INFSCI 2300
Human Information Processing

FINAL PROJECT REPORT

on

THE INTELLIGENT ASSISTANT
for
PURCHASING A LAPTOP COMPUTER

Project Term: Fall 2004

Mr.Kitipong Techapanichgul
Ms.Jaruwan Laptrakool
Mr.Warat Chesdavanijkul

INFSCI 2300
Human Information Processing

FINAL PROJECT REPORT

On

THE INTELLIGENT ASSISTANT for PURCHASING A LAPTOP COMPUTER

Project Term: Fall 2004

Group Members

Mr.Kitipong Techapanichgul (kit4@mail.sis.pitt.edu)

Ms.Jaruwan Laptrakool (jal66@pitt.edu)

Mr.Warat Chesdavanijkul (wac9@pitt.edu)

Purpose of the project

To create an intelligent assistant helping users choose a laptop computer that fits to their needs by analyzing from their individual characteristics, profession, habits and other related criterion.

Statement of problem

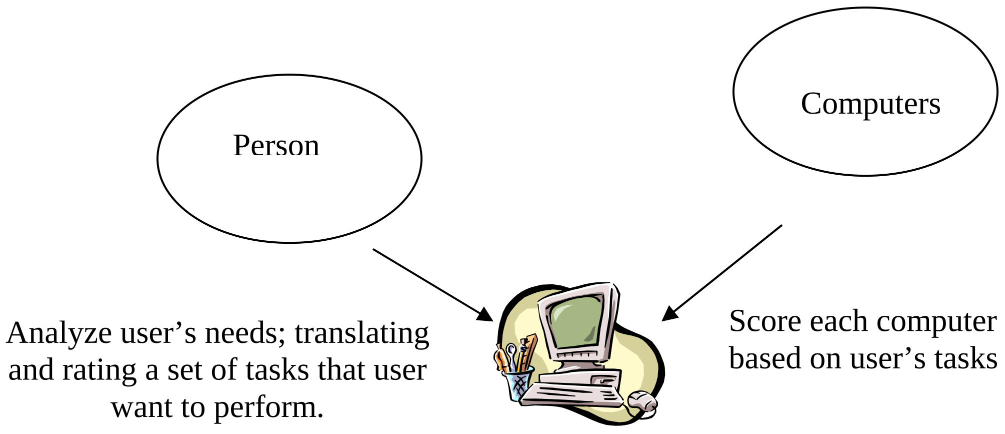
Purchasing a laptop computer that matches one's needs is such a hard decision for novice in IT fields since it requires more or less technical knowledge about the computer hardware. Also, it depends on other factors such as user's lifestyle, professions, characteristics, and etc., which these factors vary from person to person and culture to culture. Moreover, the computer manufacturers serve the market with various choices of laptop model that make people more confused about what is truly matched with their needs. Therefore, it must be great if there is an intelligent agent that can help matching the appropriate computers with the users' demands. To cover all kinds of computer products might lead to the great uncontrolled factors involve with the system, so that our system would have information on selected vendors only. On the other hand, we attempt to focus on analyzing people needs and responding them properly, which will be the core of the system. We hope this kind of project can, perhaps, be applied and implemented on other kind of product in the future.

Individual Responsibilities

Name	Responsible for
Warat Chesdavanijkul	• Design overall system • Gather criterions used for person and list them in a note • Design and Develop rule used in analyzing computer part • Create KB in user part • Refine and verify the system for integrity • Edit final report
Jaruwan Laptrakool	• Design overall system • Gather computer list • Develop preliminary system • Create KB in computer part • Design and Develop rule used in Analyzing Computer part • Edit final report
Kittipong Techapanichgul (Kit)	• Design overall system • Report • Record progress of each phase • Design and Develop rule used in intelligent assistant part • Refine and verify the system for integrity • Test and debug the error

Overview of the system

The scope of the project is to provide a list of computers, along with score associated, in which higher score indicates better match with the subject's need. There are 2 main parts in the system, involving translating the user's needs to each specification of the computer; another is to score the laptop that best match with the output from part 1. In short, the first part does analyze the user's requirement while the second part does the matching to the actual laptop.


Person
Computers
Analyze user's needs; translating and rating a set of tasks that user want to perform.
Score each computer based on user's tasks

An Intelligent Assistant

Upon analyze, the subject will be asked what we called "Intelligent questions". It works by making relevant assumptions according to the subject's answer to the question. In addition, there is no "set of same old questions" here - user will be asked different questions based on their previous answer and the existing assumptions. Analyzing all results and assumptions, the program will turn all the user's input into a set of tasks that user wants to perform.

After getting a set of tasks user wants to perform with his/her new laptop, the second part now come into plays – again, it translate the set of tasks into an actual laptop's specification, then it score each laptop based on how relevant the task affects the specification. Although this part does not seem to be intelligent as that in the first part, it is still "smart", that is, it is not just a plain calculation after all. By this, the program will do the fine tune on each task by utilizing the program's assumption[1] to tune up or down the needs for every specific task, resulting in more reliable overall result.

The output from the system will be the score of every computer in database. The higher score indicates better suit with the user's need. Those score will again be filtered by price range, and display the filtered result to the user.

[1] Note: the program's assumptions are derived from the actual data that we collect from people who is about to buy computer. Before we start doing CLIPS, we built a model in Microsoft Excel that scores the computer using the input similar to what we do in part 1.

