SDAG TECHNICAL NOTES - 86-01 November 1986

# Architecture of Autonomous Systems[1]

Piyush Dikshit
Katia Guimaraes
Maya Ramamurthy
Ashok Agrawala
Ronald L. Larsen

Systems Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742



[1]This research was supported in part by NASA under Grant NCC-2414

# Contents


1 Introduction 1

2 Conceptual Framework 1

3 Operator Interface 3

4 Task Planning and Reasoning 3

5 Sensing and Perception 5

6 Control Execution 6

7 Control Loops 6

8 Knowledge Base 9

9 General Manager 10

# 1 Introduction

Automation of Space Station functions and activities, particularly those involving robotic capabilities with interactive or supervisory human control, is a complex, multi-disciplinary systems design problem. A wide variety of applications using autonomous control can be found in the literature, but none of them seem to address the problem in general. All of them are designed with a specific application in mind.

In this report, an abstract model is described which unifies the key concepts underlying the design of automated systems such as those studied by the aerospace contractors. The model has been kept as general as possible. The attempt is to capture all the key components of autonomous systems. With a little effort, it should be possible to map the functions of any specific autonomous system application to the model presented here.

# 2 Conceptual Framework

A major portion of human endeavor involves observing the state of the world and altering that state to achieve some desired outcome. Automated systems are viewed in this context as providing a transfer function between the human and the world in order to extend human capabilities. Teleoperated systems, for example, extend human reach to remote places which are difficult or hazardous for the human to function in, and may amplify human strength and sensory abilities.

The conceptual view taken here, therefore, is that an automated system resides between a human (the operator) and the domain of concern (the external world). The system provides the operator with an ability to observe the state of the world remotely and to describe changes which are to be made to that state. The more the degree of automation in the system, the less the need for operator intervention.

The conceptual framework consists of four major functions integrated into a unifying system architecture: 1. the operator interface, 2. task planning and reasoning, 3. sensing and perception, and 4. control execution. These functions are then organized to provide three control loops: an operator control loop, an executive control loop, and a local control loop (Fig. 1).

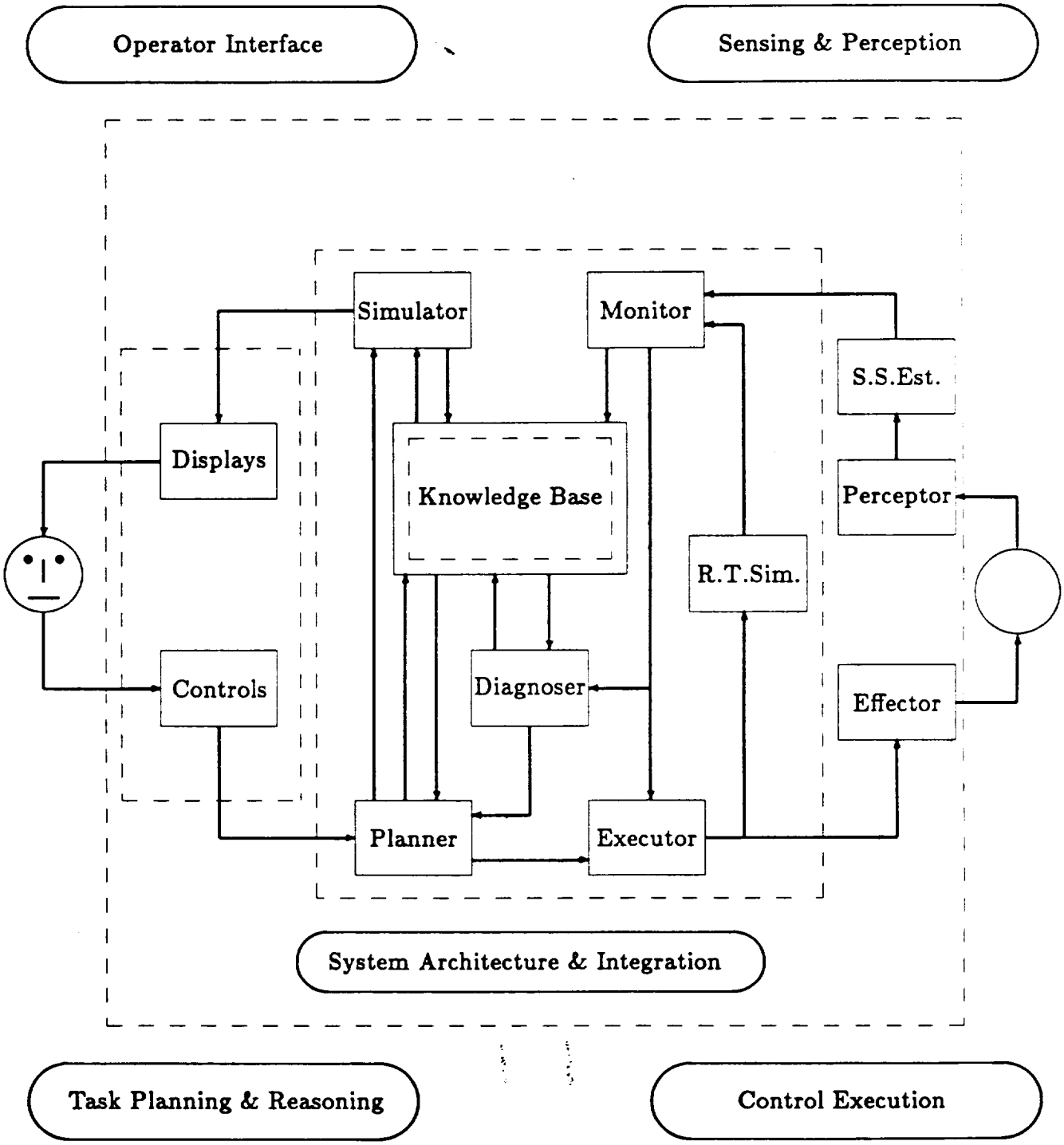


Figure 1. Automated System Control Architecture

# 3 Operator Interface

The operator interface includes displays and controls. Displays provide all of the information about system operation to the human operator, and can employ a variety of mechanisms for communicating information through the senses. Visual displays, such as graphics video display workstations, head-up displays, the body, as through hand controllers or exoskeletons, can also be used, as can aural feedback and voice generation.

Controls provide the means by which the operator communicates information to the system, such as the nature or details of the task to be performed. Alphanumeric keyboards commonly provide this capability, often augmented with function keys, light pens, and similar mechanisms. Voice input, hand controllers, exoskeletons, and pedals provide additional options for the operator.

The integration of displays with controls provides the environment for human interaction with the system. It provides the means by which the human operator can communicate the description of the task to be performed to the system, and the means by which the system can communicate the status of task execution back to the operator.

# 4 Task Planning and Reasoning

Task planning and reasoning includes most of the intelligence of the automated system. It receives the description of the task to be performed from the Controls, translates that into commands which can be remotely executed to change the state of the world, observes the changing world state, handles unexpected discrepancies in the world, seeking operator intervention only when it cannot cope with a world situation with the "intelligence" it has. Task planning and reasoning requires an extensive Knowledge Base containing all of the data required to represent and reason about the task domain. It includes a dynamic (time-dependent) world model which expresses the natural laws governing behavior of the system, a CAD/CAM-style data base which provides the data required to understand the sensed world and manipulate it, system configuration data which describes the current suite of operational equipment, and the set of heuristic rules which

are required to reason successfully and effectively in the task domain. The major capabilities of task planning and reasoning are provided by five functional modules: 1. the Planner, 2. the Executor, 3. the Simulator, 4. the Monitor, and 5. the Diagnoser.

The Planner receives the task description from the Controls, and the current state of the world from the Knowledge Base, including all known information relevant to the solution of the problem at hand. The Planner transforms the (typically) high level description of the task into a nominal plan of action to accomplish the stated objective. This requires sufficient knowledge of the problem domain, the state of the system and the environment, and solution rules which lead to successful development of an acceptable plan. The plan is a time sequenced series of primitive actions which are used by the Executor for real-time operation.

