourself a quite person or ready to have some fun?
Definitely have fun.

In a scale from one to five, how do you enjoy shopping?
Not that much I only need to buy some souvenirs, 1.

In a scale from one to five, how do you enjoy cultural places?
I am going to Rome, so four!

In a scale from one to five, how do you enjoy nightlife?
I don't know... 5?

In a scale from one to five, how much would you like to try new restaurants?
I definitely like to eat, 5.

Is there anything that you'd absolutely like to see?
Yes, I've never seen the EUR.

Is there anything that you have already seen or don't want to see?
I'm not interested in San Pietro.

The travel agent while is interviewing the customer, inserts the acquired data into the system through graphic interface.

6.2.2 Operazionalize

The system once it receives the request divides them into three categories: requirements, preferences and constraints.

Listing 6: Domain instance of the data inserted into the system

```
REQUIREMENTS:
  client.quiet = false
  itinerary.withKids = false
  itinerary.needsFreeTime = false
  VAR a,b,c,d: day;
  a.date = "14/05/2014"
  b.date = "15/05/2014"
PREFERENCES:
  VAR a,b,c,d : preference;
  a.type = shopping
  a.range = 1
  b.type = culture
  b.range = 4
  c.type = nightlife
  c.range = 5
  d.type = gastronomy
  d.range = 5
CONSTRAINTS:
  VAR a,b : constraint;
  a.location.name = EUR
  a.type = include
  a.location.name = San Pietro
  a.type = avoid
```

6.2.3 Specify

The system reads the request and compiles the skeletal design, an empty itinerary containing only the structure of the days.

```
REQUIREMENTS:
  VAR a,b,c: day;
   a.date = "14/05/2014"
   b.date = "15/05/2014"
SKELETAL−DESIGN:
  NEW−ITINERARY(a.date, b.date)
```

6.2.4 Propose

The system processes the request using the knwoledge rules and returns a first version of the itinerary.

```
ITINERARY:
  VAR a,b,c: day;
   a.date = "14/05/2014"
   b.date = "15/05/2014"
   a.morning = Colosseum
   a.afternoon = Pantheon
   a.meal = Parolaccia
   a.evening = Discoteca el muendo
   b.morning = Villa Borghese
   b.afternoon = EUR
   b.meal = Pizzeria da Matteo
   b.evening = Discoteca Roma
```

6.2.5 Verify

The system passes the itinerary to the TA through the GUI. The TA asks the Client for a confirmation or the need for modification.

This is a possible itinerary, do you have any modifications you want to do?

> No, the itinerary is fine.

7 Communication Knowledge

The communication model specifies the information exchange between tasks carried out by different agents. It has been designed as an Activity Diagram where the name of the interaction corresponds to the type of communication happensing between the agents in the connected lanes.
It has to be noted that the only task knowledge intensive is "Automatically build the itinerary" carried out by the automated system *easyAround*.
The mapping between the activities in the communication model and the inference model is schematized as follows:

- "operationalize" corresponds to "differentiate constraints, requirements and preferences"
- "specify" and "propose" are mapped to "automatically build the itinerary"
- "verify" corresponds to "obtain feedback"
- "critique" corresponds to "edit itinerary"

Our Communication Process can be see in Figure 7.

CM-1
Transaction Descritpion worksheets

Communication model	**Transaction Description Worksheet CM-1**
Transaction identifier/name	Provide requirements
Information object	Transmission of *requirements* between the starting point and *obtain requirements*
Agents involved	Sender: Client Receiver: Travel Agent
Communication plan	Figure 7
Constraints	Precondition: - Postcondition: the TA has the requirements
Information exchange specification	Oral transaction not concerning the automated system