The Functionality

The first part receive the user's input using the rule '**start_program**' to collect basic information and the rule '**ask_fact**' to pick the intelligent questions from our knowledge base to ask for additional information that are used as factors to choose the best matched. At this part, the result in our system is a set of tasks that user will want to perform with his/her new laptop. These tasks are inferred from the answer given by user.

The second part, after the program gathers got a set of tasks, the rule '**convert_inference_to_value**' will be fired in order to turn the set of task into a numerical value. To make the system more effective, we will not consider only the task getting from user's answer, but we also tune the value using the basic information of user and possibility which system get from the model to perform a particular task. The function use to tune the value is '**modify_inference_value**'. The modified value will make the value more realistic especially when the user's answer seems exceptional and concern with some bias. Whatsoever, all information still base on the user's need.

To this point, the program will collect all the needed computer's specification that should be best matched with the user's need. Next step is to filter out all the laptop that is not in the price's range. Rule '**filter_out_less_laptop**' and '**filter_out_more_laptop**' will handle this job. The result of the program will be as following:

If the budget is less than $2000, the result will be any laptop which has price ranged from the price that user input at the first place minus by $300, to $100 more than the user's input data.

If the budget is more than $2000, the result will be any laptop in database which has price more than 1800. The reason behind this is because most laptops in real world are priced between $1000 and $2000, so supposed user put the budget more than 2000, then there will be a very few result left. Besides, this technique also helps avoiding if the rich who have budget more than $5000 using the system so that the system will show out some result without having any result shown.

Next thing the rule ‘**score_all_laptop**’ will be fired to score all laptops existing in the CLIPS’s fact.

Finally, rule ‘**printout_all_laptop**’ will be fired, this print out all scores associated with the laptops those are in the price range.

This way, user will just have to get best matched laptop by looking at the system which has the highest score on the list, and then if he/she doesn’t like a specific brand or anything then he might consider another one which has fewer score.

The Engine Model

Person slots / Age Range	Children M	Children G	College M	College G	Adult M	Adult G
Game Playing	5	2	4	2	3	1
Self Study	1	3	2	4	5	5
Word Processing and Spreadsheet	1	1	3	4	4	5
Email	1	3	3	4	4	5
Chatting	1	2	4	5	2	3
Surfing Net	3	3	5	5	5	5
Homepage	4	3	5	4	1	1
Watching a Movie	3	3	4	4	4	4
Listening to Music	2	3	5	4	4	4
File Sharing	1	1	4	3	5	4
File Transfer	1	1	4	4	5	4
Backup	1	1	3	2	5	4
Programming	2	1	N/A	N/A	N/A	N/A
Calculation (Math lab)	N/A	N/A	N/A	N/A	N/A	N/A
Presentation	1	1	3	3	5	5
Graphic	1	1	N/A	N/A	N/A	N/A
Sound Engineer	1	1	N/A	N/A	N/A	N/A
Online Meeting	1	2	4	5	3	3
Teleconference	1	1	2	1	5	4
Work consistency (business can't fail)	1	1	2	2	5	5
Avg. Distance	N/A	N/A	N/A	N/A	N/A	N/A
Travel Frequency	N/A	N/A	N/A	N/A	N/A	N/A
Power Availability	N/A	N/A	N/A	N/A	N/A	N/A
Toughness (working environment)	N/A	N/A	N/A	N/A	N/A	N/A
Mobility	N/A	N/A	N/A	N/A	N/A	N/A
No. of family	N/A	N/A	N/A	N/A	N/A	N/A
No. of own Computer	N/A	N/A	N/A	N/A	N/A	N/A
No. of Cell phone	2	3	3	4	5	5
No. of Camera	1	1	3	2	4	4
No. of Camcorder	1	1	2	1	3	3

Table 1 : common values of needs divided by range of ages and sex

This table is created base on the general activities and needs of people in each range of ages and sex. In this case we roughly divided into six groups which are

- Children/male (8-15 year-old)
- Children/female
- College/male (16-22 year-old)
- College/female
- Adult/male (23 year-old and above)
- Adult/female

There are 6 possible values for indicating the degree of needs that are 1 to 5 and N/A. The values that equal to 1 mean that the activities are not interesting for the particular group of that sex and ages. In contrast, the values that equal to 5 mean that the activities are very common for the particular group of that sex and ages. Some slots cannot give any values because of no relationship between slots and range of ages and sex. After adjusting or tuning the values of user's requirement with the common values (shown in the table no.1), we will get the appropriate values for the user requirements.

Now the system will figure out whether a computer component should be used as a major or minor indicator for choosing a suitable set of computers by analyzing from the user's activities. Before that we have to understand that each activity will affect the performance of computer components in different ways and levels. Specifically, each activity requires different specification of computer component, so that we create a table of relationship between activities and computer components to show the degree of relevance for every activity related to that computer components. For instance, a graphic job must require a large size of monitor and huge capacity of memory (You can see in the table that a graphic job needs high performance of a CPU, RAM, Big Screen Size and 3D Accelerator).