The Simulator is a tool for the operator to carry out preliminary investigations of the effect of certain commands before they can be used in the real world. It gets simulation requests from the operator, a description of the real world from the Knowledge Base ( or from the operator himself if the command is to be given to a simulated world state) and computes the time series of world state changes corresponding to the request. The simulation results are sent to Displays.

The Executor accepts the nominal plan of action from the Planner and issues primitive commands in real-time as dictated by the nominal plan. It also receives minor vernier adjustments which are superimposed on the command sequence to fine tune the execution of the operation in response to unanticipated discrepancies between the expected state of the world and the actual state. These adjustments are made to compensate for errors which do not threaten successful completion of the nominal plan.

The Real Time Simulator computes Expectations about the Real World in real time based on the commands issued by the Executor, the current world state in the Knowledge Base and a command script which describes the effect of commands on the Real World. These Expectations are used by the Monitor to detect discrepancies between what is observed and what is expected in the Real World.

The State Space Estimator uses the observation data provided by the Perceptor to derive a state description of the Real World, giving meaning

to the raw information gathered by the Perceptor. The interpreted world state description is placed in the Knowledge Base.

The Monitor uses expected values of state descriptors (computed by the Real Time Simulator) and the observed data in real-time (placed in the Knowledge Base by the State Space Estimator) to detect discrepancies between the expected world state and the actual world state. Non-goal-threatening discrepancies are transformed into vernier adjustments which are fed back to the Executor. Goal-threatening discrepancies (such as failure of a required device, or detection of a broken component) require further analysis. An anomaly report is formatted and sent to the Diagnoser.

The Diagnoser receives anomaly reports from the Monitor. From the anomaly report and the current world state, it attempts to assess the nature of the anomaly, its cause, and potential effects. It computes an "inferred world state," i.e., the most plausible explanation and implied new system state, and places this in the knowledge base as the best estimate of the current world state. It then issues a replan order to the Planner, which informs the Planner that the nominal plan is no longer satisfactory. The Planner will formulate a new nominal plan or alert the operator if it cannot do so.

# 5 Sensing and Perception

The sensing and perception function provides a mapping from measurable internal and external parameters to an estimate of the world state. The function typically acquires its input from a multi-mode array of task and world sensing devices, such as cameras, force/torque sensors, proximity detectors, heat sensors, accelerometers, strain gauges, and voltage/current sensors. The objective of the sensing and perception function is to provide the Monitor with a real-time best estimate of the state of the task domain in terms which are consistent with the expected values provided by the Real Time Simulator.

# 6 Control Execution

The control execution function includes the manipulators and effectors, and tools available at the task site which can be used to accomplish the desired task, as well as the low level controls required to transform primitive commands issued by the Executor into state-changing actions.

# 7 Control Loops

The automated system control architecture illustrated consists of three major control loops. The Operator Control loop includes the human operator plus the Controls, Planner, Simulator, and Displays. This loop provides the interaction between the human operator and the system. It can operate independently, providing a pure system simulation capability. This might be effective for operator training or system evaluation, for example.

The Executive Control loop includes all of the components required to operate and control a subsystem with minimal assistance from a human operator: the Planner, Executor, Real Time Simulator, Monitor, Diagnoser, and Knowledge Base. This core loop provides a generic control architecture which is equally applicable to controlling a deep space spacecraft or an autonomous vehicle as it is to a robotic system.

The Local Control loop, consisting of the Executor, Real Time Simulator, Effector, Perceptor, State Space Estimator and Monitor, provides the low level control for the specific manipulators used in the robot, as well as the sensorbased information extraction algorithms.

These three control loops can be visualized as a train of three gears, as illustrated in Figure 2. If one visually portrays the significance of the three loops for different operating modes by their relative size, one can begin to understand the robustness of the architecture. Figure 3a illustrates the typical factory robot, in which nearly all of the emphasis has been placed on the manipulator loop. This results in relatively dumb robots (e.g., pick and place, welding, or painting) which have simple computer controllers and minimal operator interaction except during the task programming phase. Figure 3b illustrates a teleoperator control concept, in which the executive control loop has been reduced to its most simple form, performing data

transfer functions, but little more. The principle emphasis is on the operator loop to provide a good operator interface and on the local control loop to provide good manipulative capabilities. In Figure 3c in contrast, an architecture for an intelligent robot is portrayed. Here, a massive executive control loop replaces the strong operator loop of the teleoperator. Finally, figure 3d illustrates a supervisory controlled telerobot, which features a balance among the three control loops, providing the operator with the richness of a teleoperated system when the situation demands it, and providing the capability of a "semi-intelligent" autonomous robot when the situation allows it.

As the loops pictorially change in relative size, sophistication of the component subsystems changes drastically. In an attempt to maintain a consistent architecture, all subsystems are retained for all operating modes. In some cases, though, they are non-functional (essentially non-existent except to provide a data interface between the other subsystems), while in other cases, they might be million-line programs.

It is important to realize that these three control loops do not work in the same time-frame. The Local Control Loop works in real time as it needs to keep up with the changes in the real world. The Operator Control Loop, on the other hand, is used only for high level planning and simulation/testing activities. In a truly autonomous syatem, it will have little to do with the normal run-time functioning of the rest of the system. The Executive Control Loop falls somewhere in between. It is the core of the "intelligence" exhibited by the autonomous system. Its right half ( Executor, Real Time Simulator and Monitor) operates in real time. Only when there is an anomaly in the real time operation does the central part of the loop (Diagnoser) become active. The Diagnoser signals the Planner to take care of the situation. Human intervention will be necessary if the Planner cannot handle the anomaly. The frequency with which operator help is sought will determine how autonomous the system really is.