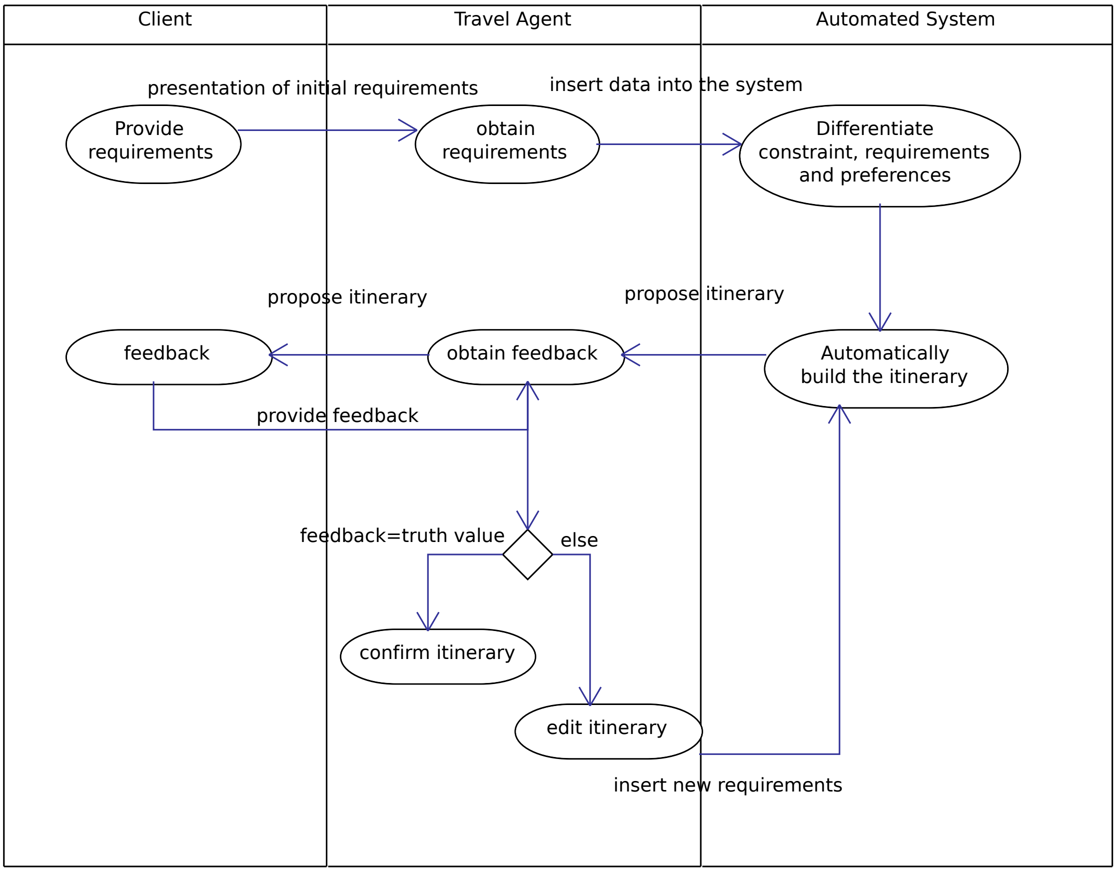

Figure 7: Communication process

Communication model	**Transaction Description Worksheet CM-1**
Transaction identifier/name	Insert data into the system
Information object	Transmission of *requirements* between *obtain requirements* and *Differentiate constraint, requirements and preferences*
Agents involved	Sender: Travel Agent Receiver: Automated System
Communication plan	Figure 7
Constraints	Precondition: the TA must have obtained the requirements Postcondition: the requirements have to be correctly separated into categories
Information exchange specification	See CM-2

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Propose Itinerary
INFORMATION OBJECT	Transmission of *itinerary* between *Automatically build the itinerary* and *Obtain feedback*
AGENTS INVOLVED	Sender: Automated System Receiver: Travel Agent
COMMUNICATION PLAN	Figure 7
CONSTRAINTS	Precondition: the automated system must have already build the itinerary Postcondition: the TA must obtain the itinerary
INFORMATION EXCHANGE SPECIFICATION	See CM-2

