Master Thesis

A Knowledge-Based Service Approach for Human Centered Robots

Md. Omar Faruque Sarker

Human Computer Interaction and Robotics

UNIVERSITY OF SCIENCE AND TECHNOLOGY

August 2007

A Knowledge-Based Service Approach for Human Centered Robots

Md. Omar Faruque Sarker

A Dissertation Submitted in Partial Fulfillment of Requirements
For the Degree of Master of Engineering

June 2007

UNIVERSITY OF SCIENCE AND TECHNOLOGY
Major of Human Computer Interaction & Robotics

Supervisor Dr. ChangHwan Kim

We hereby approve the M. Eng. thesis of
“Md. Omar Faruque Sarker”

June 2007

UNIVERSITY OF SCIENCE AND TECHNOLOGY

To my parents and family

Inspiration

From the History of Adam (peace be upon him)

Narrated in the Holy *Quran*, the Book of God (the Only Deity)

The interpretation of meaning from *Chapter 2: Verses 31-33*

And He (God) taught Adam all the names (of everything) , then He showed them to the angels and said, "Tell Me the names of these if you are truthful".

They (angels) said: "Glory be to You, we have no knowledge except what you have taught us. Verily, it is You, the All-Knower, the All-Wise".

He said: "Adam! Inform them of their names," and when he had informed them of their names, He said: "Did I not tell you that I know the Ghaib (unseen) in the heavens and the earth, and I know what you reveal and what you have been concealing?"

The interpretation of meaning from *Chapter 2: Verses 37-38*

Then Adam received from his Lord Words. And his Lord pardoned him (accepted his repentance). Verily, He is the One Who forgives (accepts repentance), the Most Merciful.

We said: "Get down all of you from this place (the Paradise), then whenever there comes to you Guidance from Me, and whoever follows My Guidance, there shall be no fear on them, nor shall they grieve".

ACKNOWLEDGEMENT

All praises and thanks are due to God, the One-Unique Almighty Lord, for the successful completion of this thesis.

I greatly thank my adviser **Dr. ChangHwan Kim** and **Mrs. Jung-Min Park** and acknowledge for their invaluable guidelines and inspiring words throughout my Masters study at Korea Institute of Science & Technology (KIST). No word can be considered enough to express my thanks for those great guidelines and supports.

I am glad to express my gratitude to **Dr. Bum-Jae You**, Director of our research group at Center for Cognitive Robotics Research (CCRR) for his keen attention and encouragement for my research. I am also happy to thank **Dr. Heedong Ko**, Director of Imaging Media Research Center, for his valuable suggestions and comments that improve my work significantly. I appreciate and thank **Dr. Mun-Ho Jeon**g for his time in reviewing my thesis.

I am also happy to thank my graduate course teachers, **Dr. Woojin Chung** and **Dr. Shinsuk Park** from Dept. of Mechanical Engineering, Korea University, who inspired me a lot in my graduate classes to develop my ideas of this thesis.

I also greatly acknowledge the thesis publication efforts of the concerned staffs of Int'l R & D Academy, KIST as well as staffs of my university, University of Science & Technology (UST).

This thesis uses a lot of Free & Open-Source Software (FOSS) tools. So, I'm cordially thanking the FOSS community specially the developers of **Player/Stage** (and **Java-Player** client library) and **XSB** (and its Java bridge **Yajxb**) for their immense contribution to build great software platforms for research.

I must appreciate and thank for the great assistances received from my lab members at KIST (Center for Cognitive Robotics Research), in performing necessary experiments of this thesis, especially Il-Ku, Jae-Gu, Snag-Ik, Jung-Min, Seok-Min, Jae Sung, Mr. Jin-Sung Kim and all other past and present researchers of my lab.

I also thankfully remember the inspirations of the members of the Pioneers-KIST group and the Friday Islamic Session, and many other IRDA students including Bangladeshi students who constantly provide me mental support to continue my study and research here in Korea.

Finally, I greatly remember the contribution and support of my parents, family members and friends staying at my home country, Bangladesh, who make good wishes and prayers to the Almighty God for my continuous achievement and success in life.

ABSTRACT*

A Knowledge-Based Service Approach for Human Centered Robots

This thesis deals with a novel knowledge-based (KB) service approach for improving the quality of services of human-centered (HC) robots such as customizing, adapting, prioritizing or even socializing services based on user guidelines/polices or other user-defined knowledge. Like any other service robot, a HC robot requires to operate robustly usually in indoor environment by considering the knowledge of physical environment (through sensors, world model or maps etc). On the other hand, in order to satisfy the user requirements and expectations in services, HC robots also require to consider the knowledge of human environment (such as human relationships, guidelines, preferences, priorities and so on). This phenomenon is the primary driving force to select a KB service approach for representing user defined knowledge to HC robots in order to provide better HC services. This KB service approach extends the generic hybrid (deliberative/reactive) paradigm of robot control and adds significant value to it by empowering HC robots with human-like high-level judgment or inference-power in service tasks. In order to materialize this approach, a Knowledge-Based Human-centered (KBH) architecture has been developed by adopting hybrid robot control strategy. Two scenarios: 1) a KB object delivery service in an office environment and 2) a KB patient-care service in a hospital environment are considered to show the effectiveness of this approach. User guidelines or policies of those services are translated into logic programming modules in XSB deductive database system and integrated with the planning component of the KBH architecture. The scenarios have been simulated in Player/Stage mobile robot simulator. Different components of the KBH architecture has been developed in Java and integrated with the simulation engine through Java client libraries of Player/Stage and XSB bridging program, YAJXB. Static and dynamic re-planning is performed by KB planner and satisfactory results have been found regarding scalability and real-time planning ability by KB planner. From these simulations it is found that a HC service robot can effectively customize/prioritize services based on user guidelines. It makes human-robot interaction more efficient by saving the user's time and effort during interaction. Furthermore, this approach can be used to customize and/or adapt HC robot-services in various application contexts based on custom user guidelines related to safety, security, social adaptability and others.

* A thesis submitted to the committee of the University of Science and Technology in a partial fulfillment of the requirement for the degree of Master of Engineering conferred in August 2007.

Contents

List of Figures

List of Tables

List of Acronyms

Frequently Used Terms

HC	Human-Centered
HRI	Human-Robot Interaction
KB	Knowledge-Base/ Knowledge-Based
KBH	Knowledge-Based Human-Centered
QoS	Quality of Service
UI	User Interface

KBH Software Architecture Related Terms

ES	Executive Supervisor
SP	Strategic Planner
EP	Executive Planner
SC	Service Coordinator
BC	Behavior Coordinator
STM	Short-Term Memory
LTM	Long-Term Memory
PB	Policy-Base

1. Introduction

1.1. Human-Centered Robots

Robotics is a broad discipline. According to Webster dictionary a robot is: *"An automatic device that performs functions normally ascribed to humans or a machine in the form of a human".* Although historically robots serve the industrial world for many decades, modern robots are now entering into human living environment with various useful services. They are emerging as personal assistants, entertainers, child educators and so on. They are not only physically interacting with people but also showing their excellence by demonstrating their intelligent behaviors in actions.

According to International Federation of Robotics (IFR) (IFR, 2007), a professional organization, the definition of service robot is: *"A robot which operates semi or fully autonomously to perform services useful to the well being of humans and equipment, excluding manufacturing operations"*. From this definition, it is clear that the terms '*service robot*' and '*intelligent service robot'*, are exchangeable. IFR also classified service robots into three categories:

1. Service robots that serves humans (personal safeguarding, entertainment etc.),
2. Service robots that serves equipments (maintenance, repair, cleaning etc.) and

3. Other service robots performing an autonomous function (surveillance, transport, data acquisition, etc.) and/or service robots that can not be classified in the above two groups.

Therefore, human-centered (HC) robots are primarily falls into the first category of service robots. They can be identified as personal robots that provide personal services with unique characteristics in terms of human-robot interaction, autonomy or intelligence etc.

There are many examples of HC robots in robotics literature. From among the categories suggested by IFR and the Technical Committee for Service Robots of the IEEE Robotics and Automation Society (IEEE/RAS TC for Service Robots, 2007), the following are the major categories of HC robots.

1. **Human Assistant Robots:** They can be an assistant for elderly, handicap or child for assisting daily life tasks, e.g., taking food or medicine, performing rehabilitation tasks and so on. Examples of this category include Care-O-Bot (Fraunhofer IPA, Germany).
2. **Office and Guide Robots:** Some service robots work as receptionist and guide in offices, hospitals, museums etc. Examples of this category include humanoid robot ASIMO (Honda, Japan).
3. **Education and Training Robots:** Some service robots have also been used as child educator and trainer for handicap.

4. **Entertainment Robots:** Some service robots focus mainly on entertaining people as well as providing basic education and training which can be called as edutainment robots. Example of this category includes child educating robot IROBI (Han et al. 2005).
5. **Home Security and Surveillance Robots:** One important application of this type of service robots is home security and surveillance. Example of this category includes CyberGuard from Cyber Motion Inc.
6. **Hobby and Toy Robots:** Some robots with limited service capabilities act as toy and many researchers considered them as a class of human centered robots. Example of this category includes Robosapien from Woewee Inc.
7. **Other Personal Service Robots:** Many service robots that do not serve the human directly, but live in domestic environment for assisting daily tasks such as floor cleaning (e.g., Roomba from iRobot Inc.), lawns mowing etc. are also considered human centered robots.

Some of the remarkable characteristics of these service robots include: aesthetic look and feel, sophisticated interfaces for human interactions and increased intelligence for robust autonomy in dynamic and unstructured environments.

1.2. Need for Knowledge-Based Service Approach

Physical and Human Environment for HC Robots

The placement of intelligent service robots in human living environment requires extending their level of knowledge and capabilities significantly. If we consider a lawn-mowing robot, it can provide lawn-mowing service by considering only its surrounding physical environment. It does not need to consider the characteristics of the humans in its environment. The term '*physical environment'* means the physical objects (static or dynamic), ambient conditions (e.g., temperature/lighting condition) etc.

However, when an office delivery robot is asked to prioritize its delivery tasks according to the seniority of the office staffs (i.e., senior staff will be served first), the attributes of persons (e.g., job title) of the human environment must be taken into account. Here, the term '*human environment'* means the relationships among humans and objects, human attributes, preferences etc. For a human-centered (HC) robot, it is necessary to consider the human environment along with the physical environment for providing HC services.

This additional requirement, i.e., representing human environment's knowledge to a robot, is considered as the starting point for proposing a knowledge-based service approach for HC robots.

Beyond Behavior Based Robot Control for HC Robots

In robotics literature and commercial service robots, it is found that, in order to operate in dynamic and unstructured environment, a *hybrid (deliberative/reactive) paradigm* is followed where both deliberative (hierarchical) planning and reactive execution of tasks are supported by a set of behaviors in context of their physical environment. (A review on robot control software architectural paradigms, as found in leading robotic texts by Arkin (1990a), Murphy (2000) and others, has been provided in Section 2.2). Under the hybrid paradigm, if only this deliberative/reactive strategy is employed to operate HC robots, it seems that in many cases it is very difficult and inconvenient to incorporate the knowledge of human environment in HC robots' services. (Hereafter, the term '*generic hybrid robot control strategy*' is used to denote the idea of already existing deliberative/reactive robot control strategy that does not consider the knowledge of human environment and the consequent robotic software architectures that follow this strategy are categorized as '*generic hybrid architecture*'.)

For example, if a robot has a specific behavior, *stay-away*, by which it keeps itself away from something; it would behave same all the time on recognizing that particular thing. If the robot works in an environment where only physical environment is main concern to its behaviors, usually this kind of behavior is sufficient to keep itself away from dangers or threats.

But in a HC service context, these rigid behaviors are not enough to support human expectations. For example, if an edutainment robot is serving to a family, the mother of the family might want to turn off this behavior during the playing time of her children (so that children can play freely with the robot). Or, she might program the robot to turn on this behavior for other times (e.g., study time). However, if she asks the robot to turn off this behavior on weekends and holidays, the robot should know the definition of *weekend or holiday* which might changes over time. In this kind of services, the robot needs to capture the knowledge of dynamic contexts of human environment and to adapt its services accordingly with this motto: *"use knowledge and regulate behaviors"*.

So, it is necessary to consider a new approach which can model the human environment by user defined knowledge. In that case, the generic hybrid robot control strategy under hybrid paradigm should be extended to incorporate that knowledge of human environment.

User Defined Intelligence in HC Robots

Furthermore, in terms of functional roles of service robots, it has been noticed that there exist significant differences between HC robots and other classes of service robots. Although there are many similarities in their basic functionalities, visible differences can also be found in many aspects. Some aspects include personalization of services, human-robot interaction, adaptation and learning from human, additional functional capabilities for human safety and security etc. Service robots living in human environments will face many unique kinds of situations that need to be solved in more elegant and insightful ways. Robot has to consider not only the perceived information through its sensors from the physical environment but also the norms and practices of the human environment where it is located.

Let us consider a scenario where a robot is standing in front of two persons in an office environment as shown in Fig 1.2-1.

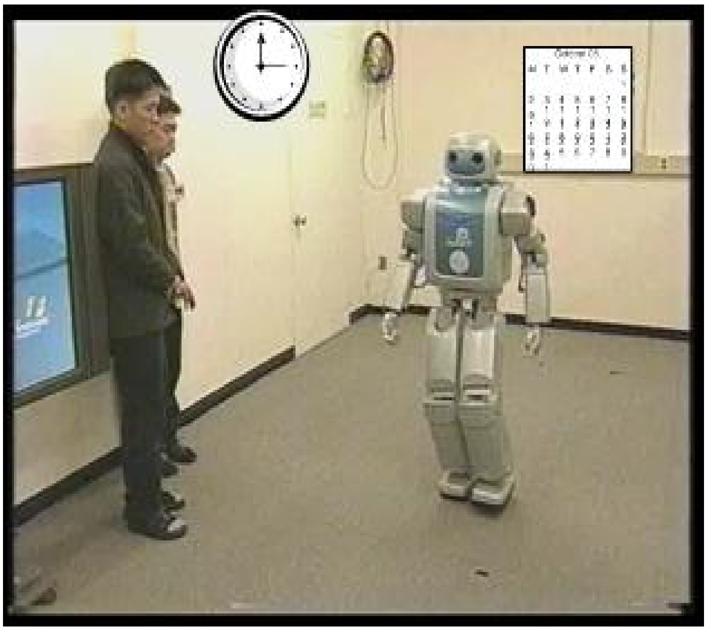


Fig.1.2-1 Humanoid robot, *Mahru*, is interacting with people

The robot is asked to move to the nearest person and shake hand with him. This is a trivial problem for most of the robots as it can sense the nearest person and approach him easily. However, if the user asks the robot to move to the *senior* person first and shake hand with him, how the robot can find the senior person. This problem might not be solved generally by supplying a generic vision algorithm to find older person from skin characteristics as the definition of a *senior* person usually differs in different environments. For example, in an office environment seniority is defined in terms of service designation or title and/or employment period. Thus, HC robots will experience similar kinds of problems in the every day services.

From the above discussions, it is evident that HC robot's knowledge, sensed and interpreted by it, is not enough to plan/execute a human-centered service. It needs to consult the knowledge-bases of user-defined policies, guidelines etc. specific to its working environment. Here, two kinds of robot intelligences are defined as stated below.

1. ***Manufacturer-defined Intelligence*****:** Robot's knowledge provided by its manufacturer/ service developer (e.g., object recognition algorithm in case of the above scenario).

2. ***User-defined Intelligence*****:** Robot's knowledge provided by its user (e.g., policy for inferring seniority in case of the above handshaking scenario).

The former kind of intelligence can help robots to perform their basic functions robustly with necessary autonomy, whereas the later one can be a basis for customizing and adapting robots' services according to their working contexts (consisting of location, time, place, concerned users etc).

However, currently almost no or very little attention is paid to develop a robot's autonomy based on user defined intelligence. From robotics literature, it seems that most of the researchers in service robotics field have put their efforts to improve the functional capabilities of robots, although the adaptation of robot behaviors or services according to human guidelines is still not taken into account.

Essence of KB Service Approach for HC Robots

Therefore, a novel knowledge-based approach is argued here to accommodate user-defined intelligence in HC robots in context of human environment. The basic research motivation for this approach is summarized below:

- To consider both physical and human environment in service planning and execution of HC robots.
- To extend the generic hybrid robot control strategy where user guidelines will be considered in planning/execution of HC services. Based on user guidelines, those services should become more socialized, secured, safe and interactive to the users.

- To adapt HC robots' services in various working contexts (place, time, user etc.) and to customize their services based on gradually developed user-defined intelligence.

Prospective Consequences of KB Service Approach

The consequence of this approach is described as below.

- The anticipated outcome of this approach is to ensure better personalization and more customization of robot services based on user supplied knowledge. This should enable a service robot to understand relationships among users, norms and manners of current environment/situation like humans and to adapt its services according to the knowledge of present working context.
- In the above cases, robots can be handed over from one place to another, one user to another, for a wide range of services with little or no system modification.
- HC robots can make their services more socialized, secured and safe based on custom user-guidelines for specific environments.
- Moreover, this approach is proposed to portray a service robot as an obedient assistant in user's mind. Here, *obedient assistant* means the robots service planning and execution should be compatible with the user supplied guidelines or user-defined intelligence which is independent of the conditions of the physical environment.

1.3. Research Objective and Scope

Major objectives of this research include:

i. To design knowledge-based services in human-centered robots.

ii. To provide modular software architecture that can integrate knowledge-bases of user-defined knowledge.

iii. To realize knowledge-based services through useful robot services.

iv. To evaluate the architecture and to ensure that this approach is applicable to a wide range of services.

In order to identify the potential benefits of this knowledge-based service as well as to realize a path to the implementation of this approach, the focus of this research has been made to the following specific areas:

- A scenario of KB object delivery service in an office environment will be used primarily to develop user defined intelligence of an office service robot in context of performing a static type of planning for a fetch-and-carry type task. Another scenario of patient care service in a hospital will also be used to show the dynamic re-planning of tasks using knowledge-bases.
- For integrating knowledge-bases with other software components of robot, modular and scalable software architecture should be built to fit it on top of most commercial/open source robot software architectures. Therefore, in this work an open source mobile robot

simulator, Player/Stage (2007), which is widely used by robotics researchers around the world, will be used to simulate knowledge-based services by a robot available within the simulator.

- This approach will be evaluated in terms of scalability, quality of service and effectiveness in applying to a real human environment such as the object delivery service in an office environment.

1.4. Thesis Outline

Chapter 1 is this introductory chapter.

Chapter 2 describes related work in intelligent service robotics, robot control software architectures and a review on knowledge-based systems.

Chapter 3 proposes the knowledge-based service approach and the background of Knowledge-Based Human-centered (KBH) software architecture with potential benefits and application areas of KB service approach.

Chapter 4 describes the design of KBH software architecture along with implement outline of the proposed KB service approach.

Chapter 5 describes two cases of developing and implementing this KB service approach: a KB object-delivery in an office and a patient-care service in a hospital. This chapter also includes the evaluation of this approach based on simulation results.

Chapter 6 concludes this thesis with a summary, remarks and future works.

2. Background and Related Work

2.1. Intelligent Service Robots

Many active researchers have been focused their attention to design and develop intelligent service robots (ISR) for various human services either from the robot manufacturer’s or from application service developer's point of view. This implies that existing research efforts are mainly on hardware and software design, development, integration and deployment in a designated environment. Some examples include: Care-O-Bot for home assistant service (Graf et al. 2004), Pearl for nursing homes (Joelle et al. 2003), IROBI for child education (Han et al. 2005), tour-guide Rackham in museum (Clodic et al. 2005), social hospital service robot IHSR (Shieh et al. 2004), and home cleaning robot Roomba (Forlizzi et al. 2006).

Sakai et al. (2005) designed an office service robot and presented some human-robot communication scenarios based on a XML language. But it is not clear how the end-user can customize the different services in different environments. Kim Moonzoo, et al. (2005) presented an interesting case study of re-engineering the software architecture of a home service robot, Samsung SHR100. Similarly, many other research groups have been working continuously for improving the functional capabilities of service robots. However, most of them are not serious about end-user’s feedback after installing a robot at a human-living environment like home.

Several other research groups integrated the service robots in real service scenarios and thoroughly reviewed their systems by end-users. Many interesting results have been published in regarding the user feedbacks. In case of Roomba (Forlizzi et al. 2006), a cleaning robot, users expressed their expectation for seeing robots always as an intelligent machine that would gain knowledge over time and adapt its behavior accordingly. In case of a tour-guide robot presented in (Clodic et al. 2005), different kinds of persons (e.g., children, adults, elderly etc.) showed different kinds of views while approaching the robot. However, in all cases, there was hardly any facility to modify the behavior of the robot for a particular service based on environmental information.

Human-robot interaction (HRI) is one of the major research issues in service robotics. Thrun (2004) reviewed the exiting technologies and major research efforts to improve the interface between human and robot using various modalities. It seems that most of those efforts have been made to improve the perception and communication capabilities of robot, although the adaptation of robot behaviors or services according to human guidelines is still not taken into account.

2.2. Robot Software Architectures

Robot software architecture represents the structure of the software, the way which the robot processes sensory inputs, performs cognitive functions, and provides signals to output actuators, independently of how it was designed (Russell & Norvig 2002). Traditionally, robotics has three paradigms of software architecture: 1) Hierarchical, 2) Reactive, and 3) Hybrid Deliberative/Reactive (Murphy 2000, Arkin 1990a). They are explained here briefly.

1) Hierarchical Paradigm

Initial design of robots used hierarchical or simply known as *Sense-Plan-Act* (SPA) paradigm for control of robots. In this paradigm, a system is divided into sensing, planning and execution systems. Those systems are cascade concatenated and the control flow is unidirectional as shown in Fig. 2.2-1. Here a system is present with an elaborate model of the world, using sensors to update this model, and to draw conclusions based on the updated model.

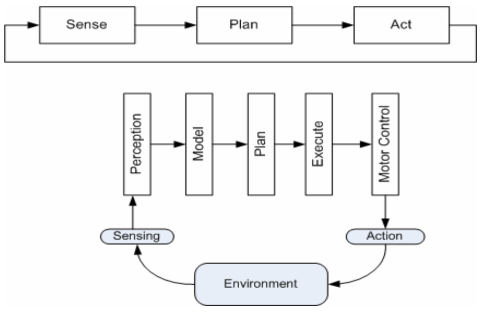


Fig. 2.2-1 Conceptual design of hierarchical paradigm of robot control

(Courtesy, Murphy 2000)

The SPA systems did not perform very well, i.e. it is found slower and less flexible, partly because of the difficulty in the modeling of the world, partly because of relying too much on inadequate sensors (Lnidstrom, et al. 2000).