User activities/ Computer Components	Brand Durability	CPU	RAM	LAN	Wireless LAN	HDD	Big Screen Size	Small Screen Size	3D Accelerator	Sound
Game playing	0	5	5	4	1	4	4	0	5	4
Self-Study	0	3	2	1	1	3	2	0	2	3
Word processing	0	2	3	1	1	2	3	0	2	1
E-Mail	0	1	1	2	1	2	1	0	1	1
Chat	0	2	2	2	1	1	1	0	1	1
Surfing net	0	4	4	2	2	3	3	0	4	3
Homepage	0	4	3	3	3	2	3	0	3	3
Watching movie	0	4	3	3	1	5	5	0	5	5
Listening music	0	2	3	3	2	5	1	0	1	5
File sharing	0	2	5	5	1	5	1	0	1	1
File Transfer	0	2	2	5	3	2	1	0	1	1
Archive	0	1	2	3	1	5	1	0	1	1
Programming	0	5	5	1	1	3	3	0	1	1
Calculation	0	5	5	1	1	3	2	0	1	1
Presentation	0	1	2	2	2	2	5	0	4	4
Graphic	0	5	5	1	1	4	5	0	5	1
Sound engineer	0	5	5	1	1	5	1	0	1	5
Online meeting	0	0	3	3	3	2	3	0	3	3
Teleconference	0	4	4	5	4	1	4	0	4	4
Work consistency	5	5	4	0	0	2	0	0	0	0
Avg. Distance	5	0	0	0	5	0	1	0	0	0
Travel frequency	5	0	0	0	5	0	1	5	0	0
Power availability	1	0	0	0	0	0	1	5	1	0
Toughness	5	0	0	0	0	0	0	0	0	0
Mobility	3	0	0	0	5	0	1	5	0	0
Num of Family member	4	0	0	3	3	5	5	0	4	4
Num of PC Own	2	0	0	5	5	3	3	0	4	0
No. of Cell phone	0	0	0	0	0	0	0	0	0	0
No. of Camera	0	0	0	0	0	0	0	0	0	0
No. of Camcorder	0	0	0	0	0	0	0	0	0	0

Table 2.1 : The values of relevance between activities and computer components

Like the table no.1, there are 6 possible values for indicating the degree of needs that are 1 to 5 and 0. The values that equal to 1 mean that the activity little affect to the computer components while the values that equal to 5 mean that the activity directly affect to the computer components. However, the values that equal to 0 mean that the activity and the computer components are not relevant.

User activities/ Computer Components	**DVD Reader**	**DVD Writer**	**Weight**	**Batt**	**Fire wire**	**USB**	**Blue tooth**	**Infrared**
Game playing	5	1	0	0	1	3	0	0
Self-Study	3	1	0	0	0	1	0	0
Word processing	1	1	0	0	0	1	0	0
E-Mail	1	1	0	0	0	1	0	0
Chat	1	1	0	0	0	0	0	0
Surfing net	1	1	0	0	0	3	0	0
Homepage	1	1	0	0	0	0	0	0
Watching movie	5	1	0	0	2	2	0	0
Listening music	4	1	0	0	2	2	0	0
File sharing	4	1	0	0	0	3	0	0
File Transfer	1	1	0	0	0	3	0	0
Archive	1	5	0	0	0	3	0	0
Programming	1	1	0	0	0	0	0	0
Calculation	1	1	0	0	0	0	0	0
Presentation	2	1	0	0	0	2	0	0
Graphic	1	1	0	0	0	0	0	0
Sound engineer	3	1	0	0	3	3	0	0
Online meeting	1	1	0	0	0	3	0	0
Teleconference	1	1	0	0	0	4	0	0
Work consistency	0	0	0	0	0	0	0	0
Avg. Distance	0	0	5	5	0	0	0	0
Travel frequency	0	0	5	3	0	0	0	0
Power availability	0	0	0	5	0	0	0	0
Toughness	0	0	0	0	0	0	0	0
Mobility	0	0	5	0	0	0	0	0
Num of Family Member	5	4	0	0	2	3	0	0
Num of PC Own	3	3	0	0	0	3	0	0
No. of Cell phone	0	0	0	0	0	0	5	5
No. of Camera	0	3	0	0	3	5	0	0
No. of Camcorder	0	4	0	0	5	3	0	0

Table 2.2 : The values of relevance between activities and computer components (continue)

At this point, the system can calculated the weight of each computer component. depending on its importance to each common user's activity. However, because user spend time doing in some particular activities more than other activities (not all activities have the same importance), the system has to re-calculate the table of relevance between activities and computer components according to individual style of the user. Then, the system will pick only the highest value in each computer component regardless what activity contributes to because , from user perspective, this highest value in each component represents the highest expectation of that component's performance. For example, imagine that the system the table of relevance between activities and computer components according to individual style of the user like in the table below (already re-calculate):

USER SCORE	CPU	RAM	3D Accelerator
Game playing	2	2	2
Self-Study	1.8	1.2	1.2
Word processing	2	3	2
E-Mail	1	1	1
Chat	2	2	1
Surfing net	4	4	2
Homepage	0.8	0.6	0.6
Watching movie	1.6	1.2	1.2
Listening music	1.6	2.4	0.8
File sharing	0.8	2	0.4
File Transfer	0.8	0.8	0.4
Archive	0.2	0.4	0.2
Programming	4	4	0.8
Calculation	1	1	0.2
Presentation	0.2	0.4	0.4
Graphic	4	4	2.4
Sound engineer	1	1	0.2
Online meeting	0	1.2	1.2
Teleconference	0.8	0.8	0.6
Work consistency	5	4	0
Avg. Distance	0	0	0
Travel frequency	0	0	0
Power availability	0	0	0.4
Toughness	0	0	0
Mobility	0	0	0
Num of Family member	0	0	1.6
Num of PC Own	0	0	1.6
No. of Cell phone	0	0	0
No. of Camera	0	0	0
No. of Camcorder	0	0	0
MAX	5	4	2.4