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Propose Itinerary
INFORMATION OBJECT	Transmission of *itinerary* between *Obtain feedback* and *feedback*
AGENTS INVOLVED	Sender: Travel Agent Receiver: Client
COMMUNICATION PLAN	Figure 7
CONSTRAINTS	Precondition: the automated system must have already build the itinerary Postcondition: the Client must obtain the itinerary
INFORMATION EXCHANGE SPECIFICATION	Oral transaction not concerning the automated system

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	provide feedback
INFORMATION OBJECT	Transmission of *constraints* between *feedback* and *obtain feedback*
AGENTS INVOLVED	Sender: Client Receiver: Travel Agent
COMMUNICATION PLAN	Figure 7
CONSTRAINTS	Precondition: the Client must have received the itinerary Postcondition: the TA must process the feedback
INFORMATION EXCHANGE SPECIFICATION	Oral transaction not concerning the automated system

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	insert new requirements
INFORMATION OBJECT	Transmission of *constraints* between *edit itinerary* and *Automatically build the itinerary*
AGENTS INVOLVED	Sender: Travel Agent Receiver: Automated system
COMMUNICATION PLAN	Figure 7
CONSTRAINTS	Precondition: the feedback must not be equal to truth-value Postcondition: the Automated system must construct a new itinerary
INFORMATION EXCHANGE SPECIFICATION	See CM-2

CM-2
Information exchange specification

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	*Insert data into the system*
AGENTS INVOLVED	Sender: Travel Agent Receiver: Automated System
INFORMATION ITEMS	**Role**: *core* object **Form**: data string **Medium**: HTTP request
MESSAGE SPECIFICATIONS	**Communication type**: Ask **Content**: startDate: DATE endDate: DATE presenceOfKids: BOOLEAN needsFreeTime: BOOLEAN exclude: LIST include: LIST existingClient: INTEGER clientName: STRING clientAge: INTEGER clientQuiet: BOOLEAN preferenceShopping: INTEGER preferenceCulture: INTEGER preferenceGastronomy: INTEGER preferenceNightLife: INTEGER
CONTROL OVER MESSAGES	Sequence

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	*Propose Itinerary*
AGENTS INVOLVED	Sender: Automated System Receiver: Travel Agent
INFORMATION ITEMS	**Role**: *core* object **Form**: data string **Medium**: HTTP request

MESSAGE SPECIFICATIONS	**Communication type**: Reply
	Content:
	days: {
	    0: {'morning': Location, ...}
	    1: ...
	}
	itineraryID: INTEGER

CONTROL OVER MESSAGES	Sequence

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	*insert new requirements*
AGENTS INVOLVED	Sender: Travel Agent Receiver: Automated system
INFORMATION ITEMS	**Role**: *core* object **Form**: data string **Medium**: HTTP request

MESSAGE SPECIFICATIONS	**Communication type**: Ask
	Content:
	`itineraryID: INTEGER`
	`excludeList: {`
	`  40, 50, 78...`
	`}`

CONTROL OVER MESSAGES	Sequence

8 Design Knowledge

DM-1
System Architecture

Design Model	**Worksheet DM-1: System Architecture**
Architecture decision	**Format**
Subsystem structure	The architecture of our system is a variation of the MVC (Model-View-Controller) architecture, as denoted in the CommonKADS. See Figure 8
Control model	Centralized control unit which handles the input
Sub-System decomposition	-