2) Reactive Paradigm

Brooks (1987) revolutionized the field of robot control by presenting an architecture based on purely reactive behaviors with little or no knowledge of the world (as shown in Fig. 2.2-2). Here, a system is divided by behaviors not by functions. Tight coupling between perception and action via behaviors is present. They perform faster than SPA robots due to reactive strategy. However, the purely reactive scheme does not perform well when performing complex tasks or achieving long-term goals.

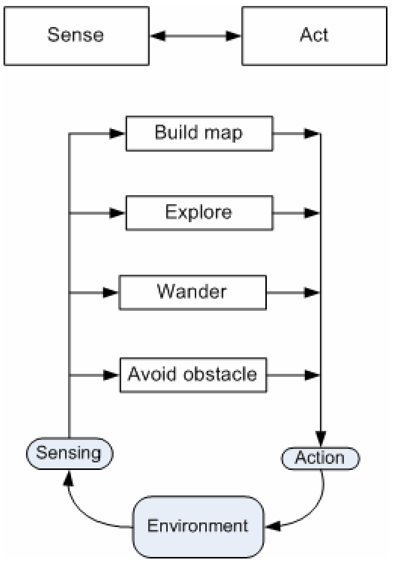


Fig. 2.2-2 Conceptual design of reactive paradigm of robot control

(Courtesy, Murphy 2000)

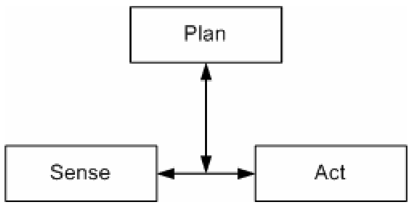


Fig. 2.2-3 Conceptual design of hybrid paradigm of robot control

(Courtesy, Murphy 2000)

3) Hybrid (Deliberative/Reactive) Paradigm

A hybrid approach has been introduced to robotics researchers, primarily by Arkin (1990a), where world model is used for planning with closed loop reactive control as seen in Fig. 2.2-3. The hybrid systems are usually modeled as having three layers (as found in Fig. 2.2-4): one deliberate, one reactive and one middle layer.

The bottom reactive layer performs repetitive calculations on raw or extracted data. These calculations should be carried out in near real-time for safety-critical considerations. The modules in this layer are normally stateless. The top deliberate layer handles planning, localization, reasoning and interaction with human operators. The middle layer, often called either the sequencer layer, or supervisory layer, bridges the gap between the deliberate and the reactive layers.

The reactive layer of a hybrid system is often behavior based. This means that the subsystem consists of separate behaviors, where each behavior has one specified non-complex task. The behaviors represent a tight coupling from the sensors to the actuators.

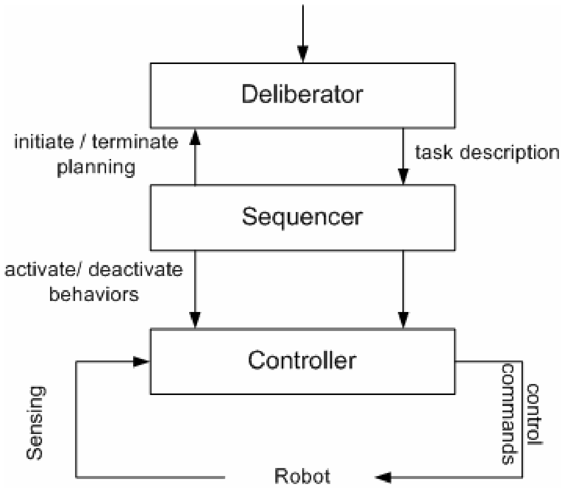


Fig. 2.2-4 Atlantis: an example of generic hybrid architecture.

The reactive layer also consists of sensors and actuators. Sensors produce data that are passed on to one or more concurrently running behaviors. Sensor fusion modules can extract higher level data from two or more sensors. Since several behaviors can be active at the same time, the results must be fused into a single crisp actuator command. This is done in an actuator command fusion module.

An example of a generic hybrid architecture, Atlantis (Gat, 1997), applied to Mars rover Robby, is shown in Fig.2.2-4. Such architectures are used in most of the robots reported in the recent literature, e.g., RAP (Firby, 1989), Xavier (Simmons, 1994), Saphira (Konolige and Myers, 1996), and 3T (Bonasso et al., 1997). The major issues of hybrid architecture have been identified as robot hardware abstraction, extensibility and scalability, run-time overhead, software tools and so on.

2.3. Knowledge-Based and Deductive Database Systems

Knowledge-Based Systems

A *Knowledge-Based System* (KBS) is a computer program that employs knowledge and inference to solve problems (Koussoub'e, 1997). A KBS has explicit representations of knowledge as well as inference process that operate on these representations to achieve a goal. An *expert system* is a kind of KBS that uses models of the knowledge and inference procedure of an expert to solve problems. A KBS deals with knowledge about states and knowledge about actions. The knowledge occurs as two kinds: declarative and procedural. *Declarative knowledge* is a description of facts. It is information about real-world objects and their properties, and the relationships among objects. *Procedural knowledge* encompasses problem-solving strategies, arithmetical and inferential knowledge. Procedural knowledge manipulates declarative knowledge to arrive at new declarative knowledge.

Knowledge Representation

A KBS needs the way to represent knowledge. Knowledge representation means that knowledge is formalized in a symbolic form, that is, to find a symbolic expression that can be interpreted. Traditionally in AI, the major way of knowledge representation is property list, rules, semantic nets, frames and logic. Most KBS rely on rules to change the state of the problem space. Each rule consists of two parts: The operator that perform the state change, and the set of conditions that determines when an operator may be executed. The general form is:

IF <*condition*> THEN <*operation*>.

If the condition is logically true, then the conclusions can be said to be logically true. Rules may be composed of a set of conditions and several conclusions. Conditions and conclusions are propositions which can be evaluated to true or false.

Architecture and Inference Procedure

The architecture of KBS is based on the production system concept. The main component is a set of rules representing the knowledge of the system. In most rule based systems the knowledge-base consists of two parts: a rule base and facts. Each rule represents a piece of knowledge and is given as "IF-THEN" statement. Facts are represented as attribute-value propositions. The conclusion of a fact contains inferred facts. The conclusion may contain actions that should be executed if the rule is fired.

The mechanism that performs the search and reasoning in a rule based KBS is called an *inference engin*e. The inference engine performs the control cycle of KBS, i.e., finds the rules whose conditions match the given facts, selects which rule to be executed, and executes the rule by adding the deduced fact to the working memory. The control cycle include the following functions:

- Selection of rule candidates: pattern matching;

- Choice of one rule: conflict resolution;
- Execution: deduction.

A rule based KBS reaches its final conclusion by a series of elementary conclusions. When a rule is applied and a new fact is concluded and added to the workspace, a new pattern of data is created that may match with new rules. This sequence of steps of linking rules with premises and conclusions is known as chaining. Inference engine works in two basic modes: forward and backward chaining. *Forward chaining* implies that the inference engine works from the initial content of the workspace toward a final conclusion through a series of *match-choose-execute* cycles. *Backward chaining* makes the inference engine work backwards from a hypothesized conclusion (a goal to be proven) to determine if there are data in the workspace to prove the truth of the goal.

Logic Programming

Logic programming is a technology that originates from declarative ideal (Russel & Norvig, 2002) that states that a KBS should be constructed by expressing knowledge in a formal language and problems should be solved by running inference process on that Knowledge. According to Robert Kowlaski (1979),

Algorithm = Logic + Control

This implies that an *algorithm* can be regarded as consisting of *logic component* which defines the logic of the algorithm and a control component which specifies the manner in which the logic component is used to solve the problem (Das, 1992). The logic component determines the meaning of the algorithm and the control component affects only the efficiency.

The logic programming language, PROLOG (PROgramming in LOGic) is widely used for developing compilers, expert systems (in legal, medicine, finance, etc.), in natural language processing and other domains. Prolog programs are sets of definite clauses written in a notation somewhat different from standard first order logic (Russell & Norvig, 2002). For formalization, Kowlaski considered a subset of first-order logic called Horn-clause logic where a clause (a disjunction of literals) with at most one positive literal. A clause or sentence in this logic may have multiple positive conditions but only one or no positive conclusion.

The execution of Prolog is done via depth first backward chaining, where clauses are tried in the order in which they are written in knowledge-base. In logic programming, a term resolution is used to refer to a mechanical method for proving statements in first-order logic. There are many different Prologs but they are all based on a technique from theorem proving known as SLD (**SL**-resolution for **D**efinite clauses) resolution. SLD-resolution is the main computation procedure used in Prolog. The instruction sets of today's computer give a poor match with Prolog's semantics, as most Prolog compilers compile into an intermediate language rather than directly into machine language. The most popular intermediate language is Warren Abstract Machine (WAM). WAM is an abstract instruction set this is suitable for Prolog and can be either interpreted or translated into machine language.

Deductive Database Systems

The deductive database concept is an extension of work by Green (1969) for his question-answer system. From the theoretical point of view, deductive databases can be regarded as logic programs which genera1ize the concept of relational databases. However, deductive database systems dea1 with sentences which are less complex than those dea1t with by logic programming systems. Also, the number of rules (sentences with non-empty conditions) in a typical deductive database system is much less than the number of facts (sentences with empty condition). Another point of contrast is that logic programming systems emphasize functiona1ities while deductive database systems emphasize efficiency. Consequently, inference mechanisms used in deductive database systems to compute answers are less general than those in logic programming systems (Das, 1992).

In addition to expressing databases sentences using logic, queries and integrity constraints may also be so expressed. Queries are equiva1ent to goa1s in the context of logic programming, while integrity constraints are properties that the data of a database is required to satisfy .Query eva1uation and integrity constraints' maintenance are two major aspects of deductive databases. Many query eva1uation techniques and a number of integrity constraint verification algorithms have been developed in deductive database contexts.

Some examples of logic programming and deductive database systems include XSB, CORAL, Aditi, LDL, Glue-Nail, Syllog etc. All of them have their own strategy and computational mechanisms for creating a deductive database using logic programs. In this thesis, discussion is limited only to XSB deductive database system.

XSB - An efficient deductive database system

As an efficient deductive database system, XSB has several rich features in many aspects. For example, XSB is based on Prolog's SLD resolution procedure for query evaluation. Since SLD mechanism has some limitations when it is used in deductive database systems, XSB extends SLD mechanism to a new mechanism called SLG by adding tabling, scheduling strategy and other mechanisms and it fully overcomes those limitations (Sagonas et al. 1994). XSB query engine is based on the extension of the WAM. So it can take the advantage of well-developed Prolog technology. The syntax of XSB is also based on HiLog, extended with Prolog operators, offers a useful means of knowledge-representation of object based schema. It has many other necessary features, such as, flexible I/O and indexing mechanisms, multiple foreign language support, interfaces and add-on packages and so on.

3. Knowledge-Based Service Approach

As discussed in Chapter 1, human-centered (HC) robots need to consider both physical and human environment for their service planning and execution. In this Chapter, the proposed knowledge-based (KB) approach has been defined and explained with some significant features and benefits in robotic applications.

3.1. Overview

The introduction of personal service robots in human living environment brings the opportunity to consider a KB service approach for HC robots. As discussed in Section 1.2, these HC robots need to consider both physical and human environment for service planning and execution. The hybrid paradigm of robot control has been successfully dealt with the robust robot control in physical environment for last decades. However, there is no or almost little scope in generic hybrid architecture to deal with human environment (e.g. human relationships/attributes, preferences, guidelines for service etc.) rather the focus is concentrated mainly on physical environment.

This situation can be explained clearly by considering an example scenario of an office service robot. If an office service robot is asked to deliver some objects to some staffs according to the descending seniority, how the robot can plan the schedule of delivery by considering the seniority of staffs? Generic hybrid architecture can support to plan a delivery mission by optimizing path or distances of the physical environment, but it can not consider the knowledge of seniority among persons.

One way to solve this problem is, to make the schedule for delivering objects or mission plan explicitly by a user according to the seniority of staffs. But this type of solution is not acceptable repeatedly over a long period of service. In reality, HC robots are expected to handle this kind of problems (like service prioritizing, customizing/personalization) by more elegant ways, not by manually interrupting the robot by a user. The root of this solution is to extend the generic hybrid architecture so that it can model the human environment along with the physical environment and it can use the knowledge of both kinds of environments for making inference in service planning/execution. Such as, in this example the robot can make the schedule or list of delivery persons/locations according to the user-defined knowledge (i.e. seniority) of the human environment and while executing the delivery task it can utilize the knowledge of the physical environment for proper path planning/navigation.

One important aspect of hybrid robot control paradigm is to deal uncertainties of physical environment by reactive behaviors. However, when HC robots need to deal with the uncertainties of human environment it should consult the knowledge-bases of user defined relationships, guidelines and so on. Therefore, a knowledge-based approach is essential in dealing with services of HC robots. So combining this approach with generic hybrid architecture can give better performance in adapting services for handling dynamic human environment while retaining the reactive behaviors for handling dynamic physical environment.

3.2. Description of KB Service Approach

Definition

A knowledge-based service approach can be defined as the service planning and execution of human-centered robots considering user supplied guidelines, preferences etc., (collectively known as user-defined knowledge) of human environment, along with sensed and perceived information from the physical environment at a given context.

Herein, *context* is defined as consisting of the current time, location, concerned users and other information that can be used to represent the current situation as interpreted by human. An example of a context might be, robot is locating in kitchen of a house, in an early morning (6 AM) and maid of the house is preparing some foods near the robot. The key context information in this example is: ‘kitchen’ (represented as location), ‘early morning’ (represented as time), ‘maid’ (represented as user) and ‘preparing some foods’ (represented as tasks of users).

Conceptual Development of KB Service Approach

The origin of the proposed KB service approach is the consequence of extending hybrid paradigm in context of HC robots. If a hybrid paradigm (as reviewed in Section 2.2) is closely observed, it can be found that the main focus of the generic hybrid architecture is to enable the robot to perform autonomously in dynamic/unstructured physical environment. So the robot's knowledge of the world is mainly the physical environment, e. g., stable map of the environment and dynamic obstacles and related other things of the environment.

Table 3.2-1: Typical features of a generic hybrid robot control architecture

Topic/Area	Key Issue(s)
Knowledge Contents	Physical environment
Major functions	Mobility (navigation), manipulation.
Functional Unit	Behavior and/or tasks
Main focus in functions	Robustness in functions
HRI Ability	Limited, can be extended by add-on modules
Social Adaptability	Limited to animal (biological) level adaptation

Here, world modeling is mainly focused on the surrounding environment of the robot. Another important noticeable thing is that the main control task of the robot is mobility, i.e., robust localization, path planning and navigation. Some robots which have manipulators also consider manipulating objects from one place to another. So, the intelligent behaviors are generally designed to support the mobility of the robot. Table 3.2-1 lists these facts.

However, in case of a HC robot, along with the dynamic physical environment it also needs to consider the human environment i.e., users and their service expectations from it. So, modeling the world must also include the human (and human to robot) relationships, norms and manners as well as contexts of the environment (e.g., office, hospital etc.), context specific guidelines for services and so on. Here, a HC service contains both mobility and interaction functions. So, when the robot is required to plan and execute a service, it needs to consider knowledge of both two environments: physical and human. Under a specific service, when it tries to perform robust mobility functions (like navigation) it considers the environmental knowledge like maps, obstacles etc. On the other hand, under the same service when the robot tries to prioritize a service tasks it needs to consider human relationships or knowledge of humans.

For example, as discussed in the scenario of the seniority based object delivery by an office service robot, at first the robot needs to make an ordered list of persons based on descending seniority and then it needs to physically deliver the objects to the persons. When the robot is initially planning the delivery tasks it is actually consulting human-defined relationships or knowledge of the logical environment (or the office). However, when it starts to deliver objects physically it needs the knowledge of the physical off ice environment like maps of corridor, room numbers of persons etc. Generic hybrid architectures are designed and used for the latter purpose (i.e., mobility of the robot in office environment). Until now, in these kind of robot control architectures, little or no attention is given to consider knowledge of human environment (i.e., human relationships, guidelines etc.) in service planning of robots (like seniority of users in office).

The role of user-defined knowledge in adapting planning of HC services can be examined in detail. In case of hybrid /hierarchical control architecture, higher levels in the hierarchy provide subgoals for lower subordinate levels and they rely heavily on symbolic representations of world models. Usually, lower levels are tied with behaviors or sensory-motor control loop. In *‘selection’* type hybrid architecture (Arkin, 1990a), where planning is viewed as configuration, the planning component determines the behavioral composition and parameters used during execution. Usually, this kind of plan is always consistent to support a set of behaviors.

For example, if a robot has a specific behavior, *stay-away*, by which it keeps itself away from something, it would behave same all the time on recognizing that specific thing (as suggested by hybrid paradigm of robot control). If the robot works in an environment where only physical environment is main concern to its behaviors, usually this kind of behavior is sufficient to keep itself away from dangers. But in a HC service context these rigid behavior is not enough to support human requirements. For example if a robot is serving in a family, a user might turn off this behavior during the playing time of the children. Or she might turn on this behavior for other times. However, if she asks the robot to turn off this behavior on *weekends* and *holidays*, the robot should know the definition of weekends and holidays which might changes over time. In this kind of services the robot needs to capture the knowledge of dynamic contexts of human environment and to behave accordingly.

Same is the case where a set of user guidelines is enforced to behave socially by a robot. In an outdoor environment a robot might have consistent behavior patterns to avoid obstacles and to plan optimum paths. However, in a public place like museum or hospital, special social norm is to be followed before attempting to avoid a crowd (treated as dynamic obstacle) or some other persons standing on its way. These kind of user guidelines or social norms of the human environment are also needed to be considered in planning the avoid obstacle task.

From the above discussions it is seen that only generic behaviors are not enough for HC robots, user-defined knowledge should regulate the behaviors to fit in the service contexts. This makes the knowledge-based approach distinct from generic behavior based hybrid robot control paradigm. Therefore under this KB service approach, the following additional features have been proposed to extend generic hybrid robot control architecture.

- **Knowledge Contents:** Human environment including the contextual knowledge of human environment.
- **Major functions:** HC services that highly rely on interacting with humans.
- **Functional Unit:** A HC servic*e* that consists of multiple behaviors and/or tasks
- **Main focus in functions:** Efficiency of HC services
- **HRI Ability:** Extended HRI (making efficient HRI by reducing interaction time/effort)
- **Social Adaptability:** Extended social adaptability (socializing services based incorporating user guidelines).

In this KB approach, it is seen that a robot can maintain services considering not only the sensor information but also user defined knowledge or guidelines. Here, the lower level physical sensory perception (e.g., current pose and other state information of robot) is used to maintain the reactive behaviors in performing necessary lower level functions such as robust navigation, localization etc.. On the other hand, the higher level logical perception (e.g., symbolic name of current location) is used to find out user defined guidelines for interpreting the situation and adapting high level behaviors such as setting high priority for a specific user in that context. Hence, the user defined knowledge should contain sufficient information for that purpose, e.g., necessary relationships among users, objects, tasks etc., service planning and executing policies such as user preferences, knowledge for judging appropriateness of a service, inferring context, priority and various other necessary information.

Another aspect of this approach is that a robot might learn new knowledge from user-defined knowledge or past service experiences based on user-guidelines or policies of inference. It implies that, this approach of teaching a robot is deductive one. In contrast to the inductive approach where robot learns from accumulated data, here it merely infers any executive decision from its own observations. Rather, it always checks the user supplied knowledge for deducing a new decision for planning and executing a service. In fine, Table 3.2-2 lists the resultant features of a *KB-service-approach-powered* hybrid architecture which is termed as Knowledge-Based Human-centered (KBH) architecture. The details of this architecture have been explained in Chapter 4.

Table 3.2-2: Major features of Knowledge-Based Human-centered (KBH) architecture.

Topic/Area	Key Issue(s)
Knowledge Contents	Both physical environment and human environment including the contextual knowledge of both environments are considered.
Major functions	Interactive HC services that highly rely on lower level functions like mobility (navigation), manipulation etc..
Functional Unit	A HC *service* that consists of multiple behaviors and/or tasks
Main focus in functions	Efficiency of HC services that also is influenced by the robustness in physical tasks
HRI ability	Extended HRI abilities (making efficient HRI by reducing interaction time/effort)
Social adaptability	Extended social adaptability (socializing services based incorporating user guidelines).

3.3. Potential Benefits of KB Service Approach

KB service approach offers a novel opportunity to make service robots more intelligent and interactive. It opens a door of enormous opportunities in terms of building reliable partnership between humans and robots. Major benefits of this approach can be found in the issues of improving quality of services of HC robots and efficiency of human-robot interaction. Some of the major benefits of this approach have been illustrated here and realized through the implementation of this approach described in Chapter 5.

1) Service Priority Management: In a traditional service approach, service robots usually plan and execute service tasks using user request and sensory input from their various sensors. In that case, prioritizing services might be done using static user information, e.g., user X can get high priority service over user Y based on predefined instructions. But in this approach, no predefined instruction is necessary as the robot can infer priority of users based on the prior knowledge of the environment, e.g., user relationships and guidelines for inference.

2) Service Customization: Specific options or parameters of a service can be customized based on the robot's knowledge of user preferences for that service or context of the environment. Any user defined guideline can be enforced in a service to satisfy the demand of the environment, e.g., office or hospital environment needs special service policies, so the robot can follow those policies if required.

3) Adapting with varying contexts:

Contextual changes, such as the change of user, location or time might influence the planning and execution of services. So, if the robot plan and execute services using a knowledge-base, it can easily adapt with those changes without reprogramming the entire service.

4) Safe, Secured and Socialized Services:

The opportunity of a robot to learn about various user-defined guidelines enables the robot to act as safe and secure as possible. A robot is not like a personal computer that a user might lock by password. So by applying user guidelines of security, a robot can conveniently set up different access levels for a specific service. For example, parents would not be worried about their children’s educator robot as it can follow proper guidelines for controlling access levels. Similarly, a robot can socialize its services by following user-defined social norms for various contexts of services.

5) Efficient Human-Robot Interaction: At run-time, if a robot spends more time for getting all detail instructions to initiate a service, the efficiency of human-robot interaction is decreased and sometimes the interactivity is seriously hampered. In case of this KB approach, robot will require minimum information to process a user’s request of service execution. Robot can implicitly infer many things from prior knowledge and hence the human-robot interaction will be more efficient and more interactive.