The system will pick the weight “5” for CPU because the user emphasizes on “work consistency” rather any other activity and work consistency mainly results from CPU Performance. Consider another example, in a 3D accelerator, the user pay attention to a graphic job so that the system choose the weight “2.4” for RAM because almost all kinds of graphic tasks need high performance of a 3D accelerator and the user works on graphic tasks rather than any other activities. All the weights of each computer components will be used to give a total score for each computer specification in a prepared list.

The table shown in the next page is one of the examples of scoring five computer in a list according to the degree of user’s needs in all aspects containing in the system. The computer with the highest score becomes the best match for the user’s requirements. Lastly, the system provides an appropriate list of reasonable computer which fits to unique needs and his budget.

User Specific Requirements

User Activities	Degree	User Activities	Degree
Game playing	2	Graphic	4
Self-Study	3	Sound engineer	1
Word processing	5	Online meeting	2
E-Mail	5	Teleconference	1
Chat	5	Work consistency	5
Surfing net	5	Avg. Distance	3
Homepage	1	Travel frequency	3
Watching movie	2	Power availability	2
Listening music	4	Toughness	4
File sharing	2	Mobility	5
File Transfer	2	Num of Family Member	2
Archive	1	Num of PC Own	2
Programming	4	No. of Cell phone	1
Calculation	1	No. of Camera	2
Presentation	1	No. of Camcorder	0

Computer Scores according to the user needs

COM SPEC	CPU	RAM	LAN	HDD	Big Screen Size	Accelerator	
unit	(Mhz)	(MB)	score	(GB)	score	score	
R51-1	**C-1300**	**256**	**100Mb**	**30**	**15"**	**Intel**	**Total score**
Specification	1040	256	80	30	75	30	
User score	2.6	1	1.92	1.5	3	0.72	30.39
R51-2	**P715/1.5**	**256**	**100Mb**	**30**	**14.1"**	**Intel**	**Total score**
Specification	1500	256	80	30	65	30	
User score	3.75	1	1.92	1.5	2.6	0.72	38.29
T42-1	**P715/1.5**	**256**	**100Mb**	**30**	**14.1"**	**ATI 7500**	**Total score**
Specification	1500	256	80	30	65	50	
User score	3.75	1	1.92	1.5	2.6	1.2	40.82
T42-3	**P735/1.7**	**512**	**100Mb**	**40**	**15"**	**ATI 7500**	**Total score**
Specification	1700	512	80	40	75	50	
User score	4.25	2	1.92	2	3	1.2	40.77
X40-1	**PLV/1.2**	**256**	**100Mb**	**40**	**12.1"**	**Intel**	**Total score**
Specification	1200	256	80	30	20	30	
User score	3	1	1.92	1.5	0.8	0.72	37.14
X40-2	**PLV/1.2**	**256**	**100Mb**	**40**	**12.1"**	**Intel**	**Total score**
Specification	1200	256	80	30	20	30	
User score	3	1	1.92	1.5	0.8	0.72	41.14

Lists of objects in domain, and their properties

The followings are the objects related to customer analyzing part:

Person's object and properties:

The object 'person' is used to keep the general person's information which is name, age, and sex. Also, it will keep the list of question that the system asked and what the user's answers, with the inferred list we getting from those answers. The template is defined as:

```
(deftemplate person "person and his information"
        (slot name)
        (slot age)
        (slot sex)
        (multislot questionlist); list of questions that system already asked
        (multislot answerlist)  ; list of facts that directly get from the person
        (multislot inferlist)   ; list of facts that infer from the answer of the person
)
```

System's Question object and properties:

The object 'system_question' is used to keep the question the system ask user to collect more information.

```
(deftemplate system_question "all possible facts collected from a person by the system"
        (slot question_name)  ;questions that are available in the system
        (slot question)       ;question category
        (multislot answerlist) ;Possible answers of the question
        (multislot require (default nil)) ;required questions before this question to ask the
user with related questions
)
```

Inference object and properties:

There are three objects in related to the inference system. The object **'new_fact'** is used to keep the information infer from available facts. The object **'inference_value'** is used to keep the inferred value, which obtain from the conversion of facts to numerical value. The object 'attribute' is used to keep the values for tuning and the default value in our system. The '*attribute*' is used in rules **'tune_attributes_down'**, **'tune_attributes_down'**, and **'no_tune_attributes'**

```
(deftemplate new_fact "all facts infer from the available facts in the system"
        (multislot require_fact)    ; conditional facts
        (multislot new_fact)        ; new deductible facts
        (slot suggestion)           ; new deductible facts that computer have learnt from
        (slot reason)               ; reason why computer learn that
        (multislot infer_answer)  ; facts that lead other related questions to ask user
about more because the system still want information to deduce
)

(deftemplate inference_value "all facts convert to numeric value"
        (slot inference_name)
        (slot inference_value)
        (slot attribute_name)
)

(deftemplate attribute "attribute for all facts from user"
        (slot name)
        (slot value(default 0))
        (slot type)
)
```