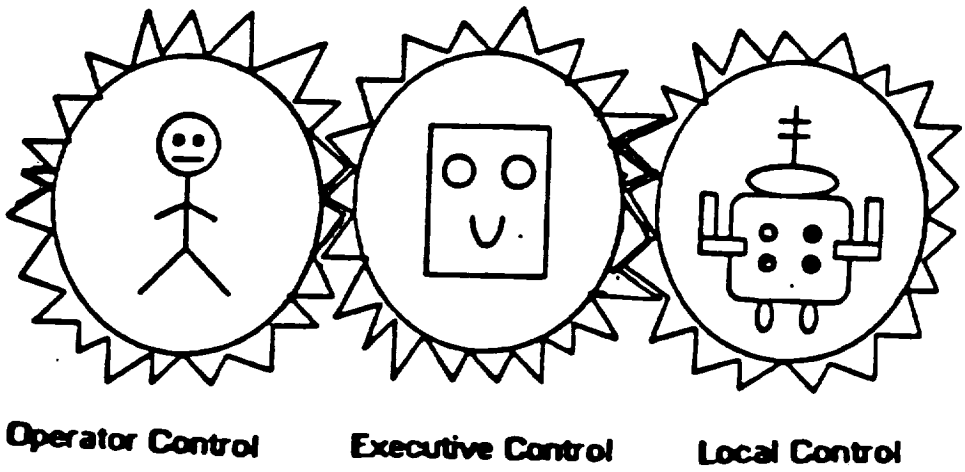


Figure 2. Meshed Control Loops

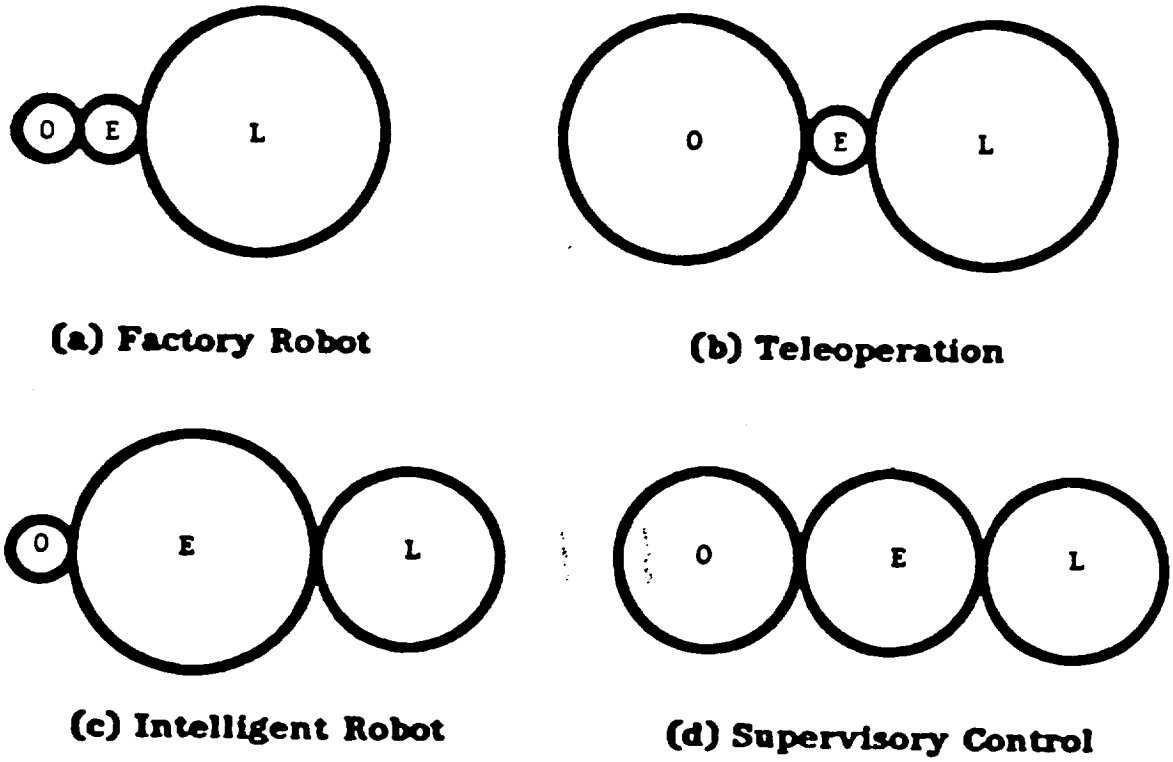


Figure 3. Alternative Control System Realizations

# 8 Knowledge Base

The Knowledge Base is the repository of all information in the system . Some of the information is about the real time operation of the system (and hence used in the Local Control Loop), some is used for planning and evaluation (by the Executive Control Loop) and some is to aid the operator in testing and simulation activities (in the Operator Control Loop) All access to the stored data is centralized using the Knowledge Base Manager.

It contains the following information:

- System Configuration
- Constraints
- Command Script (for Executor commands)
- Real World State Description (provided by State Space Estimator)
- A set of policies/algorithms for the Planner to choose from
- Objective Function (representing the long-term goal of the system)
- Statistical Information (for performance evaluation)
- Test/Diagnostic/Simulation Tools for the operator
- History of the system
- Explanation for anomalies (provided by Diagnoser when needed)

The Knowledge Base Manager is responsible for ensuring the consistency and integrity of the information in the Knowledge Base. It also acts as the interface between two or more independent autonomous systems.

# 9 General Manager

It has been considered the inclusion of a special module whose purpose is to serve as an interface between the autonomous system and an "Executive

There should be a way to control and monitor the components of an autonomous system. In other words , it is necessary to have a special module, called "General Manager", exercising control over all parts of the autonomous system.

There can be several modes of operation for a given autonomous system. Each mode entails a different set of functiona for each component of the system and, possibly, a different set of inter-connections between the components. For example, when there is a malfunction in some part of the system, it will enter a "diagnostic" mode of operation till such time as the fault can be corrected and "normal" mode resumed.

The General Manager initiates and monitors the autonomous system. It can reconfigure the system and dynamically alter the functions of and the interactions between the parts of the system. It has complete access to every component of the system for detecting and analyzing anomalous behaviour. It can use message logs and statistical information in the Knowledge Base to evaluate the performance of the system and detect bottlenecks.

SDAG TECHNICAL NOTES - 86-02 December 1986

# INCO Shuttle Communication System[1]

Piyush Dikshit
Katia Guimaraes
Maya Ramamurthy
Ashok Agrawala
Ronald L. Larsen

Systems Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742

[1]This research was supported in part by NASA under Grant NCC-2414

# Contents


**1 Introduction** 1

**2 Conceptual Framework** 1

2.1 Major Functions . . . . . 1

2.2 Control Loops . . . . . 5

2.3 Knowledge Base . . . . . 6

**3 INCO Shuttle Communication Model** 7

3.1 Environment Description . . . . . 7

3.2 Knowledge Base . . . . . 7

3.3 The Model . . . . . 8

**4 Concluding Remarks** 10

# 1 Introduction

In a previous work we have defined a general architectural model for autonomous systems, which can be mapped easily to describe the functions of any automated system (SDAG-86-01). In this note, we use the model to describe the Shuttle communication system.

First we briefly review the architecture, then we present the environment of our application, and finally we detail the specific function for each functional block of the architecture for that environment.

# 2 Conceptual Framework

The conceptual framework consists of four major functions integrated into a unifying system architecture: 1. the operator interface, 2. task planning and reasoning, 3. sensing and perception, and 4. control execution. These functions are then organized to provide three control loops: 1. an operator control loop, 2. an executive control loop, and 3. a local control loop. The overall concept is illustrated in figure 1.

## 2.1 Major Functions

The operator interface includes displays and controls. Displays provide all of the information about system operation to the human operator, and can employ a variety of mechanisms for communicating information. Visual displays, such as graphics video display workstations, head-up displays, hand controllers or exoskeletons, can also be used, as can aural feedback and voice generation.

Controls provide the means by which the operator communicates information to the system, such as the nature or details of the task to be performed. Alphanumeric keyboards commonly provide this capability, often augmented with function keys, light pens, and similar mechanisms.