3.4. Potential Application Areas of KB Service Approach

The KB service approach can be applied to almost all areas of service robotics. Some major application areas have been outlined here.

1) Personal service robots:

Personal service robots need to remember personal preferences, priorities and special instructions for providing services. This can be done by creating appropriate knowledge-bases for individuals and store those information with corresponding mapping to the service parameters.

2) Professional service robots:

Many robots that work in office, museums and other places need to follow social norms for interacting with various people in public settings rather than controlled laboratory settings. So following user-defined social norms can be done with the aid of custom knowledge-bases of the specific location or context.

3) Industrial service robots:

Service robots that work in the industry might have a limited context and autonomy. However, situations may vary from place to place. So, they can also get the benefits of custom knowledge-bases instead of reprogramming each time context changes. These might increase the end-users satisfaction by customizing the services.

4. Knowledge-Based Human-centered (KBH) Architecture

In order to materialize the ideas of KB service approach, it is necessary to integrate those ideas within a robot’s software architecture. As mentioned in Section 2.2, today’s most of robotic software architecture is based on hybrid paradigm, here a hybrid software architecture is persuaded to implement our KB service approach. The proposed architecture is named as ‘*Knowledge-Based Human-centered’* (KBH) architecture. The conceptual development of KBH service approach and the basic features of KBH architecture have been introduced in Section 3.2. The designs of this architecture, the role of its components and an implementation path for software implementation have been presented in the subsequent sections.

4.1. Design of KBH Software Architecture

The major purpose of this KBH architecture is to provide the means to develop necessary software for realizing the ideas of KB service approach in real robot hardware and/or robot simulators. As shown in Fig. 4.1-1, it is organized into three layers:

1) Controller

2) Reactive-Executive

3) Deliberative.

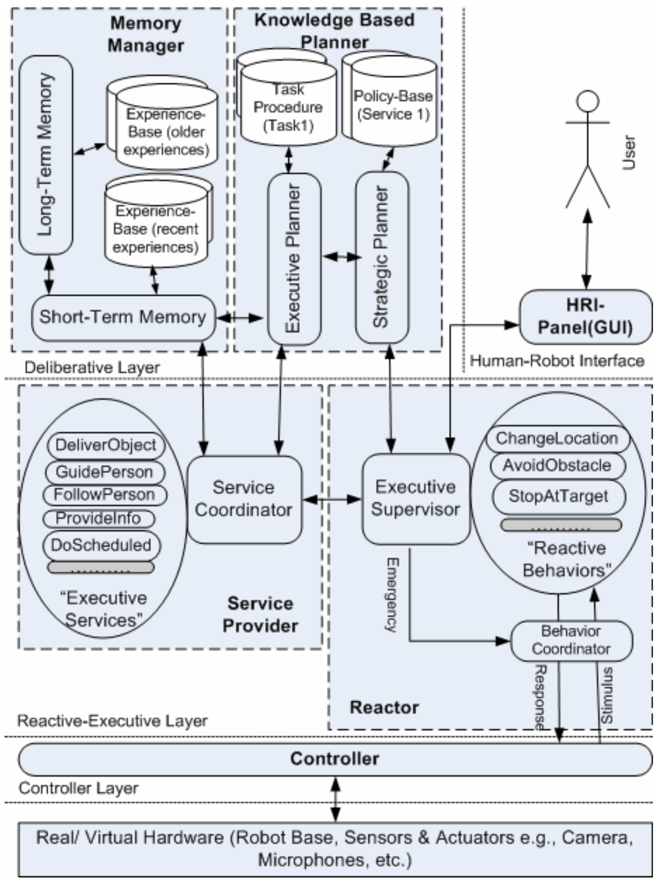


Fig. 4.1-1 Knowledge-Based Human-centered (KBH) architecture

These layers along with their component modules have been introduced here.

1. Controller Layer:

This layer containing ***Controller*** modules directly controls the robot based on reactive behaviors and/or task skills by utilizing robot's attached sensors and actuators. Controller generates the unit task of a behavior such as turning left wheel in avoiding obstacle behavior. Usually, robot manufacturer is responsible to manage the technical details of this layer and to export the necessary interfaces to the high level application developers.

2. Reactive-Executive Layer:

The necessary function of reactive-executive layer is to process the service request (received from user) by consulting with the upper deliberative layer (planning) components and employing reactive behaviors of lower controller layer. Since this layer is aware of both the user's demand for a service and the current state of the robot, it can quickly report the planner about the dynamic contexts to get necessary service instructions to carry out. Every service is consists of some tasks and every task requires some sort of reactive behaviors as well. For example, a single object delivery task requires a definite location to reach while executing *avoid-obstacle* behavior.

Two major modules: ***Reactor*** and ***Service Provider*** have two vital components of this architecture:

1) *Executive Supervisor* (ES) and

2) *Service Coordinator (SC).*

Both components have been introduced here.

1) Executive Supervisor (ES):

This ES is primarily responsible to accept users request for services, to consult planner for a decision and to activate /deactivate the reactive behaviors based on the requests from Service Coordinator (SC) and other purposes. Occasionally, it uses its *Behavior Coordinator (BC)* sub-component to send emergency commands to halt robot. BC is also used for behavior coordination and/or avoiding conflicts between multiple behaviors competing for same actuator.

2) Service Coordinator (SC):

As a HC robot will have many specific services to be served in a given environment, KBH architecture arranges the services in *Service Provider* module. A service corridor is responsible to activate, control and coordinate the *executive services* upon receiving requests from ES. It runs the service loops by consulting with *Executive Planner* (EP) in *KB Planner* and *Short Term Memory* (STM) in *Memory Manager* modules. Each service needs to complete many service tasks like localization, navigation etc. which are queued and sent to ES for timely execution by low level behaviors. SC is also responsible to organize and store the specific service experiences in STM and to retrieve it in the beginning of next service request.

3. Deliberative Layer:

In deliberative layer, the inference/planning and learning occur coherently. A human-robot interaction module is also necessary as a gateway component of this layer. For that reason, the deliberative layer contains three major modules.

1. KB Planner
2. Memory Manger
3. HRI Panel

1) KB Planner:

The physical and human environment is modeled in the KB planner. Inference is also done according to the characteristics of HC robot's services. For example, in case of a office service robot, if the planner is asked to plan an object delivery service according the seniority of the staffs of the office, a strategic plan is generated stating the list of delivery persons/locations according to the descending seniority of the staffs of the office. To avoid potential confusion, the term *strategic plan* refers to the plan that comes from human-like high level judgment (by considering the human environment or contextual knowledge in particular) about a service or simply *'what to do'* like plan (in this example, strategic plan contains a list of persons in the order of descending seniority). After a strategic plan is found the robot can generate the executive plan. The term *executive plan* refers to the plan necessary to execute a service by considering the physical environment in particular (in this example, executive plan contains the navigation/location information for delivering those objects to the staffs of the office). Consequently, a KB planner contains *Strategic Planner* (SP) and *Executive Planner* (EP) to serve two distinct kinds of planning. They gather necessary information and instructions from relational and deductive databases connected to them.

2) Memory Manager:

The opportunity of experience-based learning is also considered in KBH architecture. The accumulated service experiences of HC robots are the vital source of learning and adaptation of services. In general, higher level features of service experiences can be organized and stored in short-term and long-term memories of robot based on temporal characteristics of those experiences.

3) HRI Panel:

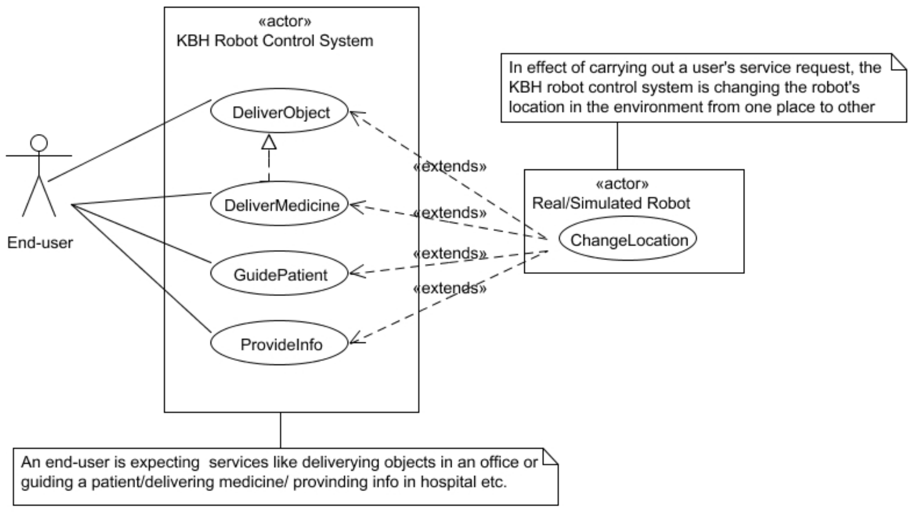
This component is for Human-Robot Interaction/Interface (HRI). It facilitates the communication between a user and *Executive Supervisor,* the kernel of the robot. It can be graphical or command line based.

4.2. Collaborative Role of KBH Software Architecture

The role of KBH software architecture and its individual components have been explained by the UML use case diagrams below.

System-level Role

As shown in Fig 4.2-1, an end-user (e.g., a secretary in an office or a nurse in a hospital who ultimately uses the system) finds the essential services from KBH robot through the use cases. The role of KBH robot control system is to provide some useful services in human living environment.



«actor»
KBH Robot Control System
DeliverObject
DeliverMedicine
GuidePatient
ProvideInfo
End-user
«extends»
«extends»
«extends»
«extends»
«actor»
Real/Simulated Robot
ChangeLocation
In effect of carrying out a user's service request, the KBH robot control system is changing the robot's location in the environment from one place to other
An end-user is expecting services like deliverying objects in an office or guiding a patient/delivering medicine/ provinding info in hospital etc.

Fig: 4.2-1 Use case diagram of KBH architecture for end-users

For example, in this use case diagram four services have been modeled after the requirements of an office and a hospital environment, namely *deliver object*, *deliver medicine*, *guide patient* and provide info. To understand the object-oriented view of services, it is seen that a *‘deliver medicine’* service is in fact *is-a ‘deliver object’* service with some extra operations (i.e., generalization of concepts).

Later on, it is also seen that every service require to execute a *‘change location’* use case in a (real/simulated) robot. That means *‘change location’* use-case is extended to create the end-user’s services. Since KBH software architecture is based on the philosophy of *‘providing service based on user guidelines of human environment’*, it is necessary to model the users’ requirements and guidelines for a service. The model should include the necessary knowledge for inferring a decision based on user guidelines.

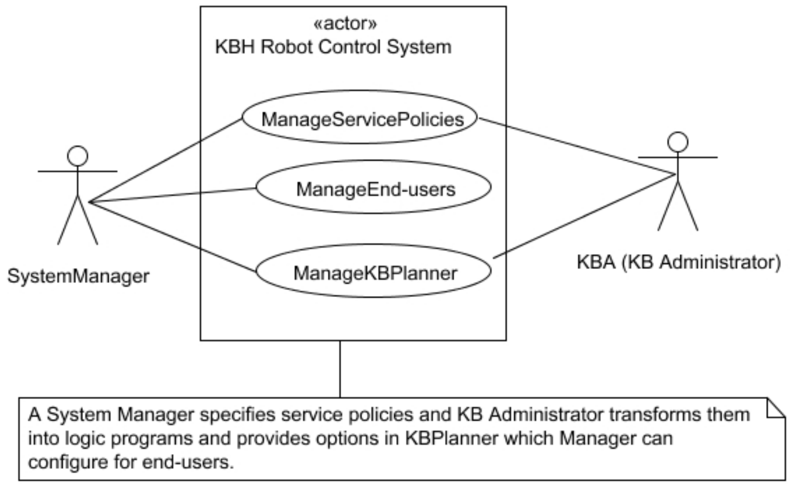

«actor»
KBH Robot Control System
ManageServicePolicies
ManageEnd-users
ManageKBPlanner
SystemManager
KBA (KB Administrator)
A System Manager specifies service policies and KB Administrator transforms them into logic programs and provides options in KBPlanner which Manager can configure for end-users.

Fig: 4.2-2 Use case diagram of KBH architecture for developers

So, in KBH system construction phase a privileged/ managerial level user (herein, termed as *system-manager*) assigns the necessary service policies to service requirement table. He or she is also responsible to manage necessary end-users who can gain access to those services. A developer level user (herein, termed as KBA or KB administrator) will take the system manager's service policies and translates into the logic programming modules that can be plugged into KB Planner, the planning component of KBH architecture. KBA will also assist system manager to configure KB planner and validate the user policies working properly. This scenario is shown in Fig. 4.2-2 as a use case diagram.

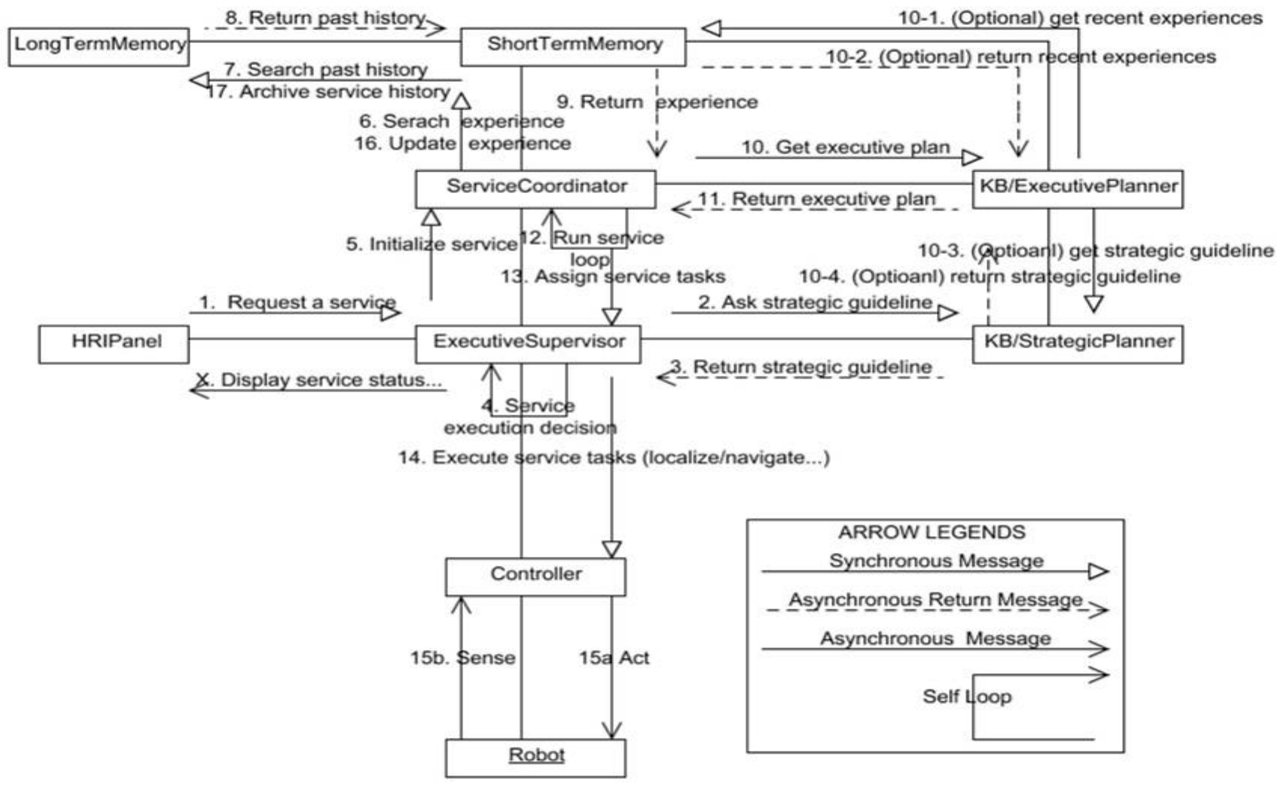


Fig: 4.2-3 Collaboration diagram of KBH architectural components

Component-level Role

In order to satisfy the system-level role mentioned above, each component is designed to support certain roles. As shown in Fig. 4.2-3, the role of each component is shown during an execution of service.

Each step of this diagram is explained below.

1. A user requests and specifies about a service through HRI Panel. For example, in case of an object delivery service, a user should describe the delivery object properties (recipient name, urgency etc.).
2. Upon receiving the user's request for a service by HRI panel, Executive Supervisor (ES) checks the availability of service resources and asks Strategic Planner (SP) about the guidelines on that particular service request and current situation or contexts.
3. SP contains the service guidelines. Such as, if the object delivery service request contains some guidelines for inferring service order based on seniority of recipients, SP returns those guidelines to ES as a reference for service planning.
4. An immediate service execution decision is made by ES based on the strategic guidelines whether the service request can proceed further. Some guidelines might prevent ES, not to proceed any more.

So, ES may send status messages to HRI Panel or proceed to initialize next service execution steps.

5. A service initialization message is sent to Service Coordinator (SC) that manages each instance of services. So, the following few steps are carried out to make an effective service plan and execute that plan in a service loop.
6. SC asks Short-Term Memory (STM) for any short-term/recent service experience.
7. Similarly, STM asks Long-Term Memory (LTM) to look up long-term/past histories related to services
8. Past histories can be returned as case structure
9. Similarly, STM can generate service experience as a case structure.
10. SC sends message to Executive Planner (EP) to get execute plan (like navigation/ path planning).
11. While planning for an executive plan, EP can consults with STM or SP as indicated in optional messages in 10-1 to 10-4. Finally it sends that plan to SC
12. SC initiates and runs a service loop
13. Each service task is delegated to ES like single movement or navigating from one location to another.

14. ES activates certain behaviors like change location, avoid obstacle while executing a service task.

15. At the lowest level of interaction, Controller commands the robot like *go X direction* or *turn a little* etc.

    Step 13-15 is repeated as needed by step 12 and occasional service messages are sent by ES to HRI Panel to update user about the current state of the system.

16. After a service loop is completed, SC prepares an experience structure containing the recent service experiences and sends it to STM for future reuse.

17. All service experiences have been archived in LTM.

X. This denotes display of asynchronous messages when necessary but without having any specific order/time of execution.

The above steps briefly identify the component level role of the KBH system. However, for brevity many synchronization messages are not shown here and a several other options for extending the interactions among components are preserved in this architecture.

4.3. Deployment models of KBH Software Architecture

The role of KBH architecture can't be realized unless it is deployed in some real systems. As shown in Fig. 4.3-1, a sample deployment diagram is presented to refer in future implementations. Although the number of deployment units (or processors) is dependent of the service requirements by end-users, this diagram outlines by the functional role of components. These deployment units are discussed below.

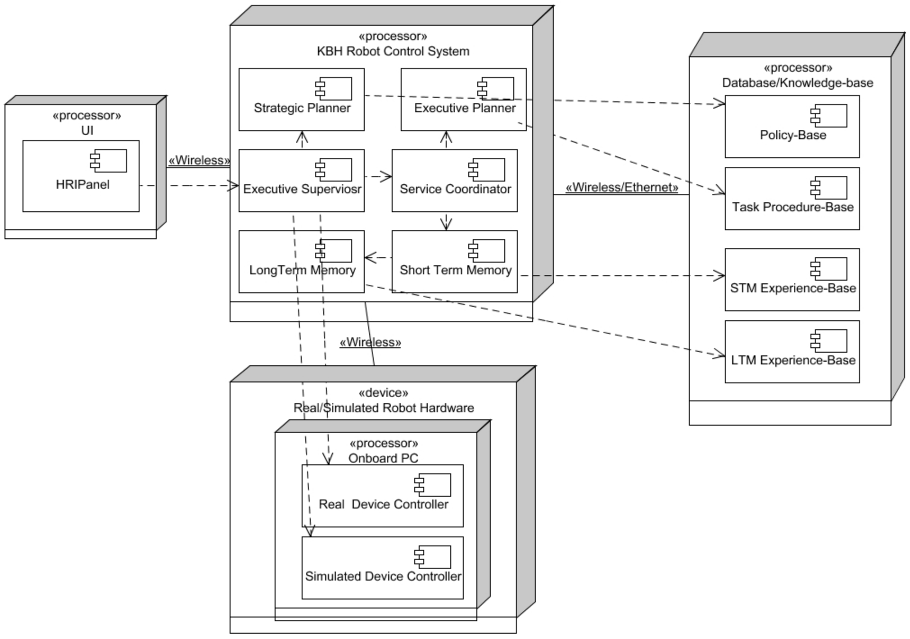


Fig: 4.3-1 Reference deployment diagram of KBH architecture

1) KBH Robot Control System: The six major components of KBH architecture: Strategic/Executive Planner, Executive Supervisor, Service Coordinator, Short-Term/Long-Term Memory reside inside this unit. Because of the close interaction of these components, it is convenient to implement these modules in a single unit. Any modern object-oriented programming language like C++ or Java can be used to develop software modules of this unit.

2) Databases/Knowledge-bases: As knowledge representation is a major issue in KB approach, so logic programs and deductive databases can be stored in separate unit of deployment. Any Prolog like logic programming language can be used to represent knowledge in Policy-bases. The facts part of Policy-bases can be represented in an extensional or relational database. The knowledge of task procedure and experiences can be represented in terms of logics, frames or any other suitable form of representation (preferably in object-oriented fashion).

3) **User Interfaces:** In this unit the user interface component such as HRI Panel can be implemented. Usually this component needs GUI support for better interaction. The commutation between this component and KBH main robot control system can be done using some remote procedure calls (RPC) such as XML-RPC or other communication method offered by modern object oriented languages and/or networking technologies.

4) Robots: Usually each robot is equipped with an on-board computer to support interfacing with sensors and actuators. Therefore, controller component is usually in this PC. In case of simulations, several deployment units can be integrated into one processor.

Customizing Software Deployment

In order to evaluate KBH software architecture, an abstract deployment model is necessary. Based on the application requirements, KBH architecture can be compacted and deployed in a single processor. This is particularly helpful for simulating the whole system before implementing it into a real robot platform. The following list highlights some customized model of KBH software architecture.

- **Minimum Deployment Model (5 Components):**

  - This model considers necessary minimum number of components of the reference deployment unit, namely Controller, Executive Supervisor, KBPlanner (that combines the role of both Strategic Planner and Executive Planner), Policy-Base (which may contain task procedures and other databases to support planning) and HRI Panel.

  - It makes a service thread in Executive Supervisor to handle service loops without making Service Coordinator a separate module. It does not contain any memory unit for learning.