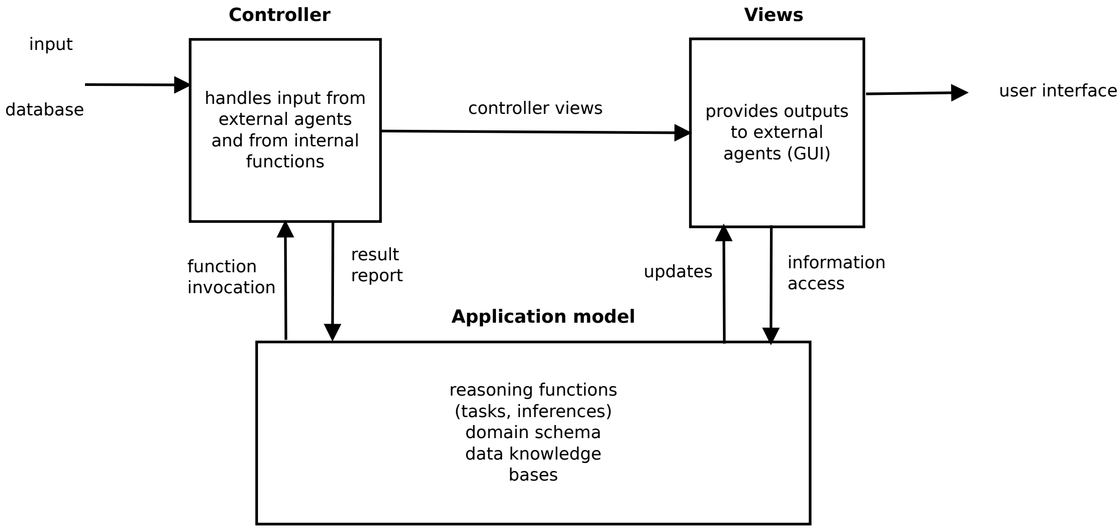

Figure 8: Model-View-Controller architecture

DM-2
Target implementation platform

Design Model	**Worksheet DM-2: Target Implementation Platform**
Software package	Python with Flask framework for the implementation of the expert system, HTML and Javascript (jQuery) for implementing the GUI, SQLite as a database
Potential hardware	Any hardware that supports python to host the webserver, and for the GUI any hardware that supports browsers capable of handling dynamic javascript.
Target hardware	Personal computer running any OS
Visualization library	HTML, CSS, jQuery
Language typing	Python is a strongly and dinamically typed language, Object Oriented, including multiple inheritance.
Knowledge representation	Procedural, the possibility of creating rulesets does not exixt natively.
Interaction protocols	HTTP requests
Control flow	Message passing through AJAX
CommonKADS support	The software does not support CommonKADS functionalities

9 Implementation

To implement the system in order to create a portable, scalable and user-friendly application it has been decided to develop a web-based service in Python with a basic GUI using HTML, CSS and Javascript (jQuery framework).

As the client/server communication is based on AJAX and standard HTTP requests, we exploited the functionalities provided by the Flask framework for pyhton. Additionally, we adopted an ulterior Python framework, SQLAlchemy, to manage the communicaton between the application and the database.

The basic setup of the final system is built as follows:

GUI

The user interface needs to be user friendly and easy to use, so to prevent the necessity of additional training for the Travel Agent that interacts directly with it. Built as a standard webpage, the GUI provides the TA with an extremely easy-to-use panel where to insert the information provided by the client such as: start date and end date of the planned trip, the presence of kids, the need for free time (time not filled in by the TA to be able to roam freely around the city), the customer's preference for quiet environments, the customer's willingness to walk long distances, and his preferences regarding four types of attractions (shopping, culture, gastronomy or night-life).

Once the TA has filled in the necessary information, can request the itinerary to the system by simply clicking the *Next* button. The interface will then show a list of the locations chosen for each day, which can be furtherly modified by the travel agent. By chosing to edit the itinerary, it will in fact be offered the opportunity to exclude some locations that do not satisfy the customer's desires. These modifications will then be forwarded to the system that will provide a new itinerary to show.

If the customer is satisfied with the proposed itinerary, then the TA can confirm the choice and end the procedure.