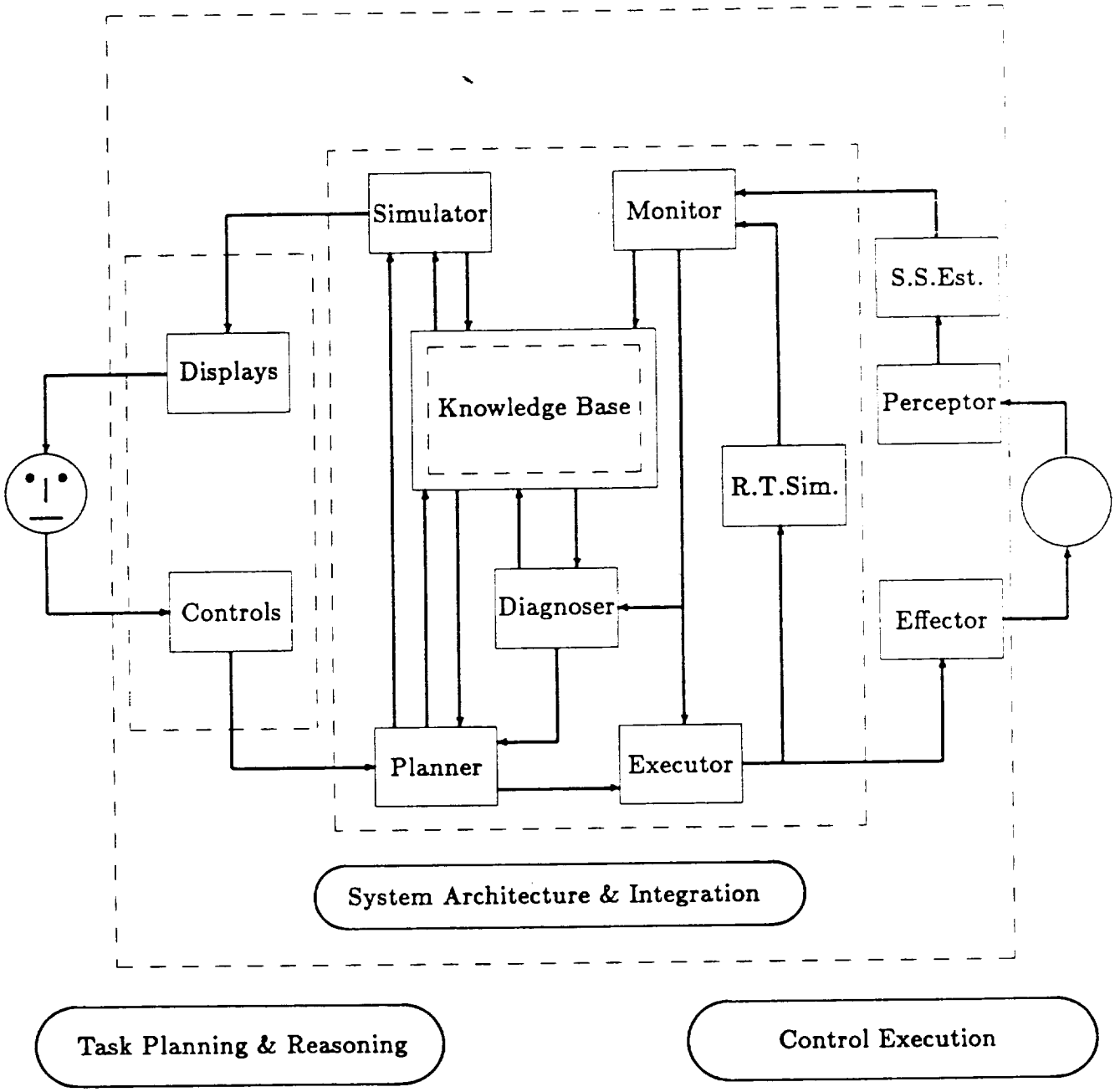


**Figure 1. Automated System Control Architecture**

The integration of displays with controls provides the environment for human interaction with the system. It provides the means by which the human operator can communicate the description of the task to be performed to the system, and the means by which the system can communicate the status of task execution back to the operator.

Task planning and reasoning includes most of the intelligence of the automated system. It receives the description of the task to be performed from the Controls, translates that into commands which can be remotely executed to change the state of the world, observes the changing world state, handles unexpected discrepancies in the world, seeking operator intervention only when it cannot cope with a world situation with the "intelligence" it has. Task planning and reasoning requires an extensive Knowledge Base containing all of the data required to represent and reason about the task domain. The major capabilities of task planning and reasoning are provided by seven functional modules: 1. Planner, 2. Simulator, 3. Executor, 4. Real Time Simulator, 5. State Space Estimator, 6. Monitor, and 7. Diagnoser.

The Planner receives the task description from the Controls, and transforms the (typically) high level description of the task into a nominal plan of action. The plan is a time sequenced series of primitive actions which are used by the Executor for real-time operation.

The Simulator is a tool for the operator to carry out preliminary investigations of the effect of certain commands before they can be used in the real world. It gets simulation requests from the operator, a description of the real world from the Knowledge Base ( or from the operator himself if the command is to be given to a simulated world state) and computes the sequence of world state changes over time corresponding to the request. The simulation results may be sent to Displays, along with the Knowledge Base.

The Executor accepts the nominal plan of action from the Planner and issues primitive commands in real-time. It also receives minor adjustments, which are superimposed on the command sequence to fine tune the execution of the operation in response to unanticipated discrepancies between the expected state of the world and the actual state.

The Real Time Simulator computes Expectations about the Real World

in real time based on the commands issued by the Executor, the current world state in the Knowledge Base and a command script which describes the effect of commands on the Real World. These Expectations are used by the Monitor to detect discrepancies between what is observed and what is expected in the Real World.

The State Space Estimator uses the observation data provided by the Perceptor to derive a state description of the Real World, giving meaning to the raw information gathered by the Perceptor. The interpreted world state description is placed in the Knowledge Base also.

The Monitor uses expected values of state descriptors (computed by the Real Time Simulator) and the observed data in real-time to detect discrepancies between the expected world state and the actual world state. Non-goal-threatening discrepancies are transformed into adjustments which are fed back to the Executor. Goal-threatening discrepancies (such as failure of a required device, or detection of a broken component) require further analysis. An anomaly report is sent to the Diagnoser.

The Diagnoser receives anomaly reports from the monitor, and attempts to assess the nature of the anomaly, its cause, and potential effects. It computes an "inferred world state", i.e., the most plausible explanation and implied new system state, and places this in the knowledge base as the best estimate of the current world state. It then issues a replan request to the Planner, which informs the Planner that the nominal plan is no longer satisfactory. The Planner formulates a new nominal plan or alerts the operator if it cannot do so.

The sensing and perception function provides a mapping from measurable internal and external parameters to an estimate of the world state. Its input is typically acquired from a multi-mode array of tasks and world sensing devices. The objective of the sensing and perception function is to provide the Monitor with a real-time best estimate of the state of the task domain in terms which are consistent with the expected values provided by the Real Time Simulator.

The control execution function includes the manipulators and effectors, and tools available at the task site which can be used to accomplish the desired task, as well as the low level controls required to transform primitive commands issued by the Executor into state-changing actions.

## 2.2 Control Loops

The automated system control architecture illustrated consists of three major control loops. The Operator Control loop includes the human operator plus the Controls, Planner, Simulator, and Displays. This loop provides the interaction between the human operator and the system. It can operate independently, providing a pure system simulation capability. This capability is desirable for operator training or system evaluation, for example.

The Executive Control loop includes all of the components required to operate and control a subsystem with minimal assistance from a human operator: the Planner, Executor, Real Time Simulator, Monitor, Diagnoser, and Knowledge Base. This core loop provides a generic control architecture which is equally applicable to controlling a deep space spacecraft or an autonomous vehicle as it is to a robotic system.

The Local Control loop, consisting of the Executor, Real Time Simulator, Effector, Perceptor, State Space Estimator and Monitor, provides the low level control for the specific manipulators used in the robot, as well as the sensorbased information extraction algorithms.

It is important to realize that these three control loops do not work in the same time-frame. The Local Control Loop works in real time as it needs to keep up with the changes in the real world. The Operator Control Loop, on the other hand, is used only for high level planning and simulation/testing activities. In a truly autonomous system, it may have little to do with the normal run-time functioning of the rest of the system. The Executive Control Loop falls somewhere in between. It is the core of the "intelligence" exhibited by the autonomous system. Its right half (Executor, Real Time Simulator and Monitor) operates in real time. Only when there is an anomaly in the real time operation does the central part of the loop (Diagnoser) become active. The Diagnoser evaluates the anomalous condition and generates the necessary action requests for the Planner. Human intervention will be necessary if the Planner cannot handle the anomaly. The frequency with which operator help is sought determines how autonomous the system really is, and other operating conditions of the system.

## 2.3 Knowledge Base

The Knowledge Base is the repository of all information in the system. Some of the information is about the real time operation of the system (and hence used in the Local Control Loop), some is used for planning and evaluation (by the Executive Control Loop) and some is to aid the operator in testing and simulation activities (in the Operator Control Loop). All audit trail information is also saved in the Knowledge Base and is available to different components of the system, including the operator, as necessary. All access to the stored data is centralized using the Knowledge Base Manager.

It contains the following information:

- System Configuration
- Constraints
- Command Script (for Executor commands)
- Real World State Description (provided by State Space Estimator)
- A set of policies/algorithms for the Planner to choose from
- Objective Function (representing the long-term goal of the system)
- Statistical Information (for performance evaluation)
- Test/Diagnostic/Simulation Tools for the operator
- History of the system
- Explanation for anomalies (provided by Diagnoser when needed)

The Knowledge Base Manager is responsible for ensuring the consistency and integrity of the information in the Knowledge Base. It also acts as the interface between two or more independent autonomous systems.