  - This model is suitable for simulating one or a few number of services.

- **Multi-Service Deployment Model (6 Components):**

  - This is an extension of minimum deployment model where multiple services are managed by the Service Coordinator module. It also has a KBPlanner (that combines the role of both Strategic Planner and Executive Planner) and it does not contain any memory unit for learning.

- This model provides more flexibility in implementing multiple services. An example of this deployment model is shown in Fig. 4.3-2.

It contains Player/Stage simulator in Controller unit.

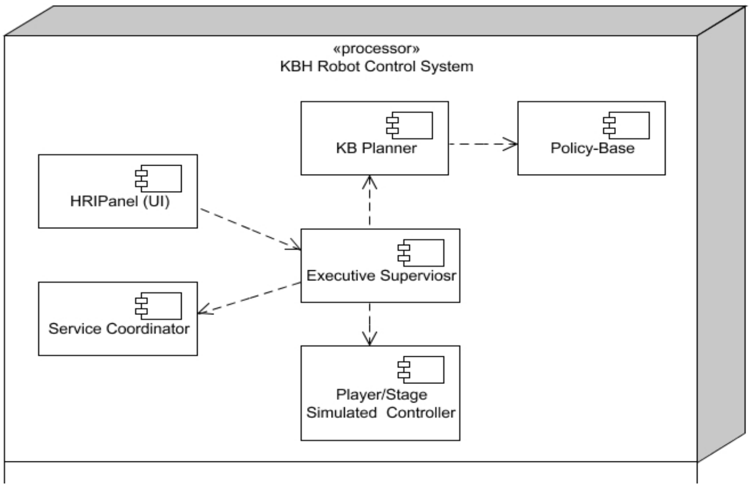


Fig: 4.3-2 Multi-service deployment model of KBH architecture

- **Multi-Service Extended Planner Deployment Model (8 Components):**

  - This model is an extension of multi-service deployment model where separate planners, Strategic Planner and Executive Planner, and Policy-Base and Task-Procedure Base are present.

  - This model offers the full power of KB planning and manages the planning process more smoothly than previous two models. It also does not contain any memory unit for learning.

- **Other Deployment Models (5+ Components):**

- Usually when the KB Planners are developed and integrated effectively, memory management modules will be appropriate to develop for learning (based on past service experiences). Initially it might contain only Short-Term Memory and STM Experience Base, i.e., a 10 component deployment model can be considered. Later on, Long-Term Memory and LTM Experience Base can be added to complete the architecture development.

Thus, based on the application requirements, KBH software architecture can be extended from 5 components minimum deployment model to 12 components full deployment model.

5. Development and Implementation

In Chapter 3 and Chapter 4, the KB service approach has been introduced and an outline of software architecture is provided respectively. In this chapter, several potential features of this novel approach will be implemented under two scenarios of a HC robot in an office and a hospital environment (as shown in the use case diagram presented in Fig 4.2-1). Section 5.1 presents an overview of both scenarios and outline of the development process. Section 5.2 and 5.3 describes the design, development and implementation of both scenarios. Following an evaluation of KB planner in Section 5.4, finally a discussion is made on the overall implementation of KB service approach in Section 5.5.

5.1. Overview

In many human-living environments including offices, hospitals and homes, HC robots are required to improve their quality of service (QoS) based on the user defined policies or guidelines for specific services. These guidelines might be part of the norms and manners of those environments or imposed by the users according to the requirements of those services. In this thesis, two scenarios of HC robots (in an office and in a hospital) have been selected to implement the ideas of KB service approach. Both scenarios have similar characteristics in implementation except in case of hospital scenario, dynamic situations have also been taken into account. The major objectives of these two implementations are outlined here.

1) Knowledge-based object delivery service at an office:

An office service robot, that can deliver objects from one place to other, might be asked by the chief executive of the office to serve the delivery objects according to importance of objects or type of recipients and/or other policies (a common example is to serve objects according to person's seniority). Therefore, this implementation in office environment has two major objectives as stated here.

- The first objective is to show how a HC robot can consider user guidelines in its service planning and execution. Here, main focus is given to develop a KB Planner that can model the human environments and consider user-guidelines into its Policy-Base (as referred in Chapter 4, KBH software architecture). Section 5.2 describes this implementation.
- Another objective of this scenario is to evaluate the performance of the KB planner while executing an object delivery service in an office environment with reasonable number of persons and user policies. Section 5.4 serves this purpose.

2) Knowledge-based patient care service at a hospital:

HC robots are also supposed to work under high work-load in dynamic situations. According to the need of the situation/context, it must respond to the changes of the environment and it must adjust its work schedule to satisfy various kinds of users' demands. This implies that the KB service approach should be verified so that a HC robot can change its service plan (or dynamically re-plan) based on the contexts of environment (here, human environment in particular). Hospital is a very dynamic and critical environment for a HC robot. So, this scenario is designed particularly to satisfy the above requirements. Major objectives are listed here.

- In this scenario it aimed to show bow a HC robot can dynamically re-plan under changing context. Here, it is assumed that when a HC robot offers a service, it gives higher priority to a doctor's request than that of a nurse. Similarly a nurse has higher priority over a patient to get a service. So, while a HC robot is serving a nurse and a patient, an emergency service request is placed to the robot by a doctor. This scenario shows how KB planner dynamically re-plans the service schedule and executes services in order.
- This scenario is also used to show the detail implementation of KBH software architecture. Although learning feature of KBH software architecture is not shown here, this scenario will help an interested reader to understand how KBH architecture can function dynamically in a multi-threaded implementation under heterogeneous software and networking technologies (such as Java, Prolog, deductive databases etc.) with an well-known robot simulator, Player/Stage.

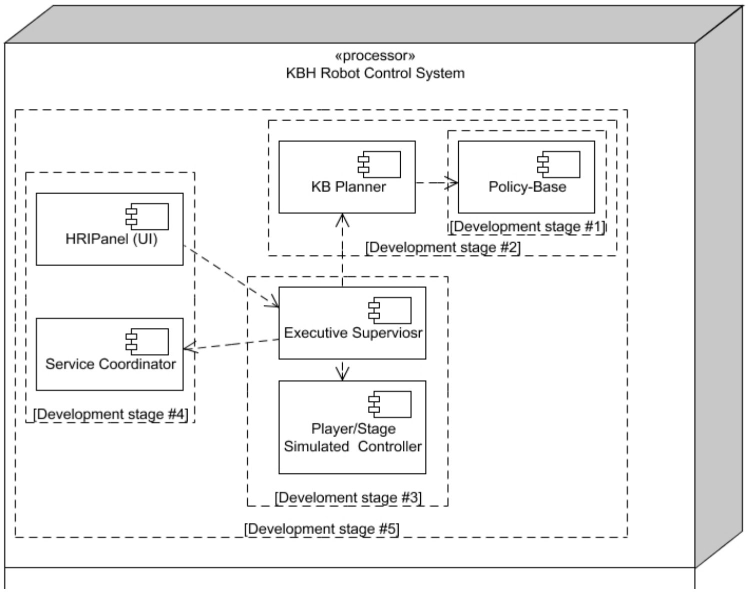


Fig 5.1-1: Stages of development of KBH software architecture

Outline of Development Process

Based on the requirements of both scenarios, Multi-Service Deployment Model (6 Components) is selected from Section 4.3. This model consists of Player/Stage Controller, Executive Supervisor, Service Coordinator, KBPlanner, Policy-Base and HRIPanel. As shown in Fig 5.1-1, five major stages for developing this model and implementing the scenarios have been outlined below.

Stage #1: Requirement analysis and knowledge-base development

In the first stage of the development process, the user- guidelines (or requirements) related to services are collected from the environment of implementation. In case of object delivery scenario in office, the user-guidelines are derived from the regular practice of an office secretary while delivering objects. These guidelines actually reflect the norms of the office and they usually reflect the instructions of the chief executive of the office. These guidelines have been discussed in Section 5.2.1. Similarly, in case of patient care scenario in hospital, the guidelines to prioritize user's request have been defined and described in Section 5.3.1.

The necessary knowledge of human environment (particulars of users, room numbers) and the knowledge of physical environment (distances among rooms, floor map etc.), are collected and documented in this stage. This knowledge is described in logic programming modules as facts (Extensional DB) and rules (Intensional DB) in XSB deductive database system. The knowledge-base development processes are outlined in Section 5.2.2 and 5.3.2.

Stage #2: KB Planner development

In this stage the KB Planner is developed so that it can generate a correct plan containing object delivery targets (in office scenario), or destination targets for a patient care service (in hospital scenario). Necessary logic programs are developed in XSB and validated the planning results in this stage. An example validation process is presented in Section 5.2.2 in case of object delivery scenario.

This stage also deals with interfacing imperative logic programs of XSB into the procedural Java programs. This is done using the Java support libraries of XSB.

Stage #3: Executive component development

After a correct service plan is generated from the KB planner, it is necessary to execute that plan in a real or virtual robot. So, in this stage the localization and navigation control routines are developed in an open source mobile robot simulator, Player/Stage and this is interfaced with Executive Supervisor.

Stage #4: Service and UI development

This stage is actually the extension of previous stage. Here Executive Supervisor is connected with HRI Panel to get user's requests and it is also connected with Service Coordinator to get the executive tasks of a service.

Stage #5: Integration of Components

After the development of all necessary components is complete, the complete scenarios are executed in this stage. In this stage the activity of all components are visualized as a whole. For example, in case of office scenario an appropriate plan is given to Executive Supervisor which executes that plan in Player/Stage simulator. On the other hand, in hospital scenario, a robot is executing the service requests of a nurse and a patient and a doctor is interrupting those services by forcing the robot to execute his request for an emergency service. In this case, all software components are working together to implement this scenario.

The description of the development stage 3, 4 and 5 is provided in Section 5.2.3 and 5.3.3 for office and hospital scenario respectively. After evaluating the KB service planning in Section 5.4, a discussion is made on the overall implementation of KB service approach in Section 5.5.

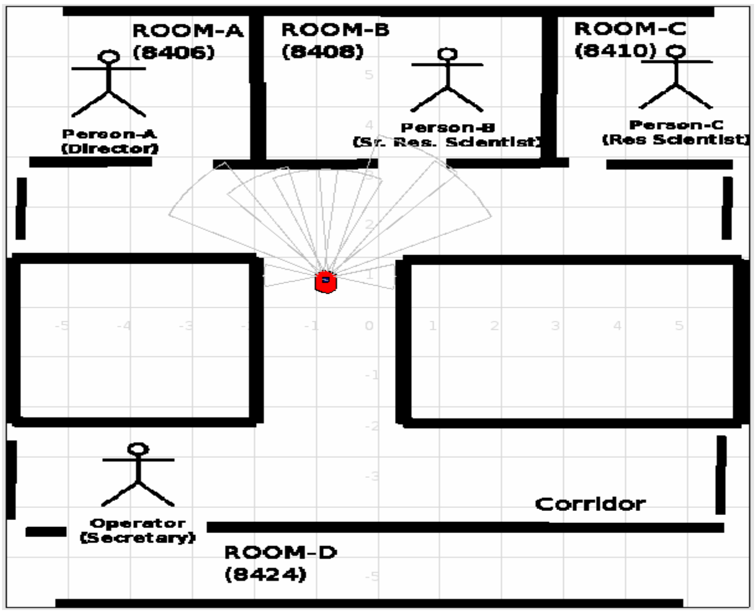


Fig. 5.2-1 A typical Stage simulator setup for an object delivery service

5.2. Scenario 1: Knowledge-Based Object Delivery Service

As initially reported by Sarker et al. (2007), this scenario has been designed as delivery of objects (e.g., mail, fax, CD, confidential document etc.), by a service robot from the office secretary's desk to different rooms of the staffs of an office (a typical set up is shown in Fig. 5.2-1). This kind of scenario has been considered by other researchers focusing mainly on system design and development (Severinson et al. 2003, Taipalus et al. 2005). However, the focus of this thesis is to demonstrate how the robot can follow human guidelines by customizing its services.

5.2.1 Design of Object Delivery Service Policies

The higher level service guidelines for this object delivery service are taken from the senior staff of the office. In this scenario, it is decided that the robot must satisfy the following three conditions:

1. Confidentiality of the delivered objects must be preserved,
2. Urgent objects must be delivered by person seniority and
3. Other regular objects, which are neither confidential nor urgent, should be delivered according to the shortest distance from the service start point to destination point.

To meet these conditions, the robot should follow the working patterns practiced by the office secretary who does this object delivery job regularly. An office secretary usually maintains the confidentiality of objects by visiting one person at a time. She also maintains the priorities of urgent objects according to the seniority of persons and if the object is neither confidential nor urgent, she sorts them out by location and visits the rooms of the staffs by choosing the nearest one first. So, the following categorized higher level policies have been derived for a robot by considering the similarities and differences of object delivery tasks with an office secretary. Such as, since the cradle of the robot where objects are placed is always visible to the recipients, an additional guideline is included for maintaining the confidentiality of objects.

1) Policies for confidential object delivery:

- In service planning phase (while receiving the user's request for object delivery) if there exist multiple instances of confidential objects for multiple persons, this request must not be processed by the robot.
- In service executing phase, when the robot is going to deliver the confidential object to a person, the person might be unavailable at that moment and the delivery can't be successful. In that case, other non-confidential (even urgent) objects of that mission will not be attempted to deliver at that moment. The robot must return to the service start point to report the failure and re-schedule the service.

2) Policies for urgent object delivery:

- Senior persons (by employment rank and employment start date) will get higher priority to receive the urgent objects.
- If an urgent object is failed to deliver for first time, it might be retried one more time after serving the remaining objects.

3) Policies for regular object delivery:

- The persons located nearer from the service start point should be served first. Also, persons in same location should be served at a time.

5.2.2 Development of Knowledge-Bases

As shown in Fig. 5.2.2-1, we have organized the knowledge-base of our object delivery service hierarchically using the modular programming strategy. Here, each file contains some facts or rules. In the deductive database literature, such as in (Bertino et al. 1994), it is common to distinguish a set of facts as the extensional database (EDB) and to refer to the set of rules as the intensional database (IDB). As we can see in Fig. 4.2.2-1, our inference tree for deriving a plan for the object delivery is rooted in three facts files. The KB planner contains EDB (e.g., object facts, person facts, room facts etc.) and IDB (remaining part of KB). EDB has static part like person facts and dynamic part like object facts. A dynamic EDB is modified to reflect the changes of service-requests in run-time whereas a static EDB is kept unchanged during service planning/execution.

The following steps have been undertaken to complete the development of our Policy-Base.

1) Finding available facts for making inferences:

At the beginning of the design process, the available unconditional facts accessible by the robot have been analyzed. When a user will make a request to the robot to deliver an object, it is natural that she will tell the robot like this: "Please deliver this *mail* to *Dr. Ali*, this is *confidential*" or perhaps in another case she might say: "Please deliver this *fax* to *Dr. Ki*m, this is *urgent*".

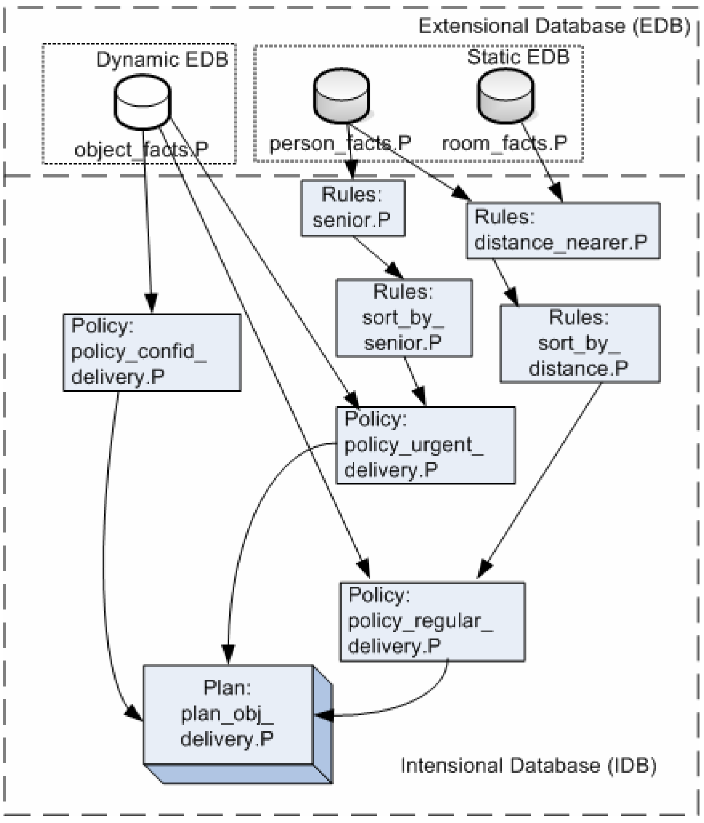

Extensional Database (EDB)
Dynamic EDB
Static EDB
object_facts.P
person_facts.P
room_facts.P
Rules: senior.P
Rules: distance_nearer.P
Rules: sort_by_ senior.P
Rules: sort_by_ distance.P
Policy: policy_confid_ delivery.P
Policy: policy_urgent_ delivery.P
Policy: policy_regular_ delivery.P
Plan: plan_obj_ delivery.P
Intensional Database (IDB)

Fig. 5.2.2-1 KB Inference tree for planning an object delivery service

By combining these cases, five major properties of an object from a run-time request of a user have been selected: object category, receiver of the object, object confidentiality, object urgency and an automatic object ID assigned for tracking it.

If this fact are cross-checked with the confidential object delivery policy, it can be seen that the collected facts are enough to make a decision to satisfy this policy (i.e., the underlying logic of inference is based on the count the number of persons who are going to receive confidential objects). A sample object facts file, *object_facts.P*, has been shown here.

```
%% object_facts.P : Run-time list of objects to be
% delivered, request supplied by the user)
:- export object/5.
% Object(Id, Category, ReceiverStaffID,
% isConfidential, isUrgent)
object(1,'Mail',ali,1,0).
object(2,'Fax',kim,0,1).
object(3,'Report',park,0,0).
```

As it can be seen here, each XSB program module exports predicates with a certain number of arity. The object predicate has been exported with an arity of 5. Later, it will be seen that other program modules will use this predicate by an import command. Any built-in predicate can also be imported from the libraries shipped with XSB.

Now, the additional database facts which are given to the robot in advance should be considered. So, the urgent object delivery policy cab be examined which says that robot has to deliver an urgent object based on the seniority of a person (by employment rank and employment start date). To satisfy this policy, some attributes of every employee are kept in the database of persons, such as a unique staff ID, full name of the person, employment rank or designation, employment start date, room number and status of attendance in office. A sample person facts file, *person_facts.P*, is shown here.

```
% persons_facts.P: persons DB (with 6 attributes:
% StaffID,Name,Designation, EmploymetStartDate,
% OfficeLocation, AttendenceStatus)
:- export person/6.
person(ali,'A. Ali','Director', 200404,8406,'Present').
person(kim, 'B. Kim', 'Sr. Res. Scientist', 200407, 8408,
'Present').
person(lee, 'C. Lee', 'Sr. Res. Scientist', 200403, 8408,
'Present').
person(park, 'X. Park', 'Res. Scientist', 199103, 8410,
'Present').
person(smith, 'Y. Smith', 'Res. Scientist', 199903, 8410,
'Present').
```

As the robot has to move from one room to next room, another database containing room facts, shown in *room_facts.P,* is created, the available rooms for the object delivery service.

```
%% room_facts.P : a sample DB of rooms under
% ObjectDelivery service
:- export room/1.
room(8406).
room(8408).
room(8410).
```

Again, it is cross checked that these files to ensure that this much information is enough for making inferences to satisfy our policies for urgent and regular object delivery cases(i.e., to find the most senior person or the nearest room from the service start point of the robot etc).

2) Constructing the inference tree by writing deductive rules:

As it is seen in Fig. 4.4-1, Strategic Planner (SP) will evaluate the KB module and it will generate a strategic plan for ES using an inference tree. So, logic program modules are incrementally written by deductive rules with a final goal for building the inference tree as shown in Fig. 5.2.2-1. The input of this tree will be the person facts and room facts extracted from relational databases of person facts and the run time object facts. The output of this tree should be the decision whether user's request is valid or not and if valid what is the correct order of delivering objects following the user specified policies. Here some examples of the KB modules are presented following the syntax of XSB's native language, Hi-Log (Sagonas et al. 1994).

As shown in Fig. 5.2.2-1, from the object facts file a decision is derived to conform the confidentiality policy in file *policy_confid_delivery.P*.

```
%% policy_confid_delivery.P
:- import length/2 from basics.
:- import object/5 from object_facts.
:-    export    confid_rcvr/1,    confid_rcvr_set/1,
confid_delivery/1.
%  Find  the  disinct  list  of  persons  who  will
receive %confidential documents
confid_rcvr(Rcvr) :-
object(_,_,Rcvr,IsConfid,_), IsConfid =:= 1.
confid_rcvr_set(List) :-
setof(Rcvr, Rcvr ^ confid_rcvr(Rcvr), List).
confid_rcvr_set([]).
% policy.confidentiality: only one person is %allowed for
confidential delivery
confid_delivery(Answer) :-
confid_rcvr_set(List),  length(List,  Len),  Len  =<  1,
Answer = 'Valid'.
confid_delivery(Answer) :-
confid_rcvr_set(List), length(List, Len), Len > 1, Answer
= 'Invalid'.
```

So if we input two files: object_facts.P and policy_confid_delivery.P to XSB engine and post a query : `? confid_delivery(Ans).`

For valid request, we can see that it says: `Ans = Valid.`

In case of implementing our policy for delivering urgent objects, the sorted list of receivers is derived by seniority rule (using employment rank and employment start date) in module *senior.P* (included in Appendix-1) and then a quick sort algorithm is used to find a list of persons arranged in the order of decreasing seniority. Some default clauses are attached to arbitrate some exceptions, e.g., "*first come first serve*" rule is set when two persons have exactly same employment ranks and same employment start dates. Then, policy_urgent_delivery.P module has been constructed.

```
%% policy_urgent_delivery.P : produces a list of
%% receivers of urgent objects by seniority.
:- import object/5 from object_facts.
:- import sort_by_senior/2 from sort_by_senior.
:- export urgent_delivery_list/1.
% Make a list of rcvr. who will rcv. urgent obj.
urgent_rcvr(Rcvr) :-
object(_,_,Rcvr,_,IsUrgent), IsUrgent =:= 1.
urgent_rcvr_set(List):-
setof(Rcvr, Rcvr^urgent_rcvr(Rcvr), List).
% Sort by seniority
urgent_delivery_list(SortedList):-
urgent_rcvr_set(RawList),    sort_by_senior(RawList,
SortedList).
urgent_delivery_list([]).
```

Similarly, in regular object delivery case, a sorted list of receivers has been made based on the nearer distance using three XSB modules: *distance_nearer.P, sort_by_ditance.P* and *policy_regular_delivery.P* module. Finally, all policy modules have been assembled to generate a plan in module *plan_obj_deliver.P* . The underlying idea behind this combining process is: the confidential delivery policy is checked first and if it succeeds then this module will make a list and add the receiver who will receive the confidential documents; otherwise it will make an empty list and incrementally add urgent and regular lists to it. Some built-in and custom predicates are used to produce an appropriate list of persons and locations. All knowledge-base modules discussed above have been included in Appendix-1.