Computer's object and properties:

The object **'spec_template'** holds all specification of every component in computer for every single computer. The slot "status" keeps track of how this system is evaluated so far.

```
(deftemplate spec_template
        (slot status)
        (slot model_text (type STRING))
        (slot CPU_text (type STRING))
        (slot RAM_text (type STRING))
        (slot LAN_text (type STRING))
        (slot wireless_LAN_text (type STRING))
        (slot HDD_text (type STRING))
        (slot screen_size_text (type STRING))
        (slot 3d_accelerator_text (type STRING))
        (slot DVD_text (type STRING))
        (slot weight_text (type STRING))
        (slot batt_text (type STRING))
        (slot firewire_text (type STRING))
        (slot USB_text (type STRING))
        (slot bluetooth_text (type STRING))
        (slot infrared_text (type STRING))
        (slot brand_dura (type FLOAT))
        (slot CPU (type FLOAT))
        (slot RAM (type FLOAT))
        (slot LAN (type FLOAT))
        (slot wireless_LAN (type FLOAT))
        (slot HDD (type FLOAT))
        (slot big_screen_size (type FLOAT))
        (slot small_screen_size (type FLOAT))
        (slot 3d_accelerator (type FLOAT))
        (slot sound (type FLOAT))
        (slot DVD_reader (type FLOAT))
        (slot DVD_writer (type FLOAT))
        (slot weight (type FLOAT))
        (slot batt (type FLOAT))
        (slot firewire (type FLOAT))
        (slot USB (type FLOAT))
        (slot bluetooth (type FLOAT))
        (slot infrared (type FLOAT))
        (slot price (type INTEGER))
        (slot score (type FLOAT) (default 0.0))
)
```

The object **'p_template'** holds the output of every specific task that the program assumes user to do, this value will be evaluated and intelligent map all these slots into with **'p_output'**.

```
(deftemplate p_template
        (slot category)
        (slot game_playing (type INTEGER))
        (slot self_study (type INTEGER))
        (slot word_processing (type INTEGER))
        (slot email (type INTEGER))
        (slot chat (type INTEGER))
        (slot surfing_net (type INTEGER))
        (slot homepage (type INTEGER))
        (slot watching_movie (type INTEGER))
        (slot listening_music (type INTEGER))
        (slot file_sharing (type INTEGER))
        (slot file_transfer (type INTEGER))
        (slot archive (type INTEGER))
        (slot programming (type INTEGER))
        (slot calculation (type INTEGER))
        (slot presentation (type INTEGER))
        (slot graphic (type INTEGER))
        (slot sound_engineer (type INTEGER))
        (slot online_meeting (type INTEGER))
        (slot teleconference (type INTEGER))
        (slot work_consistency (type INTEGER))
        (slot avg_distance (type INTEGER))
        (slot travel_frequency (type INTEGER))
        (slot power_availability (type INTEGER))
        (slot toughness (type INTEGER))
        (slot mobility (type INTEGER))
        (slot num_of_fam (type INTEGER))
        (slot num_of_pc_own (type INTEGER))
        (slot no_of_cellphone (type INTEGER))
        (slot no_of_camera (type INTEGER))
        (slot no_of_camcorder (type INTEGER))
)
```

The template **'p_output'** is the final computer specification result that would best match to the user's input. This value will then be used to score every single laptop that has an appropriate price in the list.

```
(deftemplate p_output
        (slot status)
        (slot brand_dura (type FLOAT))
        (slot CPU (type FLOAT))
        (slot RAM (type FLOAT))
        (slot LAN (type FLOAT))
```

```
            (slot wireless_LAN (type FLOAT))
            (slot HDD (type FLOAT))
            (slot big_screen_size (type FLOAT))
            (slot small_screen_size (type FLOAT))
            (slot 3d_accelerator (type FLOAT))
            (slot sound (type FLOAT))
            (slot DVD_reader (type FLOAT))
            (slot DVD_writer (type FLOAT))
            (slot weight (type FLOAT))
            (slot batt (type FLOAT))
            (slot firewire (type FLOAT))
            (slot USB (type FLOAT))
            (slot bluetooth (type FLOAT))
            (slot infrared (type FLOAT))
            (slot budget (type INTEGER))
)
```

Lists of Principal Rules

This is the first rule, it welcomes the user with greeting message and ask user with basic geneal question

```
(defrule start_program "start program"
        =>
        (printout t "=========================================" crlf)
        (printout t "          The Laptop Purchasing Assistant System          " crlf)
        (printout t "=========================================" crlf)
        (printout t " Welcome to The Laptop Purchasing Assistant System." crlf "
                I'm your Assistant and now ready to help you, sir!!!." crlf)
        (format t "- What is your name, sir? : ")
          (bind ?name (read))
        (format t "- What is your gender? [m/f] :
          (bind ?sex (read))
        (format t "- How old are you now? : ")
          (bind ?age (read))
        (assert (person(name ?name)(sex ?sex)(age ?age)))
)
```