# 3 INCO Shuttle Communication Model

## 3.1 Environment Description

In this section, we adapt the proposed architecture to a simple communication system. It consists of an orbiter moving in a 90-min orbit, two orbiter antennas (one channel each), and four ground stations. The information available to the system is:

1. Earth view periods - for each ground station, the periods in which it can be viewed from the orbiter, and
2. Orbiter view plan - for each antenna, the periods when it is in view of at least one ground station.

The main goal is to schedule communication links in space. The requests for using these channels during given intervals of time have to be handled by scheduling the available resources. This task can be done without direct human intervention. The operator establishes priorities and policies, and the system generates schedules accordingly, and see that they are properly carried out.

## 3.2 Knowledge Base

For such a system, the Knowledge Base would contain the following information:

- Earth view periods
- Orbiter view plan
- Constraints
- Pre and post conditions
- System configuration
- System dynamics

- Scheduling algorithms
- World state description
- Time history
- Objective functions
-
-
-

## 3.3 The Model

Priorities and policies to be used for scheduling are established initially, and can be changed as and when necessary by the operator. These specifications are used to select an appropriate scheduling algorithm from those available in the Knowledge Base.

The scheduling algorithm is used to update the schedule for the use of communication links, based on some optimality criteria. The schedule thus created is used to control the communication system. The autonomous system tries to take care of anomalies in the expected system behaviour as best it can.

The modules have the following functions:

- CONTROL: Generates formatted schedule or simulation request, or generates formatted control directive.
- PLANNER: Gives high level directives for the Executor to accomplish the goals (e.g., chooses algorithm that better fulfills established policies). It also evaluates the effect of an anomaly, based on the estimate of its cause (as computed by the Diagnoser, and kept in the Knowledge Base), and tries to overcome the problem by replanning (e.g., marks a device as "not available" in the Knowledge Base's world state description, and see that all tasks involving that device

be rescheduled by the Executor). If it is not possible to replan, sends an alert to the operator through Simulator.

Planner additionally passes schedule requests on from Controls to Executor, and passes simulation requests on from Controls to Simulator.

- SIMULATOR: Computes the effect of a simulation request on the real or a given world, generating a series of nominal world state changes. It can, for instance, simulate the effect of applying a certain policy to reschedule all the requests in the current schedule. Simulator additionally passes alert signal on from Planner to Display.

- DISPLAY: Formats displays for the operator (e.g., shows in a screen a chart with a certain schedule).

- EXECUTOR: Tries to attend to the communication requests, applying the algorithm defined by the Planner, in a way to optimize the use of the resources available (according to some criteria). It generates commands which are comprised of preconditions, channel to be used, start time, and duration of transmission. These commands are sent to the Effector, and a copy of them is sent to the Real Time Simulator.

- EFFECTOR: Enforces preconditions (e.g., adjusts the orientation of an antenna from one position to another), initiates transmissions, closes transmissions, etc.

- PERCEPTOR: Constantly observes the state of each device, and the status of all ongoing transmissions, and sends this information to the State Space Estimator.

- STATE SPACE ESTIMATOR: Elaborates on the information gathered by the Perceptor to synthesize a picture of the real world. Basically, it identifies:

  1. transmissions started or finished,
  2. transmissions interrupted due to problems,
  3. devices under normal operation conditions or down,

4. current earth view and orbiter view, and so on.

- REAL TIME SIMULATOR: Infers what to expect as result of applying commands to the real world. It can conclude, for instance, that in one minute an antenna must be in a certain position and executing a communication, whereas another device must be in the READY mode.

- MONITOR: Compares the states of transmissions and devices furnished by the State Space Estimator to those provided by the Real Time Simulator. Compares earth view and orbiter view to the the expected. Minor problems in a transmission will generate a request to the Executor from Monitor to reschedule it. Transmissions that have repeatedly failed, devices which do not respond satisfactorily to primitives issued by the Effector, or changes in the Earth/Orbiter view will produce alert signals to the Diagnoser.

- DIAGNOSER: Tries to identify the cause of unusual occurrences in the system. For instance, repeatedly failing transmission can be due to demaged parts in an involved device.

# 4 Concluding Remarks

We have presented an application of a previously proposed architecture for autonomous systems. The example that we chose to illustrate was a simplified version of the Shuttle communication system.

We found that the application of the model to a real world situation was very helpful in clarifying the specific functions of each module, as well as the interfaces among them. It has also been shown that the proposed architecture is robust and can be easily adapted to any given problem that involves the use of an autonomous system.

SDAG TECHNICAL NOTES - 87-01 January 1987

# Dynamic Bin Packing Problem[1]

Piyush Dikshit
Katia Guimaraes
Maya Ramamurthy
Ashok Agrawala
Ronald L. Larsen

Systems Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742

[1]This research was supported in part by NASA under Grant NCC-2414

# Contents

**1 Introduction** 1

**2 Conceptual Framework** 1
- 2.1 Major Functions . . . . . 1
- 2.2 Control Loops . . . . . 5
- 2.3 Knowledge Base . . . . . 6

**3 Bin Packing Problem** 7
- 3.1 Environment Description . . . . . 7
- 3.2 Blocks & Bins . . . . . 7
- 3.3 Knowledge Base . . . . . 7
- 3.4 The Model . . . . . 8

**4 Concluding Remarks** 11

# 1 Introduction

In a previous work we have defined a general architectural model for autonomous systems, which can be mapped easily to describe the functions of any automated system(SDAG-86-01). In this note, we use the model to describe the problem of thermal management in space stations.

First we briefly review the architecture, then we present the environment of our application, and finally we detail the specific function for each functional block of the architecture for that environment.

# 2 Conceptual Framework

The conceptual framework consists of four major functions integrated into a unifying system architecture: 1. the operator interface, 2. task planning and reasoning, 3. sensing and perception, and 4. control execution. These functions are then organized to provide three control loops: 1. an operator control loop, 2. an executive control loop, and 3. a local control loop. The overall concept is illustrated in figure 1.

## 2.1 Major Functions

The operator interface includes displays and controls. Displays provide all of the information about system operation to the human operator, and can employ a variety of mechanisms for communicating information. Visual displays, such as graphics video display workstations, head-up displays, hand controllers or exoskeletons, can also be used, as can aural feedback and voice generation.

Controls provide the means by which the operator communicates information to the system, such as the nature or details of the task to be performed. Alphanumeric keyboards commonly provide this capability, often augmented with function keys, light pens, and similar mechanisms.