5.2.3 Development of Software Components

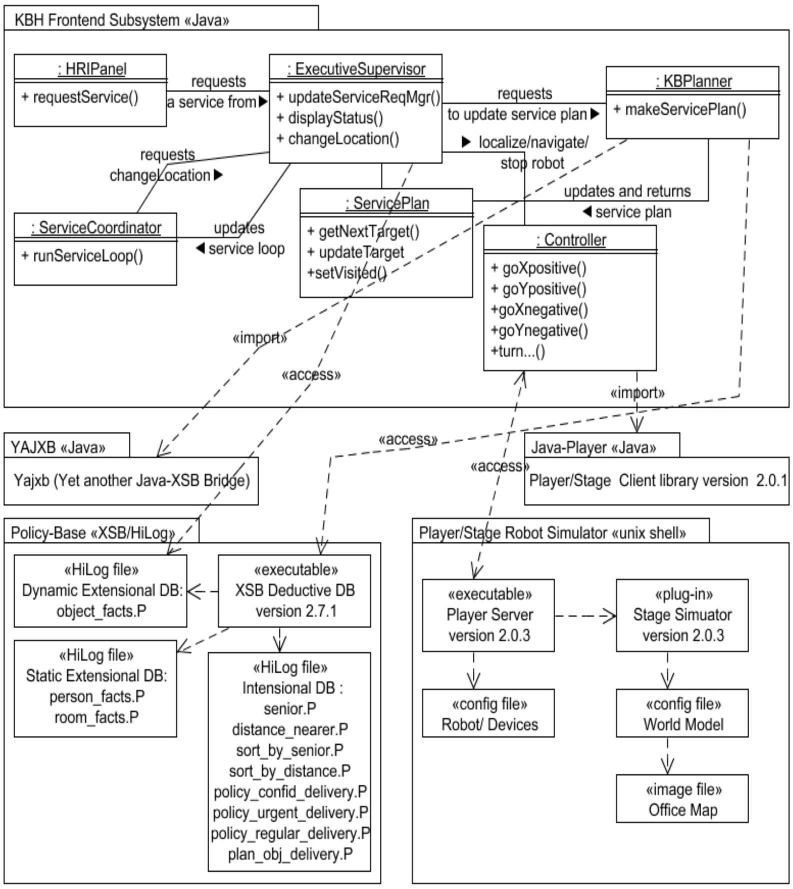


Fig. 5.2.3-1 Package diagram of KBH software components in office scenario

Fig. 5.2.3-1 shows the static configuration of entire software system. It consists of three major packages: 1) KBH Front-end Subsystem, 2) Policy-Base and 3) Player/Stage Robot Simulator. The development of Policy-Base package is already discussed in Section 5.2.2. Below the remaining two packages are highlighted.

KBH Front-end Subsystem

This package represents the major software modules of KBH software architecture. It is developed in Java under Eclipse (version 3.2) Java IDE. The class diagram in Fig 5.2.3-1 shows the relationships among the modules of KBH software architecture. For example, *HRIPanel* requests the object delivery service from *ExecutiveSupervior*. Consequently, ExecutiveSupervior requests KBPlanner to update *ServicePlan*. After that ExecutiveSupervior provides ServicePlan to S*erviceCoordinator* to run its service loop. As a result of collaborative efforts of ServiceCoordinator and ExecutiveSupervior, the *Controller* gets necessary instruction to change robot location in the office environment. In this development process, *KBPlanner* is interfaced with Policy-Base and its inference engine, XSB (an efficient deductive database from (XSB, 2007)), via a Java bridge program YAJXB obtained from (YAJXB, 2007). KBPlanner imports necessary libraries from YAJXB while accessing XSB.

Player/Stage Robot Simulator

After a brief survey on available open source and commercial mobile robot simulators, Player/Stage (Player/Stage 2007, Gerkey et al. 2003) mobile robot simulator has been selected as the simulation test bed due to the remarkable features of Player/Stage, e.g., well organized client/server framework, diverse programming language support and a large user community. Stage (version 2.0.3) 2-D simulator and Player (version 2.0.3) robot server have been used along with the Java client for Player named Java-Player (version 2.0.1) from Java-Player (2007) project. The configuration of Player/Stage is also shown in Fig 5.2.3-1. It contains necessary configuration files like world model, office maps etc. The Controller of KBH Front-end Subsystem imports Java-Player libraries to access Player server.

5.2.4 Scenario Implementation and Test Results

The scenario of delivering object in an office environment based on user-guidelines (discussed in Section 5.2.1) consists of three steps.

1. An end-user of an office such as a secretary will check the database of existing persons of her office and get the objects like mail, faxes, parcels etc. of those persons from the reception desk.
2. In the HRIPanel user interface, She will input the following object description of all objects at her desk:

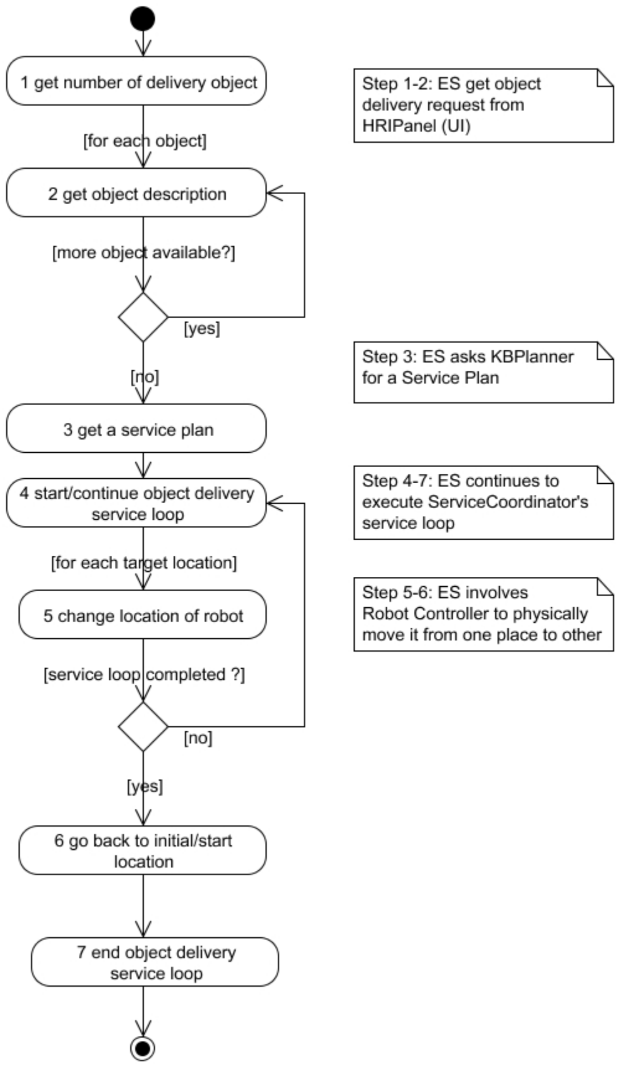
- Category of the object (e.g., Mail/Fax/CD etc.)
- Recipient of the object (e.g., Dr. Ali)
- Confidentiality of the object (e.g., yes/no)
- Urgency of the object (e.g., yes/no)

3. After getting this service request, robot will generate a object delivery plan which can be started by the secretary and wait for its completion of service. While executing service, if the robot faces any exceptions or other troubles it will report to the secretary and return to its start location. Otherwise, it will complete the delivery service, return to its start location and report status of delivery service.

The execution of above tasks has been outlined in Fig. 5.2.4-1 where the activity of ExecutiveSupervisor is explained in comment boxes. Such as collaboration of ExecutiveSupervisor with HRIPanel, KBPlanner, ServiceCoordinator and Controller in different stages are presented.

Testing and validating KB Planner

The results obtained by running our Policy-Base modules in XSB with several data sets give satisfactory performance of the KB Planner. Here, a few cases have been described with sample data sets and results. For brevity export commands and some optional lines are removed from these examples.



1 get number of delivery object
[for each object]
2 get object description
[more object available?]
[yes]
[no]
3 get a service plan
4 start/continue object delivery service loop
[for each target location]
5 change location of robot
[service loop completed ?]
[no]
[yes]
6 go back to initial/start location
7 end object delivery service loop
Step 1-2: ES get object delivery request from HRIPanel (UI)
Step 3: ES asks KBPlanner for a Service Plan
Step 4-7: ES continues to execute ServiceCoordinator's service loop
Step 5-6: ES involves Robot Controller to physically move it from one place to other

Fig. 5.2.4-1 Activity diagram of ExecutiveSupervisor while executing object delivery scenario

a) Case-1a: In this case, the user has requested our robot to deliver multiple confidential objects to multiple receivers.

```
%Object(Id,Category,Receiver,isConfid,isUrgent)
object(1,'Mail',kim,1,0).
object(2,'Fax',ali,1,0).
object(3,'Report',lee,1,1).
object(4,'Notes',kim,0,0).
```

The resulted plan from KB planner is saved in a file and is shown here:

```
Error in your delivery request!
Too many confidential receivers: [kim,lee,ali]
Please select only one receiver for your request.
```

b) Case-1b: The input fact file and output result for multiple confidential objects to one receiver are shown here.

```
%Object(Id,Category,Receiver,isConfid,isUrgent)
object(1,'Mail',kim,1,0).
object(2,'Notes',kim,1,0).
```

```
I am going to process your request with this plan-
Persons: [kim]
Locations: [8408]
```

c) Case-2: Now, the user has requested multiple urgent objects to multiple persons.

```
%Object(Id,Category,Receiver,isConfid,isUrgent)
object(1,'Mail',kim,0,1).
object(2,'Notes',ali,0,1).
object(3,'Mail',lee,0,1).
object(4,'Notes',smith,0,1).
object(5,'CD',park,0,1).
```

So, KB engine gives this answer according to seniority rule.

```
I am going to process your request with this plan-
Persons: [ali,lee,kim,park,smith]
Locations: [8406,8408,8410]
```

d) Case-3:

Here, all kinds of objects are mixed in the user’s request.

```
%Object(Id,Category,Receiver,isConfid,isUrgent)
object(1,'Notes',smith,0,0).
object(2,'Notes',ali,0,1).
object(3,'Mail',kim,0,1).
object(4,'Fax',lee,1,0).
object(5,'Fax',park,0,0).
object(6,'Mail',lee,1,0).
```

The corresponding answer is shown below.

```
I am going to process your request with this plan-
Persons: [ali,lee,kim,park,smith]
Locations: [8406,8408,8410]
```

Simulation in Player/Stage

The above plans have been carried out by our Player/Stage robot in an environment shown in Fig. 5.2-1. Here, it is observed that the HC robot can properly utilize its knowledge of certain user relationships and policies to derive a proper plan for its object delivery service. In case of the traditional service planning, a robot can not achieve this kind of plan by using simple and unrelated location and/or person information. However, using relationship information from a knowledge-base it can easily deduce an executive service plan that satisfies user policies. If the robot is working in a human-living environment, this kind of behavior of robot is highly expected by the end-users.

5.3. Knowledge-Based Patient Care Service

In this knowledge-based patient care service scenario, three kinds of services are selected to be performed by the hospital care robot as shown in Fig. 5.3-1:

1) Guiding patients,
2) Delivery of medicine and
3) Providing information.

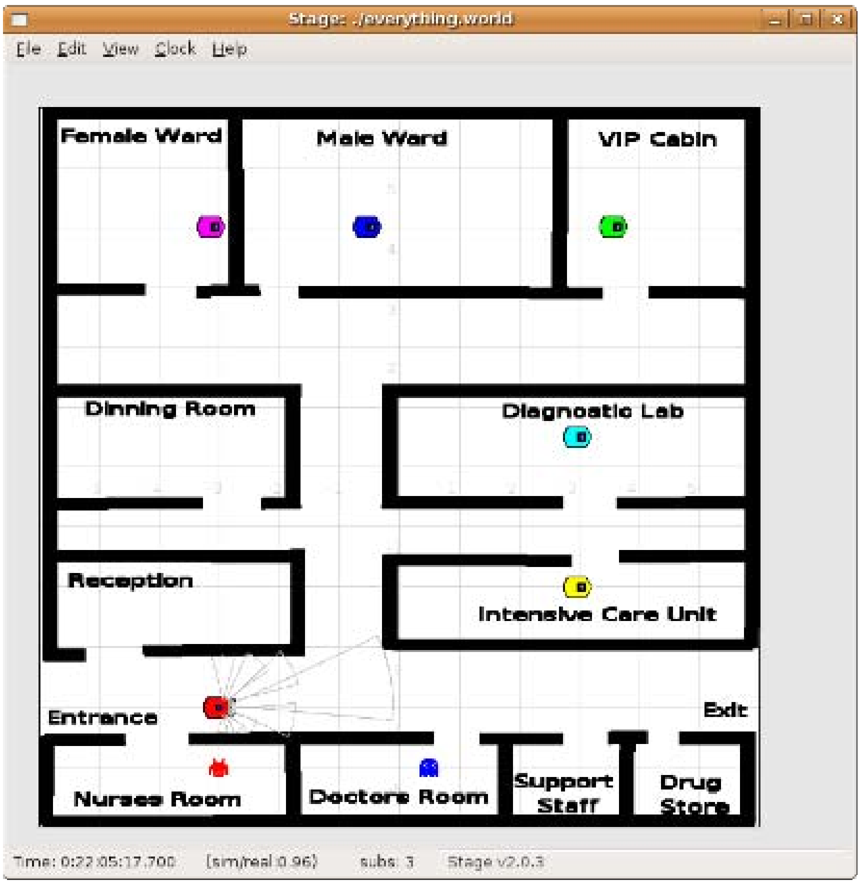


Fig. 5.3-1 A typical layout of a hospital in Stage simulator for the patient-care service scenario

1) **Guiding patients:** In hospital environment, it is very common to move a patient from one place to another place, such as from reception desk to doctor's room, from a patient ward to diagnostic room, from a ward to recreation room and so on. In this service, determining priority is most important. Priority of a service is also determined according to the type of users. For example, an emergency patient must get high priority for accessing this service over a non-emergency patient.

2) **Delivery of medicine:** One of most important tasks of nurses in hospital is to deliver medicine in time to the patients in wards. This task needs both careful attention and timely completion. In this service, some patients might expect high priority over others like senior and elderly patients might be served first according to the norm of the hospitals.

3) **Providing information:** Every patient in hospital needs to send and receive information from his/her outside relatives and visitors. Although this is a less important service, sometimes it might be very useful for some patients.

5.3.1 Design of Service Policies

In hospital environment, it is common to enforce a set of policies for providing various services to various types of patients. Most common example is to provide urgent services to emergency patients. Others include timed services like timely delivery of medicine, on demand services like guide patients from one place to another place, reservation services and so on. To access the above services an example policy has been proposed. Based on the three major kinds of users, i.e., doctors, patients and nurses, this policy has been declared.

Policy for user priority:

The priority of users for accessing a service provided by the robot must be maintained according to the user priority policy regardless of the type of services. This implies a high priority user's request can preempt a service requested by a low priority of user. By default, a doctor will have higher priority over a nurse and a nurse will have higher priority over a patient. Additional priority level among each user category can also be mentioned by additional rules under this policy, e.g., an emergency patient will have higher priority over a regular patient, a senior nurse will get higher priority over a trainee nurse and so on.

5.3.2 Development of Knowledge-Bases

As shown in Fig. 5.3.2-1, we have organized the knowledge-base of our object delivery service hierarchically using the few XSB modules. The following steps, similar to our previous KB object delivery case, have been undertaken to complete the development of our KB. This tree is relatively simple as it deals with only a single policy and works in dynamic situations.

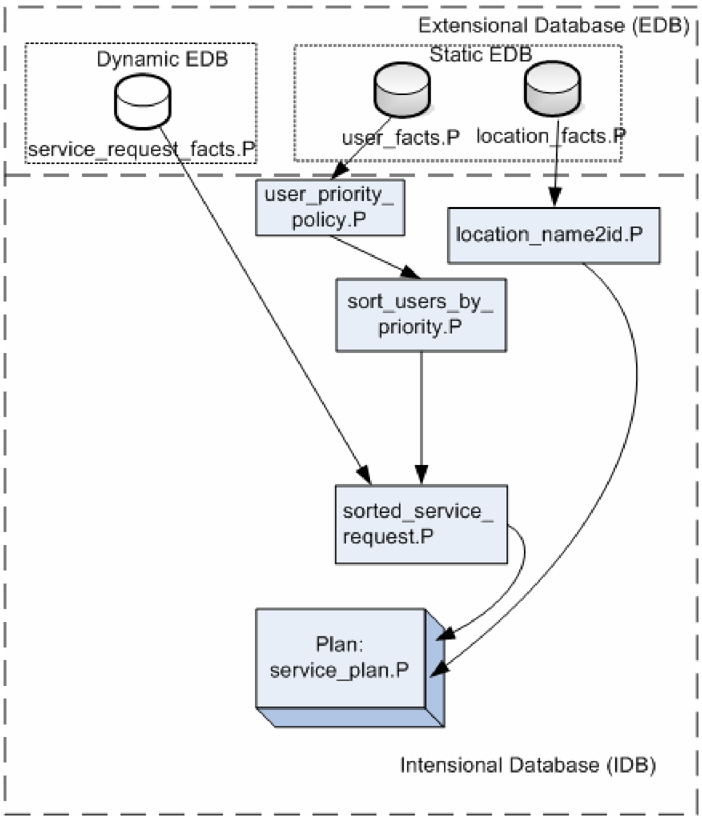


Fig. 5.3.2-1 KB Inference tree for planning a patient-care service

1) Finding available facts for making inference:

At first, a fact file (treated as a dynamic EDB), *service_request_facts.P*, is prepared that contains the service request done by the users. The format of a service request is defined as: a request id, name of service, id of the user who requests the service, id of the user (usually patient) who will receive the service and additional parameter like start location and destination location. A sample fact file is shown here.

```
%% service_request_facts.P : Runtime request of service
% Attributes: RequestId, Name, RequestedBy,
% RequestedFor, FromLoc, ToLoc,...
:- export service_request/6.

% d01 is requesting to guide p03 from west corridor left
% to Male Ward
service_request(1, guidePatient, p01, p03,
westCorridorLeft, maleWard).
service_request(2, deliverMedicine, p02, p01,
nursesRoom, femaleWard).
service_request(3, provideInfo, p04, p04,
maleWard, nursesRoom).
service_request(4, deliverMedicine, n01, p03,
nursesRoom, maleWard).
```

The KB patient care service contains two major static EDB: 1) user facts file (*user_facts.P*) and 2) location facts file (*location_facts.P*). In user facts file, the available users of the robot is listed with their five necessary attributes (ID, name, category, designation/class, office location). An example of user facts file is presented here.

```
%% user_facts.P : User DB
% Attributes: ID, Name, Category, Designation/Class,
OfficeLocation
:- export user/5.

user(d01,'A. Matin',doctor,'Medical Officer',
doctorsRoom).
user(n01, 'A. Mim', nurse, 'Senior Nurse', nursesRoom).
user(p01, 'A. Kim', patient, 'Emergency Patient',
intensiveCareUnit).
user(p02, 'B. Park', patient, 'Regular Patient',
femaleWard).
user(p03, 'C. Lee', patient, 'VIP Patient', vipCabin).
```

In location facts file, the symbolic names of the available locations are stored for easy reference in KB modules, as presented here.

```
%% location_facts.P: DB of location aliases

:- export location/1.

%% Rooms
location(nursesRoom).
location(doctorsRoom).
location(supportRoom).
location(drugStore).
location(receptionRoom).
location(intensiveCareUnit).
location(dinningRoom).
location(diagnosticLab).
location(femaleWard).
location(maleWard).
location(vipCabin).
%% Corridor left, middle, right
location(westCorridorLeft).
% aliased as:
location(hospitalEntrance).
location(westCorridorMiddle).
location(westCorridorRigt).
% aliased as:
location(hospitalExit).
```

For convenience, these symbolic location names have been converted to numbers in *location_name2id.P* module. These EDB files provide necessary information for the inference process as described here.

2) Constructing the inference tree by writing deductive rules

The heart of the intensional DB (IDB) files is the user priority policy file (*user_priority_policy.P*, attached in Appendix-2) which defines the role of users and their priorities. As stated in the user priority policy in Section 5.3.1, higher priority is assigned to doctors, then nurses and then patients. In every type of users another subclass may also be created. For example, patients can be classified as emergency, regular or very important person

(VIP) patients. Depending on the user needs these classifications can be done.

Based on the user priority policy, additional rules have been developed in *sort_users_by_priority.P* and *sorted_service_request.P* modules. Finally, service plan module extracts a prioritized service plan in module, *service_plan.P*, as shown below.

```
:- import sorted_requesters/1, requester2cmdLoc/2 from
sorted_service_request.
:- import location_id/2 from location_name2id.
:- export service_plan/1.

service_plan_string(LocList):-
    sorted_requesters(RequesterList),
    requester2cmdLoc(RequesterList, LocList).
loc_list_convert([], []).
loc_list_convert([LocListName|Tr], [LocListId|Tl]):-
    location_id(LocListName, LocListId),
loc_list_convert(Tr,Tl).
service_plan(LocListId) :-
    service_plan_string(LocListName),
    loc_list_convert(LocListName, LocListId).
```

5.3.3 Development of Software Components

As shown in Fig. 5.3.3-1, the software package configuration for hospital scenario is basically similar with the previous configuration of office scenario (shown in Fig. 5.2.3-1) except a few things. Firstly, in ServiceCoordinator of KBH Front-end Subsystem, deliver medicine service is derived from deliver object service and additional two services: guide patient and provide info, are added. Secondly, the Policy-Base reflects the hospital service policies and knowledge of hospital environment. Thirdly, the device configuration of service robot of Player/Stage simulator has been adjusted and Stage loads the map of hospital environment.