Server Side

The back-end consists in the set of rules and functions that constitute the server-side of the application. It has been neatly divided using a model-view-controller approach as illustrated in the previous diagrams, and provides functionalities of communication with the database, communication with the front-end, and a "knowledge engine" embedding the TA knowledge necessary to build the itinerary.

The entire server-side application has been build on the base of the inference model designed in the previous part of the document, and will be explained further in Section 9.2.

Database

The SQL based database contains the information relative to the client and the locations that will be handled by the "knowledge engine" to build the itinerary. The database has been build on the base of the domain model in Section 5.1, and has been filled with information coming directly from the *TripAdvisor* database. This specific way of building and filling the database favours the reusability of the whole application (for more information see Section 9.4)

9.1 Knowledge inside the software

The three types of rules designed for our system (constraint-rules, preference-rules, requirement-rules) have been implemented in the function *selectLocation()*. The rules have been implemented has a series of SQL queries to the database to carefully select the right locations, with the aid of a probability function, needed to implement the connector *prefers-with-probability* (see Section 5.3).

9.2 Role of the CommonKADS model set in the implementation

The implementation of the system was strongly based on the models prepared in the previous sections of this document, particularly on the Inference Model in section 4 and the Domain Model in section 5.1.
In this section it is illustrated briefly the correspondence between the functions implemented in *easyAround* and the roles and transfer functions of the inference model.

9.2.1 Roles

In the implementation of the software it is possible to recognize some data structures or classes that constitute the main entities of the system; each of these classes or data structures corresponds to an Inference Role. In this section it is presented a list of these data structures, complete of their role inside the Inference model and a brief explanation.

Request
Inference Role: request;
This is the class that contains the initial request coming from the customer, and represents the starting point of the whole system.

Skeletal Design
Inference Role: skeletal design;
This data structure is an aggregation of the days the itinerary is composed of, and needs to be later filled with appropriate locations. It constitutes the "skeleton" of a complete itinerary.

Preferences, Requirements and Constraints
Inference Role: preferences and requirements, constraints;
These three data structures contain the information needed to correctly classify the needs of the customer and build an itinerary accordingly.

Itinerary
Inference Role: proposal, itinerary;
This data structure constitutes the final itinerary, composed of all the days and

the locations. It can be accepted by the customer, or rejected with new directives, and in this case it undergoes modifications.

The roles in the inference not shown in this list have been implemented as much simpler data structures such as lists or arrays, and are: *violation, customer input, action list, action.*

9.2.2 Transfer Functions

The methods implemented in the software have, as well as classes and data structures, a unique correspondence to the Inference model. In this section is presented the list of the main methods implemented, complete of their role inside the Inference model and a brief explanation.

Specify(request)
Function: specify;
Creates the basic structure of the itinerary, the skeletal design.

Operationalize(request)
Function: operationalize;
Divides all the parameters in requirements, preferences and constraints.

Propose(requirements, preferences, constraints, skeletal design)
Function: propose;
Builds the itinerary based on the request of the client, using the knowledge rules.

Verify(proposal)
Function: verify;
Submits the itinerary to the client.

Critique(violation, itinerary)
Function: critique;
Edits the itinerary accordingly to the critique obtained from the client.

Select(location, itinerary)
Function: select;
For each violation, selects one single action to be performed and passes the control to modify.

Modify(constraints)
Function: modify;
Takes the selected action and commits the modification into the database.

9.3 Reflection on problems and results

The result of this project is a piece of software that represents a first completely functioning version of what could be a much broader system for travel planning. We consider ourselves satisfied with the result as the usability and the functionality of the software is coherent with the initial objectives, even though during the development process we encountered some difficulties.

The idea of developing a software in this domain was supported by the fact that knowledge is largely avaiable: information can be found on the Internet and even extracted from common sense notions (for example the fact that elderly people should not go climbing mountains). Thanks to this fact we were able to develop an efficient system without bothering too much our expert, who remained avaiable for important matters and maintained a very positive attitude towards us.
The avaiability of knowledge implies that data is easily is retrievable from any online database, complete with all the attributes needed to build an efficient knowledge engine. This made our job easier because we could avoid manually inserting all the data into the system.