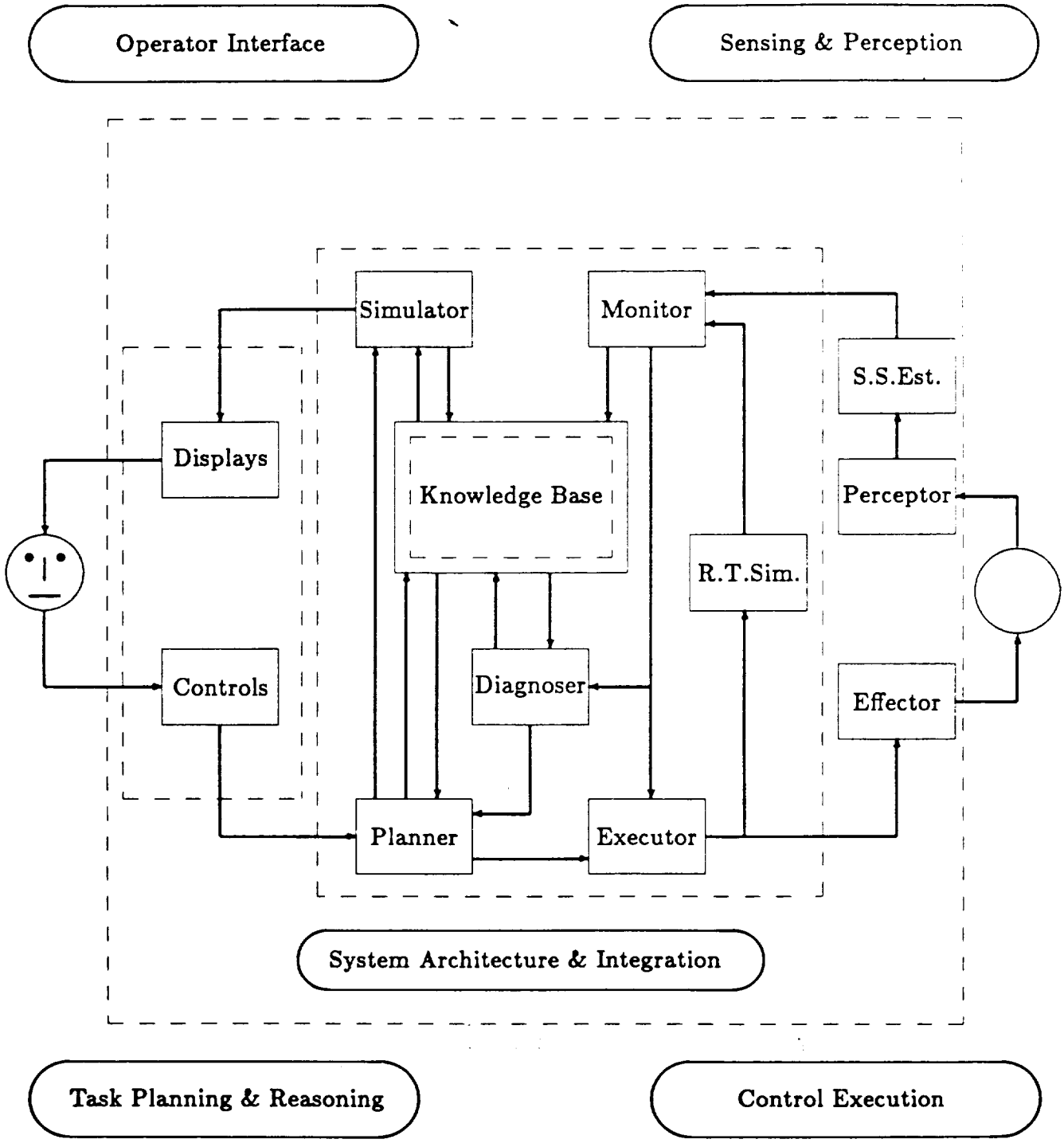


Figure 1. Automated System Control Architecture

The integration of displays with controls provides the environment for human interaction with the system. It provides the means by which the human operator can communicate the description of the task to be performed to the system, and the means by which the system can communicate the status of task execution back to the operator.

Task planning and reasoning includes most of the intelligence of the automated system. It receives the description of the task to be performed from the Controls, translates that into commands which can be remotely executed to change the state of the world, observes the changing world state, handles unexpected discrepancies in the world, seeking operator intervention only when it cannot cope with a world situation with the "intelligence" it has. Task planning and reasoning requires an extensive Knowledge Base containing all of the data required to represent and reason about the task domain. The major capabilities of task planning and reasoning are provided by seven functional modules: 1. Planner, 2. Simulator, 3. Executor, 4. Real Time Simulator, 5. the State Space Estimator, 6. the Monitor, and 7. the Diagnoser.

The Planner receives the task description from the Controls, and transforms the (typically) high level description of the task into a nominal plan of action. The plan is a time sequenced series of primitive actions which are used by the Executor for real-time operation.

The Simulator is a tool for the operator to carry out preliminary investigations of the effect of certain commands before they can be used in the real world. It gets simulation requests from the operator, a description of the real world from the Knowledge Base ( or from the operator himself if the command is to be given to a simulated world state) and computes the sequence of world state changes over time corresponding to the request. The simulation results may be sent to Displays, along with the Knowledge Base.

The Executor accepts the nominal plan of action from the Planner and issues primitive commands in real-time. It also receives minor adjustments, which are superimposed on the command sequence to fine tune the execution of the operation in response to unanticipated discrepancies between the expected state of the world and the actual state.

The Real Time Simulator computes Expectations about the Real World

in real time based on the commands issued by the Executor, the current world state in the Knowledge Base and a command script which describes the effect of commands on the Real World. These Expectations are used by the Monitor to detect discrepancies between what is observed and what is expected in the Real World.

The State Space Estimator uses the observation data provided by the Perceptor to derive a state description of the Real World, giving meaning to the raw information gathered by the Perceptor. The interpreted world state description is placed in the Knowledge Base also.

The Monitor uses expected values of state descriptors (computed by the Real Time Simulator) and the observed data in real-time to detect discrepancies between the expected world state and the actual world state. Non-goal-threatening discrepancies are transformed into adjustments which are fed back to the Executor. Goal-threatening discrepancies (such as failure of a required device, or detection of a broken component) require further analysis. An anomaly report is sent to the Diagnoser.

The Diagnoser receives anomaly reports from the monitor, and attempts to assess the nature of the anomaly, its cause, and potential effects. It computes an "inferred world state", i.e., the most plausible explanation and implied new system state, and places this in the knowledge base as the best estimate of the current world state. It then issues a replan request to the Planner, which informs the Planner that the nominal plan is no longer satisfactory. The Planner formulates a new nominal plan or alerts the operator if it cannot do so.

The sensing and perception function provides a mapping from measurable internal and external parameters to an estimate of the world state. Its input is typically acquired from a multi-mode array of tasks and world sensing devices. The objective of the sensing and perception function is to provide the Monitor with a real-time best estimate of the state of the task domain in terms which are consistent with the expected values provided by the Real Time Simulator.

The control execution function includes the manipulators and effectors, and tools available at the task site which can be used to accomplish the desired task, as well as the low level controls required to transform primitive commands issued by the Executor into state-changing actions.

## 2.2 Control Loops

The automated system control architecture illustrated consists of three major control loops. The Operator Control loop includes the human operator plus the Controls, Planner, Simulator, and Displays. This loop provides the interaction between the human operator and the system. It can operate independently, providing a pure system simulation capability. This capability is desirable for operator training or system evaluation, for example.

The Executive Control loop includes all of the components required to operate and control a subsystem with minimal assistance from a human operator: the Planner, Executor, Real Time Simulator, Monitor, Diagnoser, and Knowledge Base. This core loop provides a generic control architecture which is equally applicable to controlling a deep space spacecraft or an autonomous vehicle as it is to a robotic system.

The Local Control loop, consisting of the Executor, Real Time Simulator, Effector, Perceptor, State Space Estimator and Monitor, provides the low level control for the specific manipulators used in the robot, as well as the sensorbased information extraction algorithms.

It is important to realize that these three control loops do not work in the same time-frame. The Local Control Loop works in real time as it needs to keep up with the changes in the real world. The Operator Control Loop, on the other hand, is used only for high level planning and simulation/testing activities. In a truly autonomous system, it may have little to do with the normal run-time functioning of the rest of the system. The Executive Control Loop falls somewhere in between. It is the core of the "intelligence" exhibited by the autonomous system. Its right half (Executor, Real Time Simulator and Monitor) operates in real time. Only when there is an anomaly in the real time operation does the central part of the loop (Diagnoser) become active. The Diagnoser evaluates the anomalous condition and generates the necessary action requests for the Planner. Human intervention will be necessary if the Planner cannot handle the anomaly. The frequency with which operator help is sought determines how autonomous the system really is, and other operating conditions of the system.

## 2.3 Knowledge Base

The Knowledge Base is the repository of all information in the system. Some of the information is about the real time operation of the system (and hence used in the Local Control Loop), some is used for planning and evaluation (by the Executive Control Loop) and some is to aid the operator in testing and simulation activities (in the Operator Control Loop). All audit trail information is also saved in the Knowledge Base and is available to different components of the system, including the operator, as necessary. All access to the stored data is centralized using the Knowledge Base Manager.