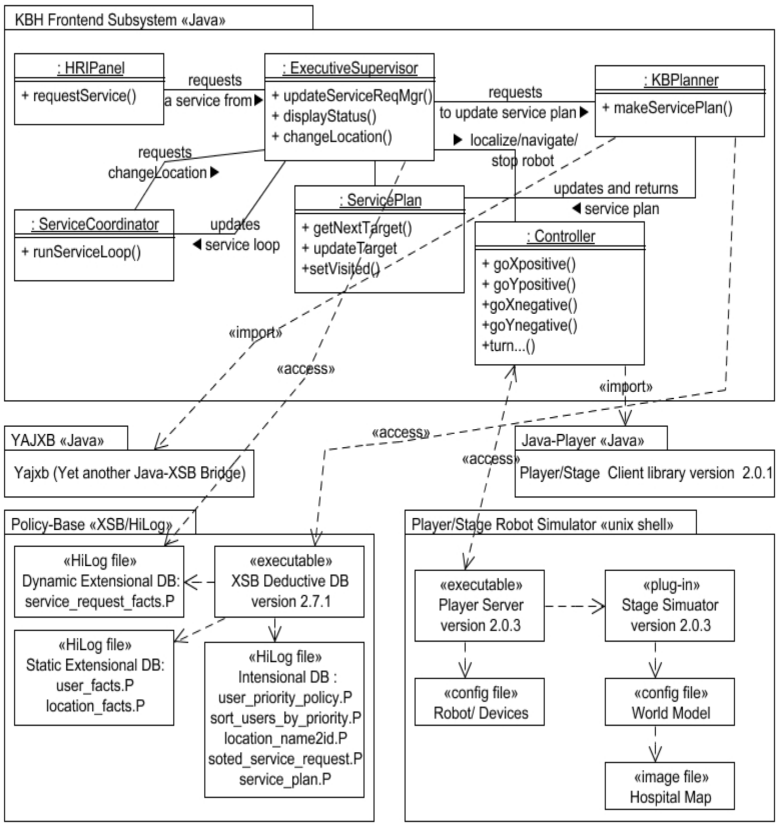

KBH Frontend Subsystem «Java»
: HRIPanel
+ requestService()
requests a service from ▶
: ExecutiveSupervisor
+ updateServiceReqMgr()
+ displayStatus()
+ changeLocation()
requests to update service plan ▶
: KBPlanner
+ makeServicePlan()
requests changeLocation ▶
▶ localize/navigate/ stop robot
updates and returns ◀ service plan
: ServiceCoordinator
+ runServiceLoop()
updates ◀ service loop
: ServicePlan
+ getNextTarget()
+ updateTarget
+setVisited()
: Controller
+ goXpositive()
+ goYpositive()
+goXnegative()
+goYnegative()
+turn...()
«import»
«access»
«import»
«access»
«access»
YAJXB «Java»
Yajxb (Yet another Java-XSB Bridge)
Java-Player «Java»
Player/Stage Client library version 2.0.1
Policy-Base «XSB/HiLog»
«HiLog file» Dynamic Extensional DB: service_request_facts.P
«executable» XSB Deductive DB version 2.7.1
«HiLog file» Static Extensional DB: user_facts.P location_facts.P
«HiLog file» Intensional DB : user_priority_policy.P sort_users_by_priority.P location_name2id.P soted_service_request.P service_plan.P
Player/Stage Robot Simulator «unix shell»
«executable» Player Server version 2.0.3
«plug-in» Stage Simuator version 2.0.3
«config file» Robot/ Devices
«config file» World Model
«image file» Hospital Map

Fig. 5.3.3-1 Package diagram of KBH software components in hospital scenario

In order to simulate dynamic events in hospital scenario, Java multi-threads have been used to deploy the various functions of Executive Supervisor. As shown in Fig. 5.3.3-2, the activity diagram of Executive Supervisor, four major threads are working together to manage the responsibilities of Executive Supervisor.

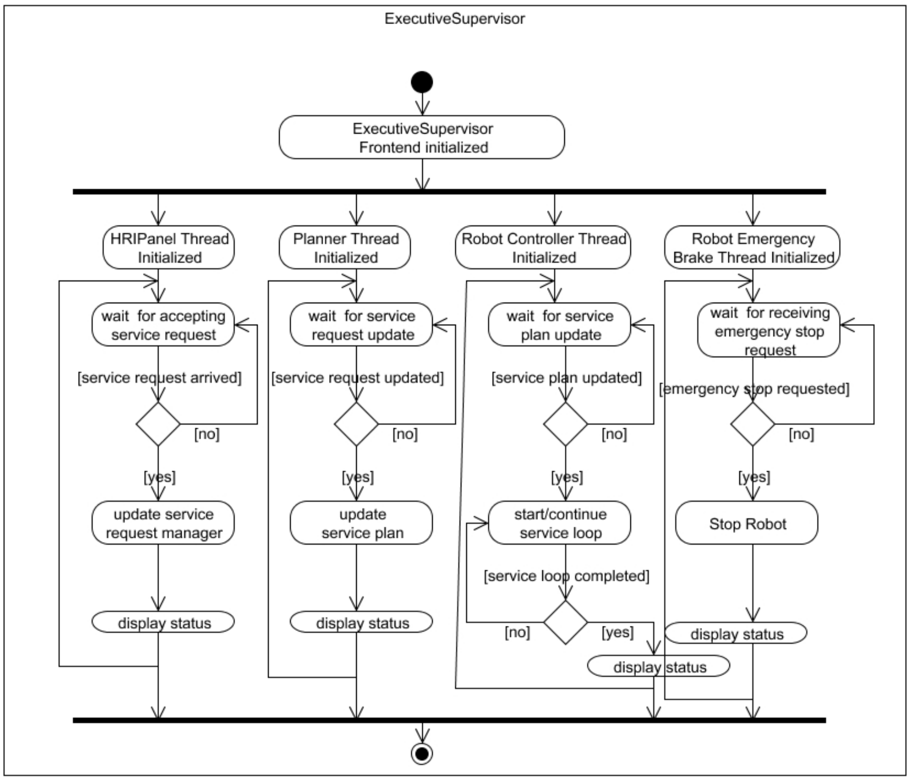

ExecutiveSupervisor
ExecutiveSupervisor Frontend initialized
HRIPanel Thread Initialized
Planner Thread Initialized
Robot Controller Thread Initialized
Robot Emergency Brake Thread Initialized
wait for accepting service request
wait for service request update
wait for service plan update
wait for receiving emergency stop request
[service request arrived]
[service request updated]
[service plan updated]
[emergency stop requested]
[no]
[yes]
update service request manager
update service plan
start/continue service loop
Stop Robot
[service loop completed]
display status

Fig. 5.3.3-2 Activity diagram of ExecutiveSupervisor

HRI Panel Thread: This thread interfaces Executive Supervisor with HRIPanel. The primary responsibility of this thread is to receive user request for a service and update service request manager.

Planer Thread: KBPlanner is interfaced with Executive Supervisor by this thread. Upon continuously receiving a update of service request, KBPlanner alters service plan according to the user guidelines in Policy-Base.

Robot Controller Thread: This thread controls the execution of a service-request by interfacing with Service Coordinator and Controller.

Upon receiving request for changing location from Service Coordinator, this thread manages Controller so that robot can move around the physical environment.

Robot Emergency Brake Thread: This thread acts as an emergency control circuit to the Controller. Upon receiving a request for emergency stop, it stops the robot. This stopping can be done to avoid collision with a moving person/ others in the environment. In some exceptional cases, this can also be used to stop the robot.

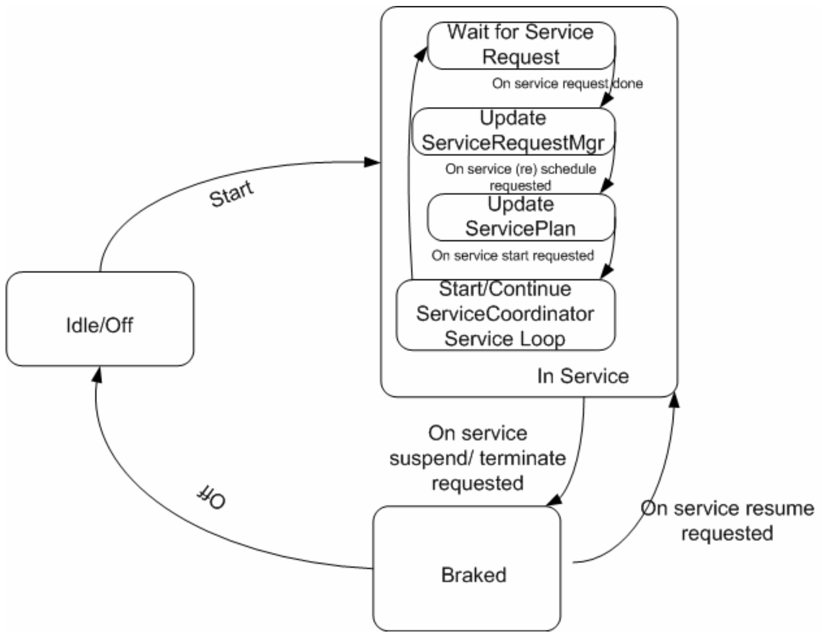

Wait for Service Request
On service request done
Update ServiceRequestMgr
On service (re) schedule requested
Update ServicePlan
On service start requested
Start/Continue ServiceCoordinator Service Loop
In Service
Start
Idle/Off
On service suspend/ terminate requested
Off
On service resume requested
Braked

Fig. 5.3.3-3 State diagram of ExecutiveSupervisor

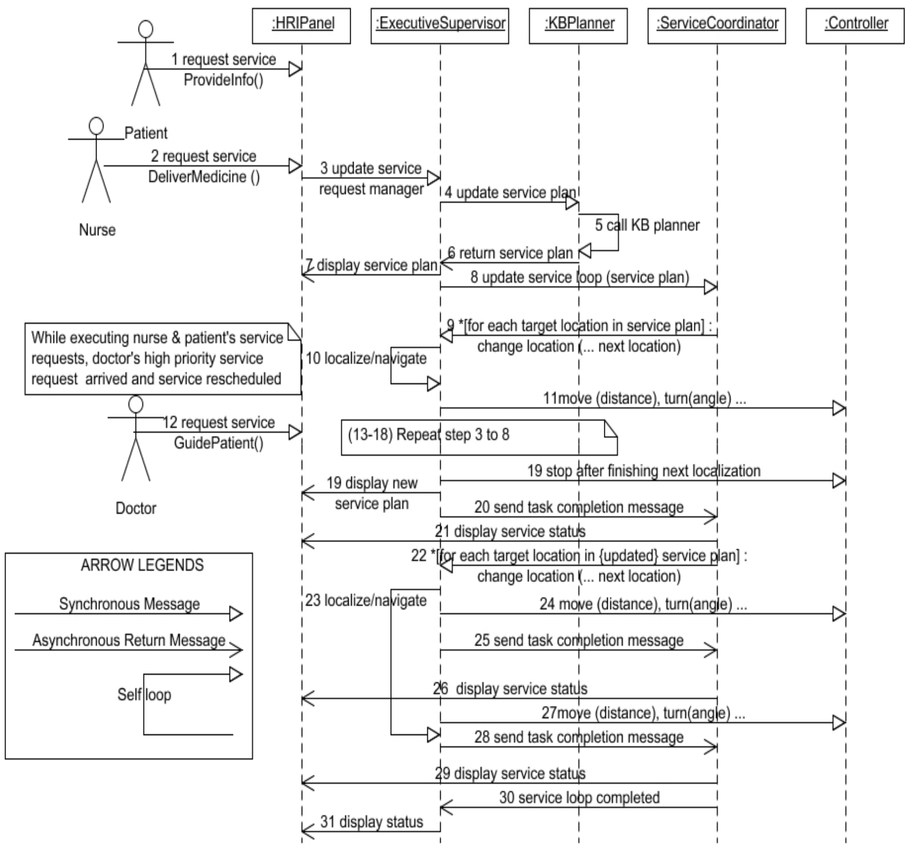
The first three threads described above work in chained fashion. Java's built-in thread synchronization functions are used to avoid object locks and other issues in multi-threading programs.

The working states of Executive Supervisor are described in Fig. 5.3.3-3. The major states of Executive Supervisor are: *Idle/Off*, *In Service* and *Braked*. When Executive Supervisor is activated from the initial state, it starts all four threads waiting for various events and entered into the *In Service* state. Inside this state, the first three threads described above manages the service activities. If an exception event occurs or an explicit user-request for suspend is generated, Executive Supervisor goes into the *Braked* state from where it can again resume to In Service state or end up in Idle/Off state. Usually, upon terminating the activities of Executive Supervisor, it at first goes into Braked state to ensure a safe shut-down of robot and to prevent any possible hazard. Later it ends up in Idle/Off state.

5.3.4 Scenario Implementation and Test Results

As described in Section 5.1, the purpose of the hospital scenario is primarily to show the dynamic re-planning of service schedule of robot under changing contexts. The detail illustration of the scenario sequences is presented in Fig. 5.3.4-1 and in Table 5.3.4-1. This scenario consists of two main phases. Below a brief description of the scenario is provided following the layout of a hospital as shown in Fig. 5.3-1.

Phase-1: Initial Service Scheduling: At the beginning of this scenario, a patient is requesting for providing-information service to another patient (from Female Ward to Male Ward) and after that a nurse is also requesting for a deliver-medicine service (from Nurses Room to Female Ward). According to the user policies in Policy-Base, a service-schedule is generated that says that robot will first move from Nurses Room to Female ward and then Female Ward to Male Ward. Since the stating location of robot is Nurses Room, robot takes medicine from nurse and starts moving to Female Ward.



:HRIPanel
:ExecutiveSupervisor
:KBPlanner
:ServiceCoordinator
:Controller
1 request service ProvideInfo()
Patient
2 request service DeliverMedicine ()
3 update service request manager
4 update service plan
Nurse
5 call KB planner
6 return service plan
7 display service plan
8 update service loop (service plan)
9 *[for each target location in service plan] : change location (... next location)
While executing nurse & patient's service requests, doctor's high priority service request arrived and service rescheduled
10 localize/navigate
11move (distance), turn(angle) ...
12 request service GuidePatient()
(13-18) Repeat step 3 to 8
19 stop after finishing next localization
19 display new service plan
Doctor
20 send task completion message
21 display service status
ARROW LEGENDS
22 *[for each target location in {updated} service plan] : change location (... next location)
Synchronous Message
23 localize/navigate
24 move (distance), turn(angle) ...
Asynchronous Return Message
25 send task completion message
Self loop
26 display service status
27move (distance), turn(angle) ...
28 send task completion message
29 display service status
30 service loop completed
31 display status

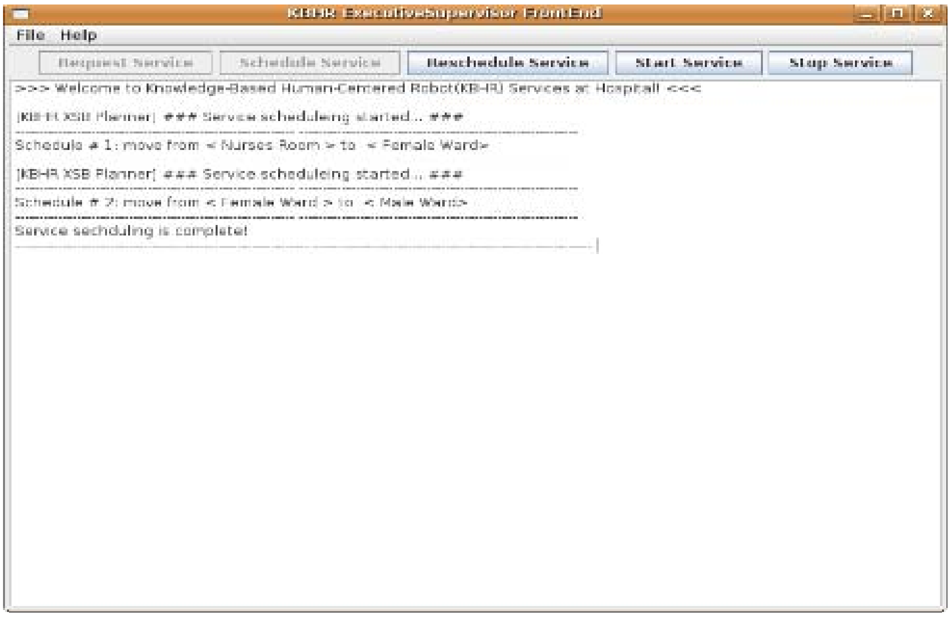
Fig. 5.3.4-1 Sequence Diagram of KB patient care scenario in hospital

Phase-2: Dynamic Re-planning of Service Schedule

But what should be done by the robot if a doctor urgently requests guide patient service while completing the nurse's request? Definitely, KB planner should support the dynamic re-planning, i.e. in this case the patient's low-priority request should be suspended and it should schedule the doctor's high-priority request immediately. In this scenario, while the robot is executing at the nurses request, a doctor urgently requests the robot to come to the Doctors Room and guide a patient to the VIP Cabin. Immediately, KBPlanner reschedules the service and now the next target location becomes Doctors Room. Since, in this scenario the localization capability of the robot is limited, the robot is allowed to complete next localization and then it responds to the new schedule. Therefore, when the robot reaches at Female Ward, it localizes itself and moves to Doctors Room (based on new service schedule) instead of moving to Male Ward (based on first service schedule). When the doctor’s service-request is completed (i.e., robot reaches at VIP cabin), it resumes its remaining tasks of first service schedule or serving the patient's request. So, it goes to Female Ward and takes some notes/letter (information) and moves to Male Ward to provide that information. At the end of this service loop it returns to the start location i.e., in front of Nurses Room. The major events of this scenario are listed in Table 5.3.4-1.

Table 5.3.4-1 Event timing of hospital service scenario in Player/Stage

Event Description	Sequence # in Sequence Diagram (Fig. 5.3.4-1)	Sample clock time in Stage upon event completion (hh:mm:ss)
Player/Stage simulator initialized	-	00:00:02
Service requests of a patient and a nurse are received	3	00:00:35
Initial service plan generated	6	00:00:37
Execution of nurse's request (service # 1) started	11	00:00:40
Doctor's service request arrived and service rescheduled	18	00:01:50
Service #1 completed	21	00:02:15
Execution of doctor's request (service # 2) started	24	00:02:20
Service # 2 completed	26	00:04:02
Execution of patient's request (service # 3) started	27	00:04:07
Service # 3 completed	29	00:06:51



```
>>> Welcome to Knowledge-Based Human-Centered Robot(KBHR) Services at Hospital! <<<

[KBHR XSB Planner] ### Service scheduleing started... ###
---------------------------------------------------------------------------
Schedule # 1: move from < Nurses Room > to  < Female Ward>

[KBHR XSB Planner] ### Service scheduleing started... ###
---------------------------------------------------------------------------
Schedule # 2: move from < Female Ward > to  < Male Ward>
---------------------------------------------------------------------------
Service sechduling is complete!
---------------------------------------------------------------------------
```

Fig. 5.3.4-2 KBH Robot Executive Supervisor Front-end showing the initial service schedule

User Interface for Executing Hospital Scenario

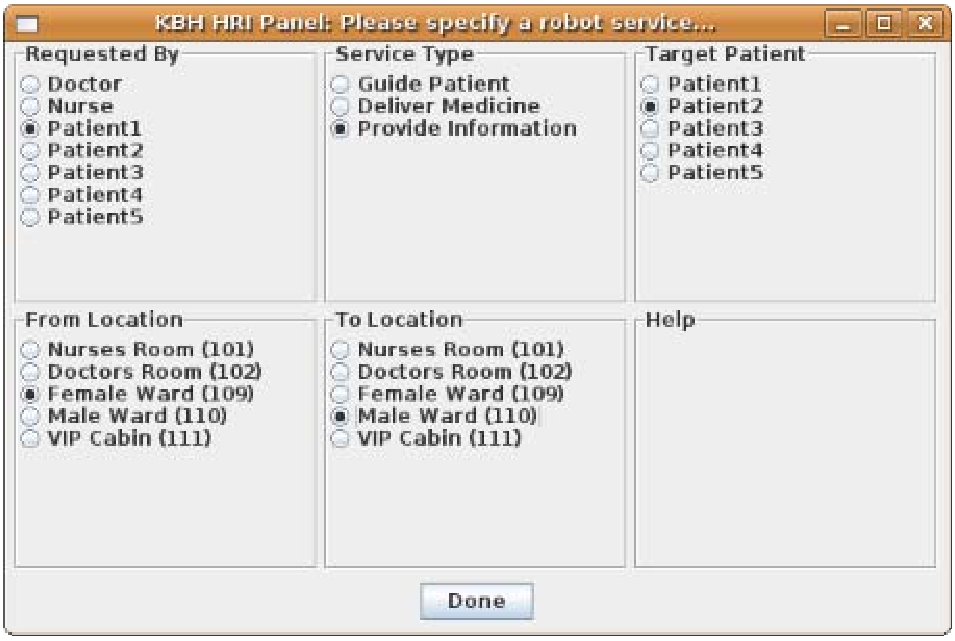
The execution of this scenario is user action or mouse click driven. Two GUI windows (as found in the screen-shots) enable a scenario-executing-user to complete this scenario. In the beginning of the scenario execution, KBH Robot Executive Supervisor Front-end window (Fig. 5.3.4-2) gives the scenario-executing-user 5 buttons with distinct activity options. Upon clicking the Request Service button, a HRI panel GUI window (Fig. 5.3.4-3) pops up. It gives several radio button options to choose requesting user type (doctor/nurse/patient), service type (guide patient/ deliver medicine/ provide info), target patient (five patients are labeled as pateient1, patient2, ..., patient5), from location (five location are labeled according to the layout of hospital) , and to location (same as from location option). After selecting the intended options, the scenario executing user can click Done button at the bottom of HRIPanel. It updates the ExecutiveSupervisor (i.e., service request manager routines) about the service requests. Any time scenario-executing-use can place a request and click Done button to get this effect.

The second button in KBH Robot Executive Supervisor Front-end window, Schedule Service, activates KBPlanner to generate a new service plan. All outputs come in the text area of KBH Robot Executive Supervisor Front-end window. Similarly, the third button, Reschedule Service also forces KBPlanner to consider newly incoming service requests and to generate a new service plan with merging the outstanding tasks of the last service plan. The Start Service and Stop Service buttons are provided to start /stop the robot in the Player/Stage simulator.