One of the problems we faced was the broadness of the domain. Although we started by restricting it to a single city, the quantity of variables to be considered in selecting a solution remained huge. To be able to present a "minimum viable product" (MVP) able to provide basic functionalities, we adopted a spiral approach where we considered just the most important factors (extracted from a second interview with the expert) excluding complicated variables like monetary costs.

In the early stages of software development we realized that one of the objectives was too ambitious: we intended to offer an itinerary where the distances between the locations were calculated according to the customers' needs. However we soon realized that given the time and resources, the task was too complicated. The problem was analogous to the Travelling Salesman Problem (TSP) which unfortunately is NP-complete [1]. We decided to avoid partial or over-simplified solutions (randomly choosing a combination of locations leaving the eventual check to the TA), and simply move these features in a more advanced level of the spiral (See section 9.3.1).

During the implementation phase we also realized that the select-modify cycle inside the inference model template "configuration design" could have been optimized and that it did not reflect precisely the needs of our software. However we decided to remain coherent to the original template, in order to exploit the reliability of a standard model.

During the whole project one of the most time expensive task was to keep the documentation syncronized with the actual implementation. Despite the fact that it was a feasible task for two people, it would have been more difficult for a larger group.

We also realized the importance of calculating the risks before starting the actual development. Despite the careful designing of the models and the scheduling we constructed for the implementation, we did not consider risks related to human components of the project: the testing phase of the software was delayed of almost four days.

9.3.1 Spiral Approach

In this section is illustrated how the various models designed for this project can be expanded in the next phase of the spiral. By adding the functionalities relative to distance calculations, we can also demonstrate how easy it is to apply modifications to the already functioning product without disrupting its internal structure.

Inference Knowledge

Inference	Input	Output	Description
specify	request	skeletal design	the function builds the default skeletal design: the basic structure of a trip where each day is composed of: an heavy activity during the morning, a relaxing afternoon, evening and meal.

Domain schema

The domain schema can be found in Figure 9

Client
: The client who goes to the travel agency. He can express preferences for a quiet enviroment and express his willingness to travel long distances by foot (*dynamic*). The clients are categorized by their age to avoid unsuitable locations (ex: elderly people in a climbing location).

Location
: This model represents the point of interests that a customer could visit. The attribute *rating* describes the quality of this place, *intensive* describes if the place is not for quite people and *excludedCategory* specifies if a client category is not suitable for the location (ex: elderly people in a climbing location). The method *distance* takes two locations and returns the distance between them. It is useful in order to create the combination of locations to visit during a trip.

Rule types

Listing 7: The client is a quite person

```
client.quiet=true, client.needsFreeTime=true, client.active=1
REQUIRES
itinerary.day.timeslot.location, location.intensive=false;
itinerary.day.timeslot, timeslot.location=NULL;
```

$\sum_{i=1}^{n-1}$ i.distance(i+1) $< \delta$, $\forall$i $\in$ location;