It contains the following information:

- System Configuration
- Constraints
- Command Script (for Executor commands)
- Real World State Description (provided by State Space Estimator)
- A set of policies/algorithms for the Planner to choose from
- Objective Function (representing the long-term goal of the system)
- Statistical Information (for performance evaluation)
- Test/Diagnostic/Simulation Tools for the operator
- History of the system
- Explanation for anomalies (provided by Diagnoser when needed)

The Knowledge Base Manager is responsible for ensuring the consistency and integrity of the information in the Knowledge Base. It also acts as the interface between two or more independent autonomous systems.

# 3 Bin Packing Problem

In this section we apply the model for the autonomous system described above to an illustrative example.

## 3.1 Environment Description

The motivation for this example comes from the problem of thermal management in space stations. On the space station there are many sources of heat which may be dissipated using several facilities available. The heat dissipation capability of each facility is limited, and may depend on parameters such as the orientation of the space station. In thermal management, one has to assign heat sources to the dissipation facilities. In this example, we abstract the major characteristics of the thermal management problem in the form of dynamic blocks and multidimensional bins.

## 3.2 Blocks & Bins

In this example, the blocks represent the heat generated, and the bins provide the mechanism for dissipation of heat.

The blocks and the bins have well defined properties, like size, shape etc. In the most general case, both blocks and bins can be n-dimensional. Each block has a given growth characteristic and each bin has a given shrinking capacity. The blocks are generated in a deterministic or stochastic manner. The problem is to pack these blocks into the bins as they are generated in real time.

It is desirable to carry out this bin-packing in an "efficient" manner. The criteria for efficiency is for the users of the system to define. For example, it may be desired that the ratio of the number of blocks packed to the number of blocks arriving per unit time is as high as possible.

## 3.3 Knowledge Base

The Knowledge Base contains the following :

- A timestamped description of the state of the blocks and the bins with their respective sizes.

- Copy of commands issued by Executor.
- A set of bin-packing algorithms for the Planner to choose from.
- Aids to help Diagnoser analyse anomalies.
- Analysis of anomalies created by the Diagnoser.
- Heuristic information for the Planner to aid in selecting the appropriate strategy for bin packing.
- Statistics to evaluate the performance of the algorithm in use.
- Historical trace of state of bins and blocks.
- Simulation tools for the operator.

## 3.4 The Model

The role of the system components is described in what follows. The blocks are generated by a Block Generator (which is needed only to simulate a source for the blocks and is not part of the autonomous system). They "arrive" in the Real World which consists of a number of bins placed on a table. The blocks are put on the table by the Block Generator where they keep growing until they are packed. On being packed, they begin to shrink at a rate depending on the bin they are placed into, until they finally disappear.

The Perceptor keeps giving a picture of the Real World to the State Space Estimator periodically. This picture may be a lower dimensional projection of the real world, and may be subject to noise corruption. The State Space Estimator interprets the picture to update the current world state in the Knowledge Base. The Monitor looks at the current world state and on detecting the presence of fresh blocks, it informs the Executor. The Executor uses the nominal plan specified by the Planner to determine which blocks to pack first, and where to put them (i.e. the bin chosen to pack the blocks and the position of the blocks within the bins).

The roles of different parts of the system are:

- CONTROL: Provides the interface for the operator to communicate with the system. It is used to
  - specify the criteria to be used in choosing an appropriate bin packing algorithm.
  - inquire about the current state of the blocks and bins.
  - input commands to try out new strategies for bin-packing.
  - handle unexpected anomalies in the system which cannot be handled by the Planner.
- PLANNER: Specifies the information that the Executor needs to know to pack the blocks in the bins. It generates a nominal plan which is used by the Executor to decide:
  1. which block to pack next
  2. how to choose the bin for a given block
  3. the bin-packing algorithm (with parameters, if necessary)

  On a signal from the Diagnoser to Replan, it may change the packing strategy currently in use. To decide on how to alter the strategy it uses the description of the anomaly and heuristic information on how to handle it from the Knowledge Base. If it is unable to handle the anomaly, it informs the operator.
- SIMULATOR: Used by the operator to test new bin-packing strategies.
- DISPLAYS: Formats displays for the operator. There could be tabular descriptions of the blocks and bins or figures showing the bins with blocks placed inside.
- EXECUTOR: It chooses one or more blocks for being packed next based on the policy specified by the Planner. It executes the bin-packing algorithm and issues commands to the Effector to place the block in a bin (for fresh blocks) or repack a block(if the Monitor decided that the block was not properly packed). It informs the Real Time Simulator about the command it has issued.

- EFFECTOR: Issues the primitives received from the Executor to the real world. Thus it physically removes the blocks from the table and puts them in the bin (or moves blocks from one part of the bin to another, if they are being repacked).

- PERCEPTOR: Passes information about the physical attributes of the blocks and bins to the State Space Estimator. It has no interpretational ability.

- STATE SPACE ESTIMATOR: Interprets the data from the Perceptor data to come up with a meaningful description of the state of real world. It updates the state of the blocks and the bins in the Knowledge Base. If there are new blocks it informs the Executor. If a block disappears from a bin, or the size of one or more blocks changes , it informs the Monitor.

- REAL TIME SIMULATOR: Uses the last state description of the blocks and bins and the commands issued by the Executor to compute the new dimensions of all the blocks. It also determines which blocks will disappear due to shrinkage. It passes these (timestamped) Expectations to the Monitor.

- MONITOR: Determines whether blocks that disappear are expected to disappear. It uses the Expectations computed by the Real Time Simulator to detect discrepancies in real time . If the disappearance of a block happens too early or too late, it requests the Executor to repack the block. If the difference between the real and expected size(s) of one or more blocks is greater than some specified tolerance value, it alerts the Diagnoser.

- DIAGNOSER: On getting reports of serious anomalies from the Monitor, it tries to determine the cause of the anomaly and places its analysis in the Knowledge Base. For instance, if the Monitor reports that some block has not disappeared at the time it was supposed to, the Diagnoser will attempt to explain it by analysing the effect of other factors in the Real World which could prevent the disappearnce

of the block. (There are aids in the Knowledge Base to enable the Diagnoser to know what to check in case a given anomaly is reported). It then signals the Planner to Replan.

# 4 Concluding Remarks

We have presented an application of a previously proposed architecture for autonomous systems. The example that we chose to illustrate was an abstraction of the problem of thermal management in space stations.

We purpose to design a testbed based on this example to develop an experimental autonomous system. We hope that it will help in clarifying the numerous interfaces among the modules that constitute the system. The testbed may enhance our understanding of the functions of these modules. It might also enable us to get a clearer picture of what constitutes the Knowledge Base.

SDAG TECHNICAL NOTES - 87-02 February 1987

# Testbed for an Autonomous System[1]

Piyush Dikshit
Katia Guimaraes
Maya Ramamurthy
Ashok Agrawala
Ronald L. Larsen

Systems Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742

[1]This research was supported in part by NASA under Grant NCC-2414

# Contents


**1 Introduction** 1

**2 Overview of Architectural Model** 1
- 2.1 Functional Description . . . . . 1
- 2.2 Knowledge Base . . . . . 4

**3 Bin Packing Problem** 5
- 3.1 Blocks & Bins . . . . . 5
- 3.2 Knowledge Base . . . . . 5
- 3.3 The Model . . . . . 6

**4 Testbed** 9
- 4.1 Assumptions . . . . . 9
- 4.2 Implementation . . . . . 9

**A Some Implementation Details** 13
- A.1 Real World . . . . . 13
- A.2 Generator . . . . . 14
- A.3 Perceptor . . . . . 14
- A.4 Monitor . . . . . 14
- A.5 Executor . . . . . 15
- A.6 Effector . . . . . 16
- A.7 Diagnoser . . . . . 16

# 1 Introduction

In previous works we have defined a general architectural model for autonomous systems, which can be mapped easily to describe the functions of any automated system (SDAG-86-01), and we illustrated that model by applying it to the thermal management system of a space station (SDAG-87-01). In this note, we will further develop that application and design the details of the implementation of such a model.