Scenario Events

The step by step real-time execution of this scenario is presented with the following actions of a scenario-executing-user along with several screen-shots.

Step 1: Scenario-executing-user clicks on the Request Service button in KBH Robot Executive Supervisor Front-end window and specify two service requests by selecting options in HRIPanel window and Done button: first one as a patient and second one as a nurse (Fig. 5.3.4-3 and Fig. 5.3.4-4).



KBH HRI Panel: Please specify a robot service...

Requested By
- Doctor
- Nurse
- [✓ Patient1]
- Patient2
- Patient3
- Patient4
- Patient5

Service Type
- Guide Patient
- Deliver Medicine
- [✓ Provide Information]

Target Patient
- Patient1
- [✓ Patient2]
- Patient3
- Patient4
- Patient5

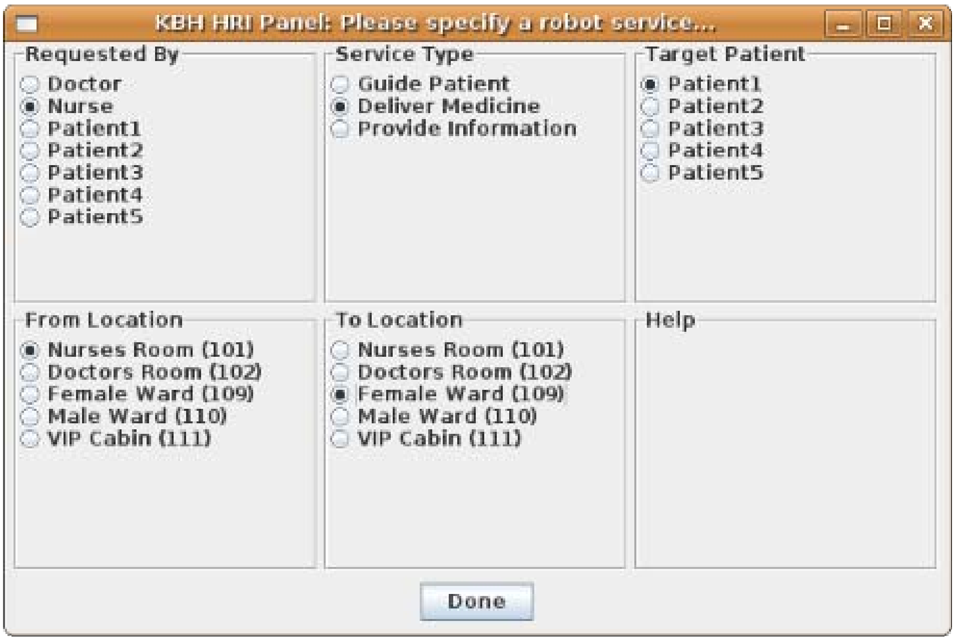
From Location
- Nurses Room (101)
- Doctors Room (102)
- [✓ Female Ward (109)]
- Male Ward (110)
- VIP Cabin (111)

To Location
- Nurses Room (101)
- Doctors Room (102)
- Female Ward (109)
- [✓ Male Ward (110)]
- VIP Cabin (111)

Help

Done

Fig. 5.3.4-3 KBH HRI-Panel capturing a patient's request for a service



KBH HRI Panel: Please specify a robot service...

Requested By
- Doctor
- [✓ Nurse]
- Patient1
- Patient2
- Patient3
- Patient4
- Patient5

Service Type
- Guide Patient
- [✓ Deliver Medicine]
- Provide Information

Target Patient
- [✓ Patient1]
- Patient2
- Patient3
- Patient4
- Patient5

From Location
- [✓ Nurses Room (101)]
- Doctors Room (102)
- Female Ward (109)
- Male Ward (110)
- VIP Cabin (111)

To Location
- Nurses Room (101)
- Doctors Room (102)
- [✓ Female Ward (109)]
- Male Ward (110)
- VIP Cabin (111)

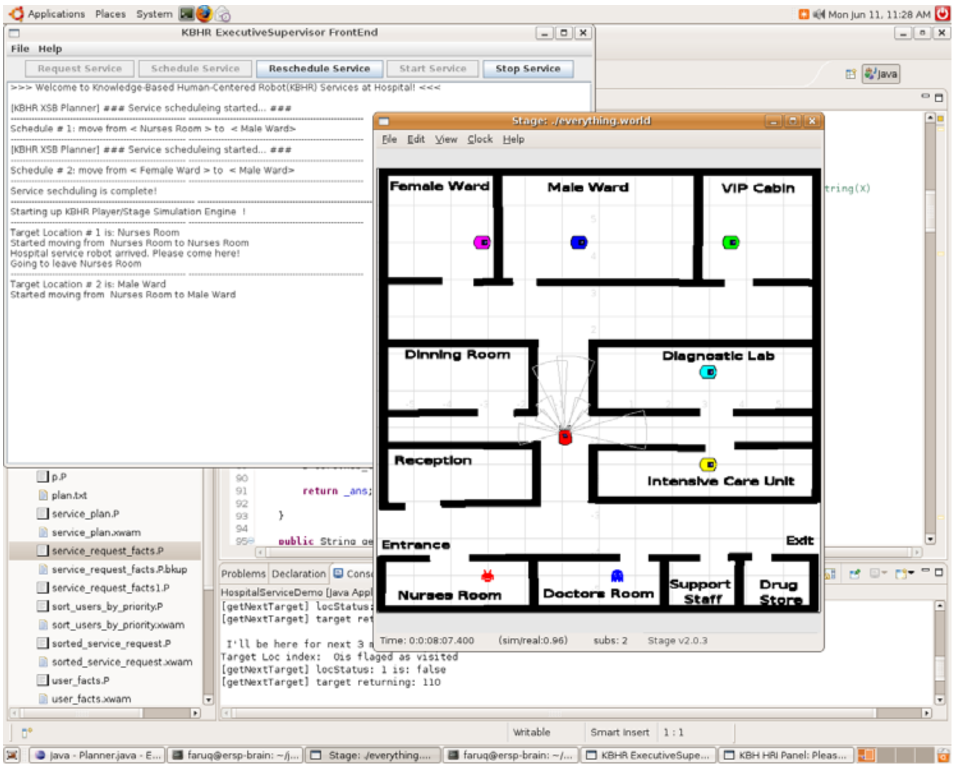
Help

Done

Fig. 5.3.4-4 KBH HRI-Panel capturing a nurse's request for a service

Step 2: Scenario-executing-user clicks on the Schedule Service button in KBH Robot Executive Supervisor Front-end window and view the initial service plan (Fig. 5.3.4-2)

Step 3: Scenario-executing-user clicks on the Start Service button in KBH Robot Executive Supervisor Front-end window and see the robot's movement in Stage window (Fig. 5.3.4-5).

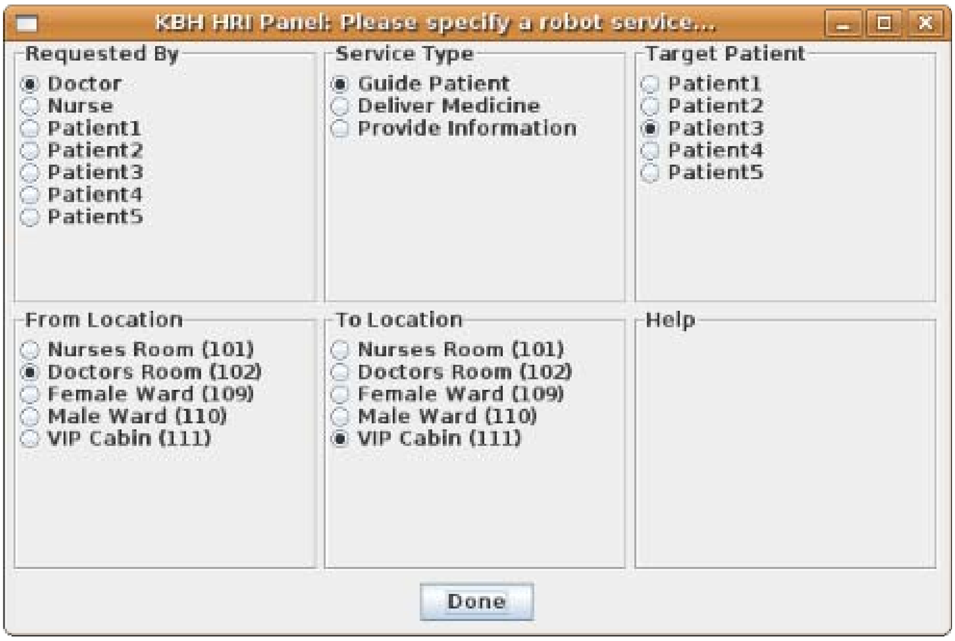


```
>>> Welcome to Knowledge-Based Human-Centered Robot(KBHR) Services at Hospital! <<<

[KBHR XSB Planner] ### Service scheduleing started... ###
-------------------------------------------------------------
Schedule # 1: move from < Nurses Room > to  < Male Ward>
-------------------------------------------------------------
[KBHR XSB Planner] ### Service scheduleing started... ###
-------------------------------------------------------------
Schedule # 2: move from < Female Ward > to  < Male Ward>
-------------------------------------------------------------
Service sechduling is complete!
-------------------------------------------------------------
Starting up KBHR Player/Stage Simulation Engine  !
-------------------------------------------------------------
Target Location # 1 is: Nurses Room
Started moving from  Nurses Room to Nurses Room
Hospital service robot arrived. Please come here!
Going to leave Nurses Room
-------------------------------------------------------------
Target Location # 2 is: Male Ward
Started moving from  Nurses Room to Male Ward
```

```
HospitalServiceDemo [Java Appl
[getNextTarget] locStatus:
[getNextTarget] target ret

 I'll be here for next 3
Target Loc index:  0is flaged as visited
[getNextTarget] locStatus: 1 is: false
[getNextTarget] target returning: 110
```

Fig. 5.3.4-5 Screen-shot captured during executing first scheduled service

Step 4: Now scenario-executing-user specifies another service request in HRIPanel window as a doctor (Fig. 5.3.4-6)



Requested By: [✓ Doctor] Nurse Patient1 Patient2 Patient3 Patient4 Patient5

Service Type: [✓ Guide Patient] Deliver Medicine Provide Information

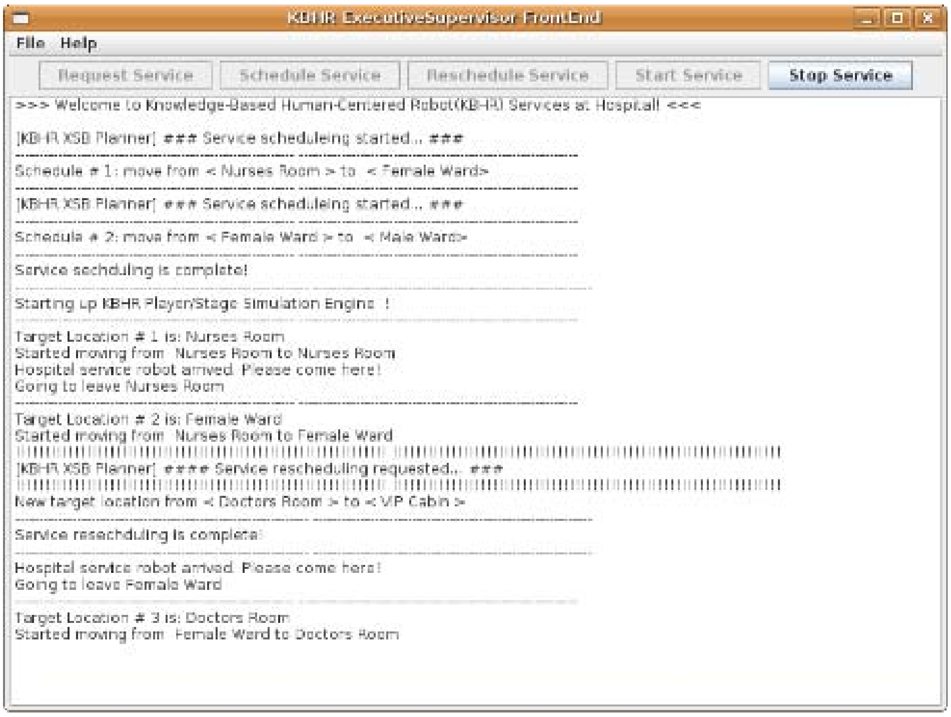
Target Patient: Patient1 Patient2 [✓ Patient3] Patient4 Patient5

From Location: Nurses Room (101) [✓ Doctors Room (102)] Female Ward (109) Male Ward (110) VIP Cabin (111)

To Location: Nurses Room (101) Doctors Room (102) Female Ward (109) Male Ward (110) [✓ VIP Cabin (111)]

Help

Fig. 5.3.4-6 KBH HRI-Panel capturing a doctor's request for a service



```
>>> Welcome to Knowledge-Based Human-Centered Robot(KBHR) Services at Hospital! <<<

[KBHR XSB Planner] ### Service scheduleing started... ###
--------------------------------------------------------------------------------
Schedule # 1: move from < Nurses Room > to  < Female Ward>
--------------------------------------------------------------------------------
[KBHR XSB Planner] ### Service scheduleing started... ###
--------------------------------------------------------------------------------
Schedule # 2: move from < Female Ward > to  < Male Ward>
--------------------------------------------------------------------------------
Service sechduling is complete!
--------------------------------------------------------------------------------
Starting up KBHR Player/Stage Simulation Engine  !
--------------------------------------------------------------------------------
Target Location # 1 is: Nurses Room
Started moving from  Nurses Room to Nurses Room
Hospital service robot arrived. Please come here!
Going to leave Nurses Room
--------------------------------------------------------------------------------
Target Location # 2 is: Female Ward
Started moving from  Nurses Room to Female Ward
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
[KBHR XSB Planner] #### Service rescheduling requested...  ###
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
New target location from < Doctors Room > to < VIP Cabin >
--------------------------------------------------------------------------------
Service resechduling is complete!
--------------------------------------------------------------------------------
Hospital service robot arrived. Please come here!
Going to leave Female Ward
--------------------------------------------------------------------------------
Target Location # 3 is: Doctors Room
Started moving from  Female Ward to Doctors Room
```

Fig. 5.3.4-7 KBH ES showing the revised service plan after the doctor's

request for a service is placed

Step 5: Scenario-executing-user clicks on the Reschedule Service in KBH Robot Executive Supervisor Front-end window and looks at the text area in the same window (Fig. 5.3.4-7). He finds the Target location # 3 as the Doctors Room (not the Female Ward) and watches the later movements of the robot in Stage window. It completes the service requests according to the specification of the scenario.

The timing of major events in Stage simulation window can be in Table 5.3.4-1.

5.4. Evaluation

For evaluating the KB service approach, the primary scenario of KB object delivery is selected for further analyses regarding performance and scalability of the KB planner.

As shown in Fig. 5.2.2-1, our KB planner contains EDB (e.g., object facts, person facts, room facts etc.) and IDB (remaining part of KB). EDB has static part like person facts and dynamic part like object facts. Usually the sizes of both static and dynamic EDB have an impact on planning time. In order to verify the feasibility of our proposed approach in a real robot environment, we have performed a series of simulations for analyzing the impact of varying the dynamic EDB (object facts) size, for a given static EDB (person facts) size on planning time spent in KB planner modules.

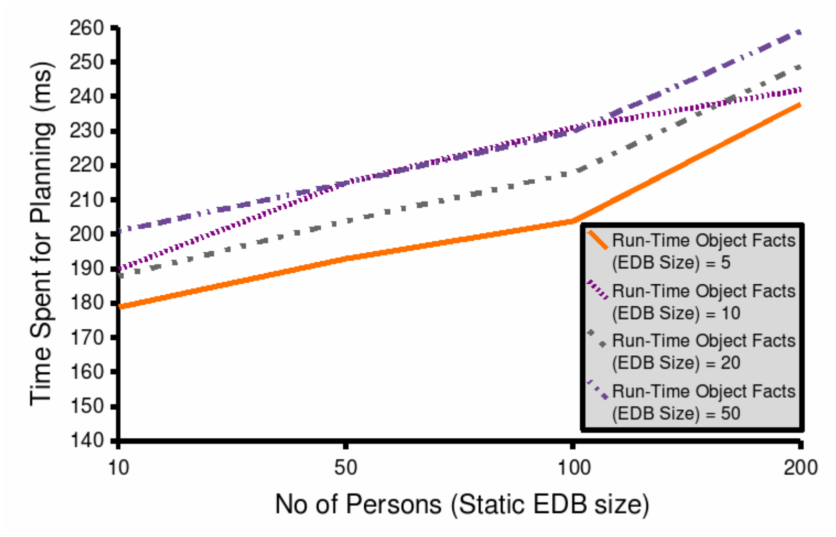


Fig. 5.4-1 Effect of varying EDB size on KB planning time

This gives us a general idea how the planning time increased if the number of persons of an office environment is changed and the number of delivery objects is also increased. The results have been plotted in Fig. 5.4-1. From this graph, we can see that if the number of persons (static EDB size) is increased planning time is also increased proportionally, but increasing the number of delivery objects (dynamic EDB size) has less impact on overall planning time. And for a typical case of an office environment with 100 persons and 20 delivery objects our planner took less than 250 milliseconds to generate the schedule of delivery.

Similarly, we have plotted the linear relationship between the increasing number of IDB rules and time spent in planning in planner modules in Fig. 5.4-2. For 4 sets of predetermined set of EDB we have measured the time for planning with three cases: 1) using only confidential object delivery policy (13 rules), 2) using both confidential and urgent object delivery policies (36 rules) and 3) using all three policies (49 rules).

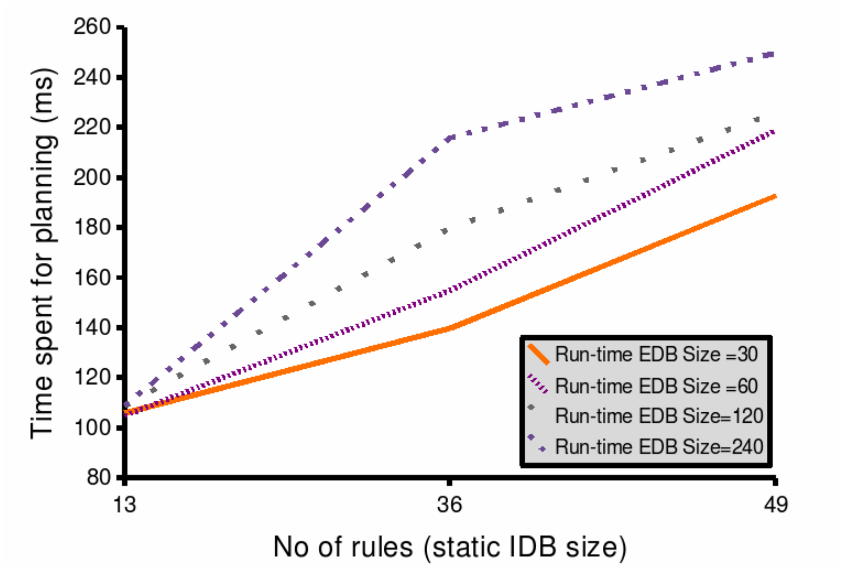


Fig. 5.4-2 Effect of varying IDB size on KB planning time

The EDB size is dynamically adjusted at run-time, e.g., in our first trial we have used EDB size = 30, (10 object facts, 17 person facts and 3 room facts). In each of three later trials, we have doubled EDB size, i.e., we have used EDB size 60 (7 object facts, 50 person facts and 3 room facts), 120 (17 object facts, 100 person facts and 3 room facts) and 240 (37 object facts, 200 person facts and 3 room facts). From these trials, we have found that for a given EDB size, planning time in-creases almost linearly with the increasing number of rules.

These simulations were performed dedicatedly in a Pentium IV machine with 256MB memory. From these observations, it is seen that planning time is feasible to implement it in a real robot in real-time.

5.5. Discussion

From the implementation of two scenarios discussed in the previous sections, several important aspects of KB service approach have been found. Major findings are discussed here.

- **Reusable Knowledge:** Under a KB service approach, a HC robot can plan and execute its service according to the user guidelines or policies for a service based on the necessary knowledge of the human environment. In case of object delivery service in an office, it is found that robot can plan the object delivery schedule according to the provided user policies such as policy for urgent/confidential object delivery etc. In this process, HC robot understands the meaning of seniority of persons based on user-guidelines for deriving seniority of persons. This knowledge is reusable in other services as well. This implies that KB service approach is expandable and it can use knowledge components over a wide range of services.

- **Fast Interaction with Humans:** If a human operator of a HC robot makes an explicit plan of a specific service provided by the robot, there is not much interactivity and efficiency in interaction with the robot. This robot is not more than a programmable machine. But in case of using KB service approach, a HC robot needs less information from human user (e.g., it does not need the order of senior persons in an office repeatedly) to plan a service according to the user guidelines. This savings of time and effort will play a very important role in efficient human-robot interaction.

- **Dynamic Response in Critical Services:** In dynamic situations HC robots need to reschedule its service tasks according to the changes of the environment and corresponding user-guidelines applicable for that context. Under hospital scenario, it is seen how a HC robot can handle dynamic situations for rescheduling service tasks based on priority or emergency of a task. Such differential services are very important in context of human living environment. This improves the QoS of HC robots significantly.

- **Better Integration with Human Environment:** A HC robot which has the knowledge of the human environment and another ordinary robot which does not have this knowledge are not the same. A HC robot having knowledge of human environment can better adjust and adapt to the changing situations of the environment and can perform better. Under hospital scenario, it is seen that if the robot has no knowledge of the hospital service policies, a human operator must interrupt the robot to reschedule the service manually. However, in this case robot can deduce a decision based on the knowledge of user policies. This improves the overall integration of HC robots in dynamic situations of human-living environment.

- **Feasibility of Implementation in Real Robot:** The evaluation of KB planner in Section 5.4 shows the feasibility of practical implementation of this approach in a real robot. Using a modular programming strategy, this approach can be implemented in real robot with reasonable number of user-policies and environmental constrains. The availability of open source robot simulators (like Player/Stage), KB engineering tools (like XSB deductive database) and large user community of these tools show the practicality of implementing this approach in a real robot platform.

6. Conclusion

In this chapter a summary of this thesis with concluding remarks on this research is presented. Major recommendations for future works have also been presented at the end of this chapter.