This is the rule to ask a set of selected questions to user in ordered to gathering more information.

```
(defrule ask_fact "ask the most important questions about all relevant aspects in depth"

 ?p  <- (person (name ?name)(questionlist $?questionlist)(answerlist $?panswerlist))
 ?s  <- (system_question(question_name ?question_name)(answerlist $?answerlist)
        (question ?question)(require $?require))
        (not(test(member$ ?question_name $?questionlist)))
        (or (test(member$ $?require $?panswerlist))(test(member$ nil $?require)))
```

```
            =>
            (printout t "??? - " ?question crlf)
            (printout t  $?answerlist crlf)
            (format t "please select one: ")
        (bind ?newfact (read))
            (modify ?p (questionlist ?question_name $?questionlist)(answerlist ?newfact
            $?panswerlist))
            (retract ?s)
)
```

With the facts getting from user’s answer, we can infer the other set of fact using the following rule.

```
(defrule infer_from_fact "infer some new facts from available facts"

 ?p  <- (person (name ?name)(answerlist $?panswerlist)(inferlist $?inferlist)
            (questionlist $?questionlist))
            (new_fact(require_fact $?require_fact)(suggestion ?s)(reason ?r)(new_fact
            $?new_fact)(infer_answer $?infer_answer))
            (test(member$ $?require_fact $?panswerlist))
            (not(test(member$ $?new_fact $?inferlist)))
            =>
            (modify ?p (inferlist $?new_fact $?inferlist)(answerlist $?infer_answer
            $?panswerlist))
            (printout t " *ASSISTANT HAVE LERNT: " ?s crlf)
            (printout t " *REASON FROM THE FACTS: " ?r crlf)
)
```

After we got a set of facts which is the need of user and what we infer from those needs, we will convert in to the numerical value by the following rule.

```
(defrule convert_inference_to_value "convert inferable facts to be value format"

        (person (name ?name)(inferlist $?inferlist))
        (inference_value (inference_name ?infer_name)(inference_value
        ?value)(attribute_name ?attribute_name))
        (test(member$ ?infer_name $?inferlist))
        (not(attribute(name ?attribute_name)(type user)))
        =>
        (assert(attribute(name ?attribute_name)(value ?value)(type "user")))
        (printout t " #System: " ?attribute_name " is already converted to be numeric
        value used in the system =" ?value crlf)
)
```

Now there are a set of rules to assert the default value of each activities for each groups of people. In our system, there are 6 groups of people according to the gender and age that are the male and female children, the male and female teenage/college student, and the male and female adult. The rule for each group of people is simply the same, except for the values to be asserted are different. The rule shown below is the rule for 'male adult'

```
(defrule mainrule_adult_m "male adult default value"
        (person (age ?age)(sex ?sex))
        (test(> ?age 22))
        (test(< ?age 120))
        (test (eq ?sex m))
        =>
        (assert(attribute(name game_playing)(value 3)(type "default")))
        (assert(attribute(name self_study)(value 5)(type "default")))
        (assert(attribute(name word_processing)(value 4)(type "default")))
        (assert(attribute(name email)(value 4)(type "default")))
        (assert(attribute(name chat)(value 2)(type "default")))
        (assert(attribute(name surfing_net)(value 5)(type "default")))
        (assert(attribute(name homepage)(value 1)(type "default")))
        (assert(attribute(name watching_movie)(value 4)(type "default")))
        (assert(attribute(name listening_music)(value 4)(type "default")))
        (assert(attribute(name file_sharing)(value 5)(type "default")))
        (assert(attribute(name file_transfer)(value 5)(type "default")))
        (assert(attribute(name archive)(value 5)(type "default")))
        (assert(attribute(name programming)(value 0)(type "default")))
        (assert(attribute(name calculation)(value 0)(type "default")))
        (assert(attribute(name presentation)(value 5)(type "default")))
        (assert(attribute(name graphic)(value 0)(type "default")))
        (assert(attribute(name sound_engineer)(value 0)(type "default")))
        (assert(attribute(name online_meeting)(value 3)(type "default")))
        (assert(attribute(name teleconference)(value 5)(type "default")))
        (assert(attribute(name work_consistency)(value 5)(type "default")))
        (assert(attribute(name avg_distance)(value 0)(type "default")))
        (assert(attribute(name travel_frequency)(value 0)(type "default")))
        (assert(attribute(name power_availability)(value 0)(type "default")))
        (assert(attribute(name toughness)(value 0)(type "default")))
        (assert(attribute(name mobility)(value 0)(type "default")))
        (assert(attribute(name num_of_fam)(value 0)(type "default")))
        (assert(attribute(name num_of_pc_own)(value 0)(type "default")))
        (assert(attribute(name no_of_cellphone)(value 5)(type "default")))
        (assert(attribute(name no_of_camera)(value 4)(type "default")))
        (assert(attribute(name no_of_camcorder)(value 3)(type "default")))
)
```