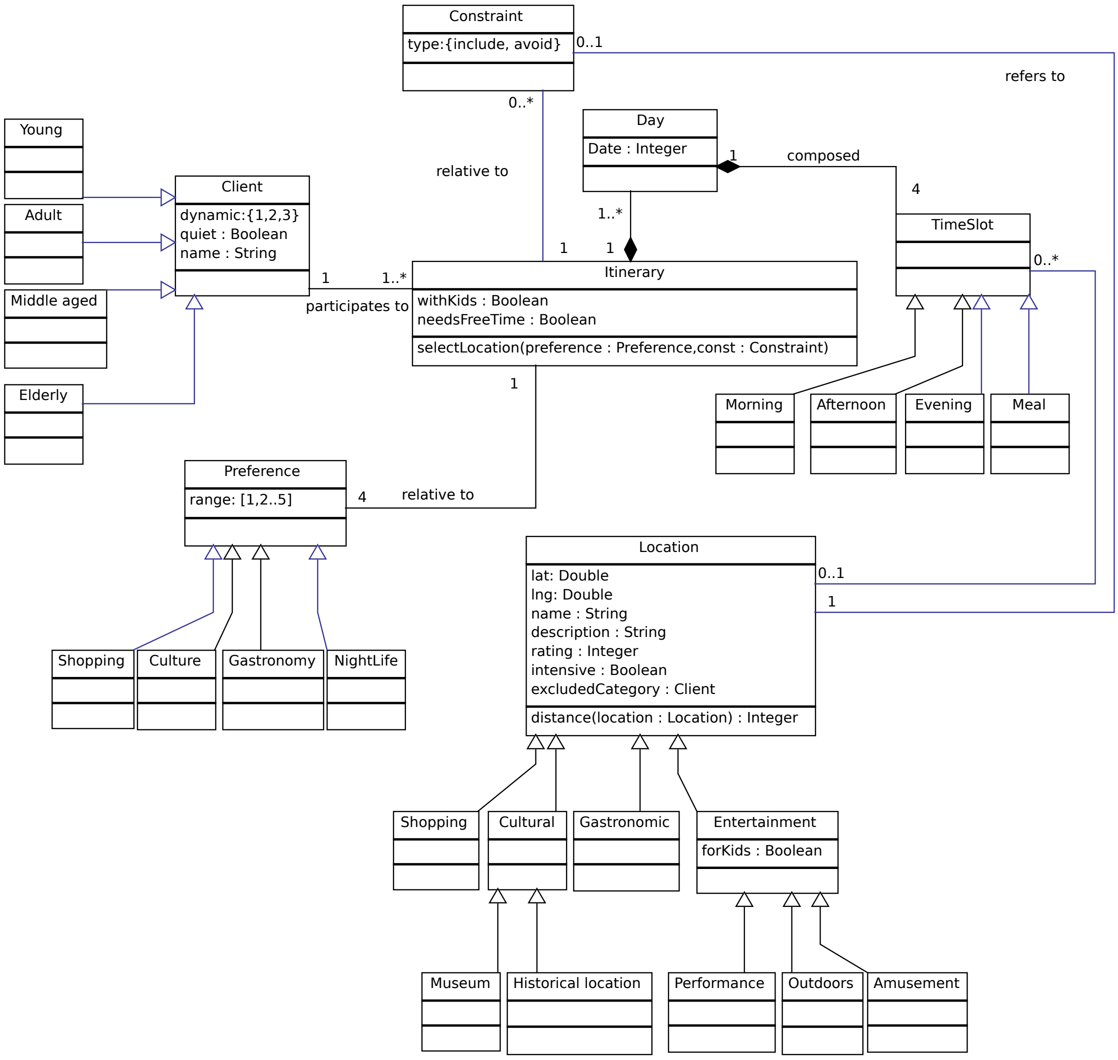

Figure 9: Domain schema

Communication Knowledge

CM-2a
Information exchange specification

Communication model	**Information Exchange Specification Worksheet CM-2**
TRANSACTION	*Insert data into the system*
AGENTS INVOLVED	Sender: Travel Agent Receiver: Automated System
INFORMATION ITEMS	**Role**: *core* object **Form**: data string **Medium**: HTTP request

MESSAGE SPECIFICATIONS	**Communication type**: Ask
	Content:
	`startDate: DATE`
	`endDate: DATE`
	`presenceOfKids: BOOLEAN`
	`needsFreeTime: BOOLEAN`
	`exclude: LIST`
	`include: LIST`
	`existingClient: INTEGER`
	`clientName: STRING`
	`clientDinamicity: INTEGER`
	`clientAge: INTEGER`
	`clientQuiet: BOOLEAN`
	`preferenceShopping: INTEGER`
	`preferenceCulture: INTEGER`
	`preferenceGastronomy: INTEGER`
	`preferenceNightLife: INTEGER`

CONTROL OVER MESSAGES	Sequence

9.4 Reusability

The application was build to be completely reusable, making sure that the choice of a restricted domain (the city of Rome) would not influence the performance of the system itself.