6.1. Summary and Concluding Remarks

In this thesis, a novel KB service approach has been proposed for HC robots that can appropriately adopt the knowledge of human environment (e.g., user relationships, guidelines/policies, preferences etc.) into its service planning and execution. This approach extends the generic hybrid (deliberative) paradigm of robot control by shifting the focus from the reactive behaviors to the adaptive services. This approach has been materialized on top of the Knowledge-Based Human-centered (KBH) software architecture. One of major components of this architecture is a KB planner that is integrated with XSB logic programming and deductive database system. This KBH architecture facilitates to simulate KB services in an open source mobile robot simulator Player/Stage. Here a robot can take the high-level knowledge of user's guidelines into account and plan and execute KB service accordingly. Two scenarios: 1) a KB object delivery service at office and 2) a KB patient care service at hospital have been simulated and the performance of scalable and real-time planning/execution of KB services have been reported.

Here, it is found that not only following the static user guidelines but also the dynamic re-planning of above mentioned services is feasible under this approach.

The following concluding remarks have been made based on the experiences of development and implementing this approach in this thesis.

1. Appropriateness of KB service approach

KB service approach is suitable for human-centered services in human-living environments where the quality of service depends heavily on the knowledge of the human environment. The model of this kind of environment should be represented in the form of facts or logical rules into a knowledge-base. The planning and execution of HC services should primarily be based on the knowledge of user policies and preferences.

2. Knowledge engineering issues

Representing knowledge of environment is a big issue of this approach. Therefore, suitable knowledge engineering tools and technologies should be selected depending on the requirements of application. Usually when the knowledge management is critical, frame based knowledge representation is used instead of rule based representation. In any case, selection of inference engine (e.g., solely inference engine or extended with deductive database) and its inference mechanism is also important in terms of development ease and run-time performance.

3. Software integration issues

Implementing heterogeneous software components in distributed fashion also creates bottle-necks in integration. So, during selection of software platforms and languages it should be carefully verified the capabilities of software components to be integrated. For example, inference engine of KB should be interfaced or connected to the planning component of the software architecture.

Research Contributions

The contribution from this research primarily goes to the issues of improving the quality of service (QoS) of a HC robot in human-living environment. Such qualities include (but not limited to) service prioritizing, customization, context adaptation and so on. Under this KB service approach, when the HC robots will follow the user guidelines for service planning and execution, the level of autonomy of these robots will be increased significantly. In such a case, HC robots can customize/adapt its services very quickly by the power of deductive inference and experience based learning. Consequently, an end-user can view this robot not only as his/her obedient assistant but also social, secure and safety-aware companion.

This research also contributes in the area of human-robot interaction (HRI). An HC robot possessing the knowledge of the human environment can be more interactive than a HC robot which has no knowledge of the human environment. That's why a KB service approach can create broader abstractions in human-robot communications. In this case, it is not necessary to explain the user preferences or priorities to a HC robot in each communication since this robot is already aware of those user attributes as a part of the knowledge of the human environment. This will greatly enhance HRI by saving the time and efforts of a user during interaction. Thus, a KB service approach can present an efficient HRI model.

6.2. Future Works

In this thesis a novel KB service approach has been introduced with limited implementation scope. So, it is necessary to include the following areas of future works for completing the implementation of this approach that can extend the potential benefits for the users of HC robots.

1. Implementation of learning with Memory Manager

The KBH software architecture contains a scope of learning based on experiences. As discussed in Section 4.2, Memory Manager module of KBH architecture is intended to provide the learning ability so that the services can be adapted based on the continuous service execution experiences (i.e., from reusable experiences). In this thesis, implementation of this module has not been addressed. Future work should be carried out for developing and implementing this important module in HC robot services.

2. Improving the interfaces among software components

Interfacing the various components of KBH architecture is a big issue for practical implementation of this KB service approach. Systems managers who provide the service policies will likely want to review and configure the service policies in planning component. Therefore, an extensive work is required to express the description of service policies in a suitable mark-up language like XML and translate them back and forth into logic programming modules. Interfacing with experience-base or memory management modules is also an issue when service experience will be reused.

3. Realizing effectiveness through practical HC services

The most important success criteria of KB service approach in HC robots are to implement this approach into real robot services and thoroughly review the performance of this system. One major aspect of this review process is to collect end-user feedbacks which can help to improve a specific implementation maximizing the effectiveness of KB service approach.

References

Arkin, R.C. (1990a). *Behavior Based Robotics.* The MIT Press.

Arkin, R.C. (1990b). ‘Integrating behavioral, perceptual, and world knowledge in reactive navigation’, Robotics and Autonomous Systems, Vol. 6, pp. 105–122.

Bertino E., et al. (1994). ‘Deductive Object Databases’, Proceedings of the 8th European Conference on Object-Oriented Programming. M. Tokoro and R. Pareschi, Eds., in Lecture Notes In Computer Science, vol. 821. Springer-Verlag, London, pp. 213-235.

Bonasso, R.P. et al. (1997). ‘Experiences with architecture for intelligent, reactive agents’, Journal of Experimental and Theoretical Artificial Intelligence, 9(2): 237–256.

Brooks, R. (1987). ‘A hardware retargetable distributed layered architecture for mobile robot control’, Proceedings of the IEEE International Conference on Robotics and Automation.

Clodic, A., et al. (2005). ‘Supervision and interaction: Analysis of an autonomous tour-guide robot deployment’, Proceedings of the 12th Int’l Conference on Advanced Robotics, ICAR, 2005, pages 725-732.

Das, S. K. (1992). *Deductive Databases and Logic Programming*. Addison-Wesley.

Forlizzi J. & DiSalvo, C. (2006). ‘Service robots in the domestic environment: a study of the roomba vacuum in the home', Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 258 - 265.

Firby, R.J. (1989). 'Adaptive execution in complex dynamic worlds', Ph.D. Thesis, Yale University.

Gat. E. (1997). 'On three-layer architectures', D. Kortenkamp, R. P. Bonnasso, and R. Murphy, editors, Artificial Intelligence and Mobile Robots. MIT/AAAI Press.

Gerkey B., Vaughan R. T. & Howard A. (2003). 'The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems', Proceedings of the 11th International Conference on Advanced Robotics, pages 317-323.

Graf, M., Hans, M. & Schraft R.D. (2004). 'Care-O-bot II—Development of a next generation robotic home assistant', Autonomous Robots, vol. 16, no. 2, pages 193-205.

Green C. (1969). 'Application of Theorem Proving to Problem Solving', Proceedings of the International Joint Conference on Artificial Intelligence', pages 219-239.

Han, J., et al. (2005). 'The educational use of home robots for children', Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, ROMAN 2005, pp. 378- 383.

IFR (2007). 'International Federation of Robotics', http://www.ifr.org.

IEEE/RAS TC for Service Robots (2007). ‘The Technical Committee for Service Robots of the IEEE Robotics and Automation Society’, http://www.service-robots.org.

Java-Player (2007), ‘The Java-Player’, http://java-player.sourceforge.net/

Joelle, P. et al. (2003). ‘Towards robotic assistants in nursing homes: Challenges and results’, Robotics and Autonomous Systems, vol. 42, no. 3-4, pages 271-281.

Kim Moonzoo, et al. (2005). ‘Re-engineering software architecture of home service robots: a case study’, Proceedings of the 27th International Conference on Software Engineering, ICSE 2005, pages 505-523.

Konolige, K. and Myers, K. (1996). ‘The Saphira Architecture for Autonomous Mobile Robots’, SRI International.

Koussoub'e S. (1997). ‘Knowledge-Based Systems: Formalisation and Applications to Insurance’, Research Report 108, UNU/IIST, P.O.Box 3058, Macau.

Kowlaski, R. A. (1979). ‘Algorithm = Logic + Control’, Communications of the ACM, 22(7), pp. 424-435.

Lindstr¨om, M., Oreb¨ack, A. and Christensen, H. (2000). ‘Berra: A research architecture for service robots’, International Conference on Robotics and Automation.

Murphy R. R. (2000). *Introduction to AI Robotics*. The MIT Press.

Player/Stage (2007). ‘The Player/Stage Project’, http://playerstage.sourceforge.net.

Russell S. J. & Norvig P. (2002). *Artificial Intelligence A Modern Approach*, 2 edn, Prentice-Hall, Inc.

Sagonas, K., Swift, T., and Warren, D. S. (1994). 'XSB as an efficient deductive database engine' SIGMOD Rec. vol. 23, no. 2, pages 442-453.

Sarker, M. O. F., Park, J-M., Kim, C. and You, B-J. (2007). 'A Knowledge-Based Service Approach for Human-Centered Robots', Proceedings of the IEEE Int'l Workshop on Robot and Human Interactive Communication, ROMAN 2007, *to appear.*

Sakai, et al. (2005). 'Developing a service robot with communication abilities', Proceedings of the IEEE Int'l Workshop on Robot and Human Interactive Communication, ROMAN 2005, pages 91-96.

Severinson, K. et al. (2003). 'Social and collaborative aspects of interaction with a service robot', Robotics and Autonomous Systems, vol. 42, no. 3-4, pages 223-234.

Shieh, M.Y. et al. (2004). 'Design of an intelligent hospital service robot and its applications', Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pages 4377- 4382.

Simmons, R. (1994). 'Structured control for autonomous robots', IEEE Transactions on Robotics and Automation.

Taipalus T. and Kosuge K. (2005). 'Development of service robot for fetching objects in home environment', Proceedings of the IEEE Int'l Symposium on Computational Intelligence in Robotics and Automation, CIRA, page(s): 451- 456.

Thrun S. (2004). 'Toward a framework for human-robot interaction', Human-Computer Interaction, vol. 19, no. 1-2, pages 9-24.

XSB (2007). 'The XSB Project', http://xsb.sourceforge.net.

YAJXB (2007). 'Yet another Java-XSB Bridge', http://infolab.stanford.edu/~stefan/rdf/yajxb/.

Appendix 1: Policy-Base Modules for Office Scenario

1. 1 Module senior.P

```
%% senior.P : basic rules of seniority in target environment
:- import person/6 from person_facts.
:- export rank/2, rank_senior/3, date_senior/3,
        rank_same/2, date_same/2, senior/3.

%% Set and decide seniority between 2 persons
rank(SId, 1) :-
    person(SId,_,Designation,_,_,_), Designation == 'Project Manager'.

rank(SId, 2) :-
    person(SId,_,Designation,_,_,_), Designation == 'Director'.

rank(SId, 3) :-
    person(SId,_,Designation,_,_,_), Designation == 'Sr. Res. Scientist'.

rank(SId, 4) :-
    person(SId,_,Designation,_,_,_), Designation == 'Res. Scientist'.

% Find rank seniority
rank_senior(SId1, SId2, SrSId) :-
    rank(SId1, Rank1), rank(SId2, Rank2), Rank1 < Rank2, SrSId = SId1.

rank_senior(SId1, SId2, SrSId) :-
    rank(SId1, Rank1), rank(SId2, Rank2), Rank2 < Rank1, SrSId = SId2.

rank_same(SId1, SId2) :-
    rank(SId1, Rank1), rank(SId2, Rank2), Rank1 =:= Rank2.

% Find employment date seniority
date_senior(SId1, SId2, SrSId) :-
    person(SId1,_,_,EmpStartDate1,_,_),
person(SId2,_,_,EmpStartDate2,_,_),
    EmpStartDate1 < EmpStartDate2, SrSId = SId1.

date_senior(SId1, SId2, SrSId) :-
    person(SId1,_,_,EmpStartDate1,_,_),
person(SId2,_,_,EmpStartDate2,_,_),
    EmpStartDate1 > EmpStartDate2, SrSId=SId2.

date_same(SId1, SId2) :-
    person(SId1,_,_,EmpStartDate1,_,_),
person(SId2,_,_,EmpStartDate2,_,_),
    EmpStartDate1 =:= EmpStartDate2.

% Policy.Seniority: First by Rank, then by EmpStartDate
% for similarity use "First come (in query) First serve (in answer)"

senior(SId1, SId2, SrSId) :-
    rank_senior(SId1, SId2, SrSId).

senior(SId1, SId2, SrSId) :-
    rank_same(SId1, SId2), date_senior(SId1, SId2, SrSId).

senior(SId1, SId2, SrSId) :-
     date_same(SId1, SId2), rank_same(SId1, SId2), SrSId = SId1.
    % First come first serve applied here.
```

1. 2 Module distance_nearer.P

```
% sort according to the near distance
:- import person/6 from person_facts.
:- import room/1 from room_facts.
:- export  room_no/2, distance_nearer/3.
% Find room no of a corresponding persons.
room_no(SId,RoomNo):- person(SId,_,_,_,RoomNo,_ ).

% distance_nearer: is nearer distance from our service start point
distance_nearer(Person1, Person2, Person1):-
    room(Room1), room(Room2), room_no(Person1, Room1),
    room_no(Person2, Room2), Room1 < Room2, !.
distance_nearer(Person1, Person2, Person2):-
    room(Room1), room(Room2), room_no(Person1, Room1),
    room_no(Person2, Room2), Room1 > Room2.
```

1. 3 Module sort_by_senior.P

```
% sort_by_senior(+Unsorted, ?SortedAsPerSeniority)
% % Adapted from the quicksort algorithm.
:- import append/3 from basics.
:- import senior/3 from  senior.
:- export sort_by_senior/2.

sort_by_senior([],[]).
sort_by_senior([X|Tail], Sorted) :-
    split(X, Tail, Small, Big), sort_by_senior(Small, SortedSmall),
    sort_by_senior(Big, SortedBig), append(SortedSmall,
[X|SortedBig], Sorted).

split(X, [],[],[]).
% customization is here: (X > Y) replaced by senior(X,Y,Y)
split(X, [Y|Tail], [Y|Small], Big) :-
    senior(X ,Y, Y), !, split(X, Tail, Small, Big).
split(X, [Y|Tail], Small, [Y|Big]) :-
    split(X, Tail, Small, Big).
```

1. 4 Module sort_by_distance.P

```
% sort_by_distance(+Unsorted, ?Sorted_by_distance)
% % Adapted from the quicksort algorithm.
:- import append/3 from basics.
:- import distance_nearer/3 from  distance_nearer.
:- export sort_by_distance/2.
sort_by_distance([],[]).
sort_by_distance([X|Tail], Sorted) :-
    split(X, Tail, Small, Big), sort_by_distance(Small, SortedSmall),
    sort_by_distance(Big, SortedBig), append(SortedSmall,
[X|SortedBig], Sorted).

split(X, [],[],[]).
% customization is here: (X > Y) replaced by senior1(X,Y,Y) :-)
split(X, [Y|Tail], [Y|Small], Big) :-
    distance_nearer(X ,Y, Y), !, split(X, Tail, Small, Big).
split(X, [Y|Tail], Small, [Y|Big]) :-
    split(X, Tail, Small, Big).
```

1.5 Module policy_regular_delivery.P

```
%% produces a regular delivery list by distance from service start point.

:- import object/5 from object_facts.
:- import sort_by_distance/2 from sort_by_distance.

:- export regular_delivery_list/1.

regular_rcvr(Rcvr):-
    object(_,_,Rcvr,_,IsUrgent),IsUrgent =:= 0.


% Make a list of rcvr. who will rcv. urgent obj.
regular_rcvr_set(List):-
    setof(Rcvr, Rcvr ^ regular_rcvr(Rcvr), List) .
regular_rcvr_set([]).

% Sort by seniority
regular_delivery_list(SortedList):-
    regular_rcvr_set(RawList), sort_by_distance(RawList, SortedList).
```

1. 6 Module plan_obj_delivery.P

```
%% plan_obj_delivery.P produces a plan for object
% delivery by combining 3 policies:
% policy_confid_delivery, poicy_urgent_delivery,
% policy_regular_delivery.
:- import append/3 from basics.
:- import remove_duplicates/2 from lists.
% Custom library import
:- import confid_rcvr/1, confid_rcvr_set/1,
confid_delivery/1 from policy_confid_delivery.
:- import urgent_delivery_list/1 from
policy_urgent_delivery.
:- import regular_delivery_list/1 from
policy_regular_delivery.
:- import rcvr2loc/2 from rcvr2loc.
:- export confid_delivery_list/1,
urgent_list_added/1,
delivery_list/1, plan_obj_delivery/0.

%% First check answer from policy_confid_delivery
% for invalid answer, report error to exit.
plan_obj_delivery :-
confid_delivery(Ans), Ans == 'Invalid',
confid_rcvr_set(R),
writeln('Error in your delivery request !'),
write('Too many confidential receivers:'),
 writeln(R), writeln('Please select only one receiver for your requst.').

% For valid answer, make a list and add the
% receiver at the head of the list.
% when confid_delivery_list is empty
:-confid_delivery_list([]).
confid_delivery_list(Rcvr):-
confid_delivery(Ans), Ans == 'Valid',
confid_rcvr_set(Rcvr), !.

% Add urgent list to it.
urgent_list_added(AddedList):-
confid_delivery_list(L1), ugent_delivery_list(L2),
append(L1,L2,AddedList).

% Add regular list to it.
delivery_list(FinalList):-
urgent_list_added(L1), regu-lar_delivery_list(L2),
append(L1, L2, AddedList), !,
remove_duplicates(AddedList, FinalList).
% The above ! (cut) prevents backtracking
delivery_location(LocList):-
delivery_list(Rcvrs), rcvr2loc(Rcvrs, Locs),
remove_duplicates(Locs, LocList).
plan_obj_delivery :-
confid_delivery(Ans), Ans == 'Valid',
delivery_list(R), delivery_location(L),
writeln('Your delivery request is valid.'),
writeln('I am going to process your request with this plan-'),
write('Persons: '), writeln(R),
write('Locations: '), writeln(L).
```

Appendix 2: Policy-Base Modules for Hospital Scenario

2. 1 Module user_priority_policy.P

```
% user_priority_policy.P
:- import user/5 from user_facts.
:- export user_rank/2, prioritized_user/3.

% Define ranks
user_rank(UID, 101):- user(UID,_,Type,_,_), Type == doctor.

user_rank(UID, 201):- user(UID,_,Type,_,_), Type == nurse.

user_rank(UID, 301):- user(UID,_,Type,Desc,_),
    Type == patient, Desc == 'Emergency Patient'.
user_rank(UID, 311):- user(UID,_,Type,Desc,_),
    Type == patient, Desc == 'VIP Patient'.
user_rank(UID, 321):- user(UID,_,Type,Desc,_),
    Type == patient, Desc == 'Regular Patient'.

% Compare & Find prioritized user
prioritized_user(UID1, UID2, Ans):-
    user_rank(UID1, Rank1), user_rank(UID2, Rank2),
    Rank1 < Rank2, Ans = UID1.
```

2. 2 Module location_name2id.P

```
% converts symbolic name of location to room no. or location id

:- import location/1 from location_facts.
:- export location_id/2.

location_id(Name, 101) :- location(Name), Name == nursesRoom.
location_id(Name, 102) :- location(Name), Name == doctorsRoom.
location_id(Name, 103) :- location(Name), Name == supportRoom.
location_id(Name, 104) :- location(Name), Name == drugStore.
location_id(Name, 105) :- location(Name), Name == receptionRoom.
location_id(Name, 106) :- location(Name), Name == intensiveCareUnit.
location_id(Name, 107) :- location(Name), Name == dinningRoom.
location_id(Name, 108) :- location(Name), Name == diagnosticLab.
location_id(Name, 109) :- location(Name), Name == femaleWard.
location_id(Name, 110) :- location(Name), Name == maleWard.
location_id(Name, 111) :- location(Name), Name == vipCabin.

location_id(Name, 1001) :- location(Name), Name == westCorridorLeft ;
                                        Name == hospitalEntrance.
location_id(Name, 1002) :- location(Name), Name == westCorridorMiddle.
location_id(Name, 1003) :- location(Name), Name == westCorridorRight ;
                                        Name == hospitalExit.
location_id(Name, 1004) :- location(Name), Name == centerCorridorLeft.
location_id(Name, 1005) :- location(Name), Name ==
centerCorridorMiddle.
location_id(Name, 1006) :- location(Name), Name == centerCorridorRight.
location_id(Name, 1007) :- location(Name), Name == eastCorridorLeft.
location_id(Name, 1008) :- location(Name), Name == eastCorridorMiddle.
location_id(Name, 1009) :- location(Name), Name == eastCorridorRight.
```

2. 3 Module sort_users_by_priority.P

```
% sort_users_by_priority(+Unsorted, ?SortedAsPerPriority)

:- import append/3 from basics.
:- import prioritized_user/3 from  user_priority_policy.
:- export sort_users_by_priority/2.

sort_users_by_priority([],[]).
sort_users_by_priority([X|Tail], Sorted) :-
    split(X, Tail, Small, Big), sort_users_by_priority(Small,
SortedSmall),
    sort_users_by_priority(Big, SortedBig), append(SortedSmall,
[X|SortedBig], Sorted).

split(X, [],[],[]).
% customization is here: (X > Y) replaced by senior1(X,Y,Y)
split(X, [Y|Tail], [Y|Small], Big) :-
    prioritized_user(X ,Y, Y), !, split(X, Tail, Small, Big).
split(X, [Y|Tail], Small, [Y|Big]) :-
    split(X, Tail, Small, Big).
```

2. 4 Module   sorted_service_request.P

```
% sorted_service_request.P orders service req #
:- import service_request/6 from service_request_facts.
:- import sort_users_by_priority/2 from sort_users_by_priority.
:- export requesters/1, sorted_requesters/1,
           sorted_service_request/1, requester2cmdLoc/2.

requesterID(RUID):- service_request(_,_,RUID,_,_,_).
requestID(RID):- service_request(RID,_,_,_,_,_).
requesters(RList) :-
    setof(RUID, RUID ^ requesterID(RUID), RList).%requesters([]).
requests(RList):- setof(RID, RID ^ requestID(RID), RList).

% sort by priority
sorted_requesters(SortedList):-
    requesters(RawList), sort_users_by_priority(RawList, SortedList).

% Find corresponding service_request #
requester2requestNo([], []).
requester2requestNo([RUID|Tr], [RID|Tl]):-
    service_request(RID,_,RUID,_,_,_),  requester2requestNo(Tr,Tl).

% For taking multiple request from single user
sorted_service_request(ReqNoList):-
    sorted_requesters(RequesterList),
    requester2requestNo(RequesterList, ReqNoList).

% Convert RequesterID to task plan of  cmdLoc
%cmdLoc([]).
requester2cmdLoc([], []).
requester2cmdLoc([RUID|Tr], [FromLoc,ToLoc|Tl]):-
    service_request(_,_,RUID,_,FromLoc,ToLoc),
    requester2cmdLoc(Tr,Tl).
```