Next step, we need to tune the fact getting from user and the default value to make the value we have to use in matching with computer more realistic and sensible especially when the distorted exceptional occur. The tune rules have to check if the values need to be added up or cut down. It is possible that the values are already reasonable and no adjustment needed.

```
(defrule tune_attributes_up "tune attributes to the proper level according to the standard value"

        (attribute (name ?name)(value ?dvalue)(type "default"))
        (attribute (name ?name)(value ?uvalue)(type "user"))
        (test (> (- ?uvalue ?dvalue) 2))
        (not(test (eq ?dvalue 0)))
        (not(attribute (name ?name)(value ?avalue)(type "tune")))
        =>
        (assert (attribute (name ?name)(value (- ?uvalue 1))(type "tune")))
        (printout t " + tune down with default value " ?name crlf)
)

 (defrule tune_attributes_down "tune attributes to the proper level according to the standard value"

        (attribute (name ?name)(value ?dvalue)(type "default"))
        (attribute (name ?name)(value ?uvalue)(type "user"))
        (test (> (- ?dvalue ?uvalue) 2))
        (not(test (eq ?dvalue 0)))
        (not(attribute (name ?name)(value ?avalue)(type "tune")))
        =>
        (assert(attribute(name ?name)(value (+ ?uvalue 1))(type "tune")))
        (printout t " - tune up with default value " ?name crlf)
)
```

```
 (defrule no_tune_attributes "no tuning because the value is in proper range"

            (attribute (name ?name)(value ?dvalue)(type "default"))
            (attribute (name ?name)(value ?uvalue)(type "user"))
            (and(not(test (> (- ?uvalue ?dvalue) 2)))(not(test (> (- ?dvalue ?uvalue) 2))))
            (not(test (eq ?dvalue 0)))
            (not(attribute (name ?name)(value ?avalue)(type "tune")))
            =>
            (assert ( attribute (name ?name)(value ?uvalue)(type "tune")))
            (printout t " <> no tuning with default value " ?name crlf)
)
```

To prepare for the matching with computer, the system needs to know the score for each component of the computer. There is a rule to calculate the score for each component best on the tuned values.

```
(defrule calculate_score "calculate activity-component score"

?z <-   (computer_component (component_name ?component)(defined_value
            ?dvalue)(score ?score)(attribute_name ?name))
            (attribute (name ?name)(value ?avalue)(type "tune"))
            (test(eq ?score 0.0))
            (test(neq ?dvalue 0))
            =>
            (modify ?z (score (/(* ?dvalue ?avalue)5)))
            (printout t " % "?component " get score in " ?name crlf)
)
```

Now we have the value on **'p_output'**, what we do next is use this value to evaluate every laptop on the list.

First we need to filter out the computer that is not in an appropriate price range.

```
(defrule filter_out_less_laptop ""
        (p_output (status final_data) (budget ?b))
        ?com <- (spec_template (status unmatched) (price ?p))
        (and (test (< ?p (+ ?b 100))) (test (> ?p (- ?b 400))))
        (test (< ?p 2001))
=>
        (modify ?com(status filtered))
)

(defrule filter_out_more_laptop ""
        (p_output (status final_data) (budget ?b))
        ?com <- (spec_template (status unmatched) (price ?p))
        (and (test (< ?p (+ ?b 100))) (test (> ?p 1800)))
        (test (> ?p 2000))
=>
        (modify ?com(status filtered))
)
```

Then, these rules and functions are used to score the computer.

```
(deffunction add_multivalue (?a ?b ?c ?d ?e ?f ?g ?h ?i ?j ?k ?l ?m ?n ?o ?p ?q ?r)
        (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ (+ ?a ?b) ?c) ?d) ?e) ?f) ?g)
?h) ?i) ?j) ?k) ?l) ?m) ?n) ?o) ?p) ?q) ?r)
)

(deffunction cal_score (?a ?b ?c)
  (* (/ ?a ?b) ?c)
)


(defrule score_all_laptop ""
	?com <- (spec_template (status filtered) (brand_dura ?c_br) (CPU ?c_cp) (RAM
?c_ra) (LAN ?c_la) (wireless_LAN ?c_wi) (HDD ?c_hd) (big_screen_size ?c_bs)
(small_screen_size ?c_ss) (3d_accelerator ?c_3d) (sound ?c_so) (DVD_reader ?c_dr)
(DVD_writer ?c_dw) (weight ?c_we) (batt ?c_ba) (firewire ?c_fi) (USB ?c_us)
(bluetooth ?c_bl) (infrared ?c_in))
	(spec_template (status max_score) (brand_dura ?m_br) (CPU ?m_cp) (RAM
?m_ra) (LAN ?m_la) (wireless_LAN ?m_wi) (HDD ?m_hd) (big_screen_size ?m_bs)
(small_screen_size ?m_ss) (3d_accelerator ?m_3d) (sound ?m_so) (DVD_reader ?m_dr)
(DVD_writer ?m_dw) (weight ?m_we) (batt ?m_ba) (firewire ?m_fi) (USB ?m_us)
(bluetooth ?m_bl) (infrared ?m_in))
	(p_output (status final_data) (brand_dura ?u_br) (CPU ?u_cp) (RAM ?u_ra)
(LAN ?u_la) (wireless_LAN ?u_wi) (HDD ?u_hd) (big_screen_size ?u_bs)
(small_screen_size ?u_ss) (3d_accelerator ?u_3d) (sound ?u_so) (DVD_reader ?u_dr)
(DVD_writer ?u_dw) (weight ?u_we) (batt ?u_ba) (firewire ?u_fi) (USB ?u_us)
(bluetooth ?u_bl) (infrared ?u_in))
=>
	(modify ?com(status match) (score (add_multivalue (cal_score ?c_br ?m_br
?u_br) (cal_score ?c_cp ?m_cp ?u_cp) (cal_score ?c_ra ?m_ra ?u_ra) (cal_score ?c_la
?m_la ?u_la) (cal_score ?c_we ?m_wi ?u_wi) (cal_score ?c_hd ?m_hd ?u_hd)
(cal_score ?c_bs ?m_bs ?u_bs) (cal_score ?c_ss ?m_ss ?u_ss) (cal_score ?c_3d ?m_3d
?u_3d) (cal_score ?c_so ?m_so ?u_so) (cal_score ?c_dr ?m_dr ?u_dr) (cal_score ?c_dw
?m_dw ?u_dw) (cal_score ?c_we ?m_we ?u_we) (cal_score ?c_ba ?m_ba ?u_ba)
(cal_score ?c_fi ?m_fi ?u_fi) (cal_score ?c_us ?m_us ?u_us) (cal_score ?c_bl ?m_bl
?u_bl) (cal_score ?c_in ?m_in ?u_in))))
)
```