If, for example, the travel agency is interested in focusing the choice of points of interest on Amsterdam instead of Rome, the only thing that would be subjected to change is the data inside the database. The construction of the database itself and the whole application are absolutely not domain dependant, so they will automatically adapt the choice for the new itinerary on the available data, independently from its origin. The only action that needs to be performed in this case is just the one that recovers the data from the *TripAdvisor* database (or any other tourism-related database) accordingly to the need of the application.

This can be possible thanks to the generality of the rules in the *knowledge engine*, that base the choice of points of interest not on the locations themselves, but on the category to which they belong: a location dedicated to culture like a museum is easily recognizable both in Amsterdam and in Rome, even if the theme or purpose of that specific location are different. For example, *Van Gogh Museum* in Amsterdam and *Museo Civico di Zoologia* in Rome have nothing in common, but the application is able to suggest both to a person interested in culture.

Note however that the data inside the database *must* be complete of all the necessary information: if a location is not correctly classified (for example its coordinates are not known or its category is not specified) the application will not be able to perform at its full capacity.

Appendix A Design Knowledge

In preparation for the interview with the expert we listed a series of concepts to be clarified in order to better structure the application domain.

1. Target of the application: which kind of customer the application is more suitable for;

2. Subcategories of the target: is it possible to recognize different subcategories in the target that correspond to different needs for the creation of an itinerary;

3. Locations of interest: understand which categories of locations can be created and in which way they can be matched with the customer's preferences;

4. Composition of the itinerary: understand the basic structure of an itinerary, and whether it can be composed automatically.

The interviewing techniques applied were mainly two: problem solving (the interviewer poses himself as a customer and watches the expert "in action") and 20-Questions (the interviewer thinks about a destination for an hypothetical itinerary and the expert needs to guess which one it is). Relatively to the categorization of the locations, it has also been used the "Card sorting". The results of the interview were satisfying:

1. The target of the application are the "lonely travelers", people who prefer traveling on their own, at most with their family.

2. It has been concluded that the target can be divided in four age-based category, such as "Young" (18-30), "Adults" (30-40), "Middle aged" (40-60), "Elderly" (60+). Relating to these categories the aspects that change the most are: need for entertainment, need for quiet, free time available, amount of time spent walking.

3. The locations can be divided in four main categories such as "shopping", "gastronomy", "cultural" and "entertainment"; of these, "cultural" can be divided in "historical locations" (such as monuments) and "museums", and "entertainment" can be divided in "amusement" (such as clubs, pubs and discos), "performance" (such as cinemas, theatres, ...) and "outdoor" (such as amusement parks or gardens).

4. It has been concluded that the itinerary can be seen as an aggregation of days, which in turn have a basic fixed structure.

   Morning non intensive activity (monuments, gardens...);

   Afternoon intensive activity (museums, shopping...);

   Evening the intensity of the activity depends on personal preferences.

From the interview emerged an aspect not considered before, that is to say the presence of kids. The expert pointed out that in case a child is present, the itinerary is to be built as usual but having care of including children activities every once in a while. A constraint to be considered is the fact that the customer can request a location to be included or excluded from the itinerary.

References

[1] Christos H Papadimitriou, *The euclidean travelling salesman problem is np-complete*, Theoretical Computer Science **4** (1977), no. 3, 237–244.

[2] Guus Schreiber, *Knowledge engineering and management: the commonkads methodology*, the MIT Press, 2000.