Finally, the result are printed out by this rule

```
(defrule printout_all_laptop ""
	?com <- (spec_template (status match) (model_text ?c_mo) (CPU_text ?c_cp)
(RAM_text ?c_ra) (LAN_text ?c_la) (wireless_LAN_text ?c_wi) (HDD_text ?c_hd)
(screen_size_text ?c_bs) (3d_accelerator_text ?c_3d) (DVD_text ?c_dr) (weight_text
?c_we) (batt_text ?c_ba) (firewire_text ?c_fi) (USB_text ?c_us) (bluetooth_text ?c_bl)
(infrared_text ?c_in) (score ?c_sc) (price ?c_pr))
=>
	(printout t " **SCORE:" ?c_sc "** Price: " ?c_pr " Model name " ?c_mo "
CPU:" ?c_cp " RAM:" ?c_ra " LAN:" ?c_la " WIFI:" ?c_wi " HDD:" ?c_hd " SCREEN:"
?c_bs " ACCELERATOR:" ?c_3d " DVD:" ?c_dr " WEIGHT:" ?c_we " BATT:" ?c_ba "
FIWI:" ?c_fi " USB:" ?c_us " BLUETOOTH:" ?c_bl " INFRARED:" ?c_sc crlf)
)
```

The user’s answer and how to analyze

Normally if the question we ask the user is not a general question, the system tends to offer the user with a list of allowed answers. For example, the user is asked

“When you have leisure time, what do you usually do?”

The possibly answers provided by the system will be :

(reading/ chatting/ surfing/_Internet/ play_computer_games/ travel/ stay_with_family/ going_with_friend/ listen_to_musics/ watching_movies/ create_homepage/ outdoor_activities/ other)

If no choice matches to the user’s activity, the user can input ‘other’. This is make the system more flexible as we do not want to force the user to choose what is really not match with their real activity.

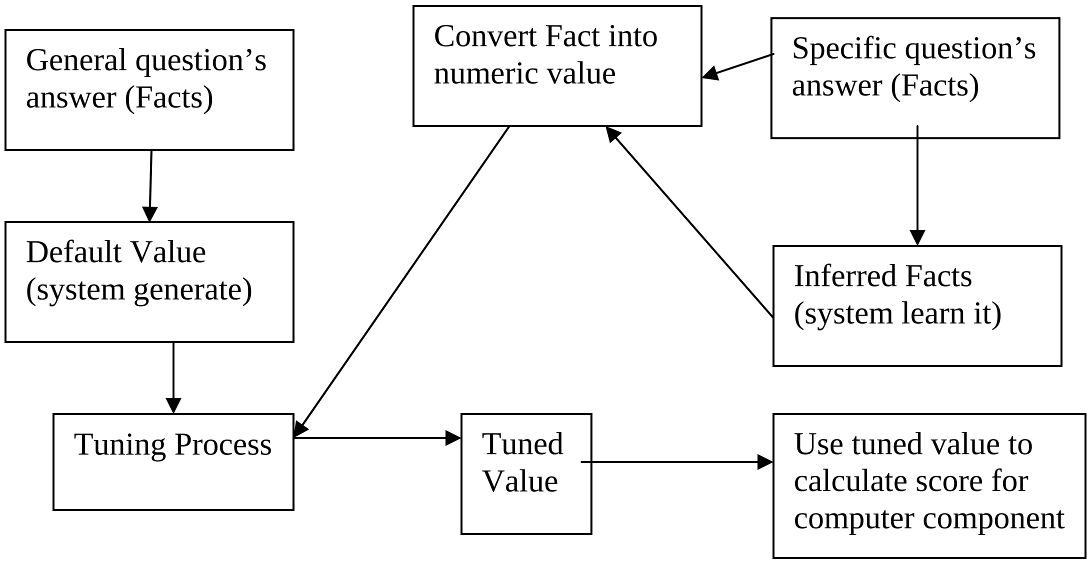
Due to the some limitation, the answer insert into the system must be inserted in exactly the same pattern with the choice offered and the only one choice can be chosen at a time. The following is another question.

“Do you have any computer or laptop now?

(have_com not_have_com)

please select one: have_com”

After getting all answers, we will analyze these facts. Please see the following diagram for clearer flow of facts.


General question's answer (Facts)
Convert Fact into numeric value
Specific question's answer (Facts)
Default Value (system generate)
Inferred Facts (system learn it)
Tuning Process
Tuned Value
Use tuned value to calculate score for computer component

Thank you for showing an interest in our project.