First we present the environment of our application, by describing the thermal management problem and an abstraction of the problem, as used in SDAG-87-01. Then we detail the implementation of such abstraction, which was called TESTBED. That includes specific function for each module in the architecture, and the nature of the interfaces between each pair of blocks.

# 2 Overview of Architectural Model

## 2.1 Functional Description

The overall architectural model is illustrated in figure 1.

The Planner receives the task description from the Controls, and transforms the high level description of the task into a nominal plan of action.

The Simulator is a tool for the operator to carry out preliminary investigations of the effect of certain commands before they can be used in the real world. It gets simulation requests along with a description of the real world from the Knowledge Base (or from the operator if the command is to be given to a simulated world state) and computes the sequence of world state changes over time corresponding to the request. The simulation results may be sent to Displays, along with the Knowledge Base.

The Executor accepts the nominal plan of action from the Planner and issues commands in real-time. It also receives minor adjustments, which are superimposed on the command sequence to fine tune the execution of the operation in response to unanticipated discrepancies between the expected state of the world and the actual state.

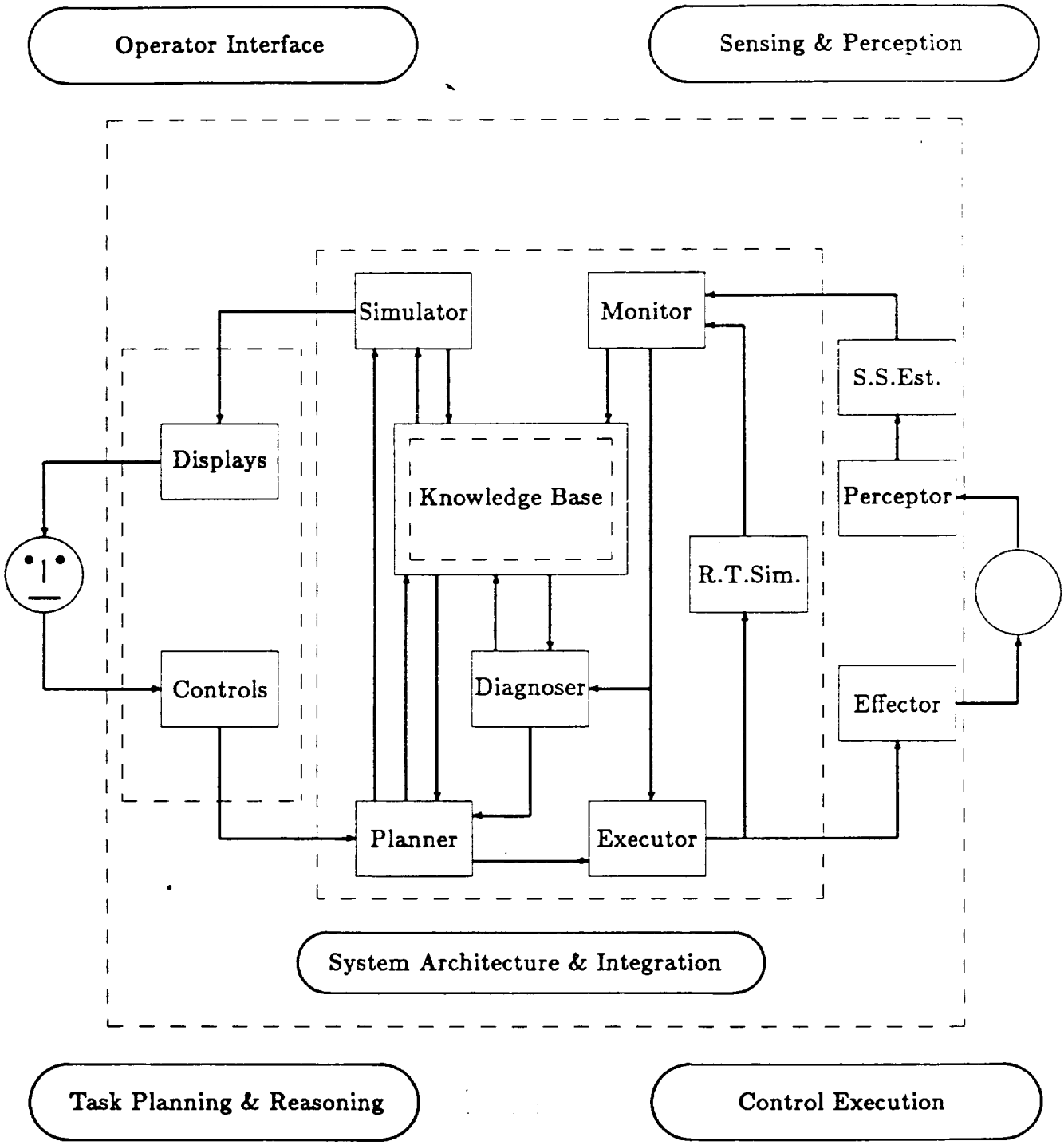


Figure 1. Automated System Control Architecture

The Real Time Simulator computes expectations about the Real World in real time based on the commands issued by the Executor, the current world state in the Knowledge Base and a command script which describes the effect of commands on the Real World. These expectations are used by the Monitor to detect discrepancies between what is observed and what is expected in the Real World.

The State Space Estimator uses the observation data provided by the Perceptor to derive a state description of the Real World, giving meaning to the raw information gathered by the Perceptor. The interpreted world state description is placed in the Knowledge Base also.

The Monitor uses expected values of state descriptors (computed by the Real Time Simulator) and the observed data in real-time to detect discrepancies between the expected world state and the actual world state. Non-goal-threatening discrepancies are transformed into adjustments which are fed back to the Executor. Goal-threatening discrepancies (such as failure of a required device, or detection of a broken component) require further analysis. An anomaly report is sent to the Diagnoser.

The Diagnoser receives anomaly reports from the monitor, and attempts to assess the nature of the anomaly, its cause, and potential effects. It computes an "inferred world state", i.e., the most plausible explanation and implied new system state, and places this in the knowledge base as the best estimate of the current world state. It then issues a replan request to the Planner, which informs the Planner that the nominal plan is no longer satisfactory. The Planner formulates a new nominal plan or alerts the operator if it cannot do so.

The sensing and perception function provides a mapping from measurable internal and external parameters to an estimate of the world state. Its input is typically acquired from a multi-mode array of tasks and world sensing devices. The objective of the sensing and perception function is to provide the Monitor with a real-time best estimate of the state of the task domain in terms which are consistent with the expected values provided by the Real Time Simulator.

The control execution function includes the manipulators and effectors, and tools available at the task site which can be used to accomplish the desired task, as well as the low level controls required to transform primitive

commands issued by the Executor into state-changing actions.

For more details on the architecture, refer to SDAG-01.

## 2.2 Knowledge Base

The Knowledge Base is the repository of all information in the system. Some of the information is about the real time operation of the system some is used for planning and evaluation and some is to aid the operator in testing and simulation activities. All audit trail information is also saved in the Knowledge Base and is available to different components of the system, including the operator, as necessary. All access to the stored data is centralized using the Knowledge Base Manager, which is responsible for ensuring the consistency and integrity of the information in the Knowledge Base.

It contains the following information:

- System Configuration
- Constraints
- Command Script (for Executor commands)
- Real World State Description (provided by State Space Estimator)
- A set of policies/algorithms for the Planner to choose from
- Objective Function (representing the long-term goal of the system)
- Statistical Information (for performance evaluation)
- Test/Diagnostic/Simulation Tools for the operator
- History of the system
- Explanation for anomalies (provided by Diagnoser when needed)

# 3 Bin Packing Problem

The motivation for this example comes from the problem of thermal management in space stations. On the space station there are many sources of heat which may be dissipated using several facilities available, each of which has limited capability. In thermal management, one has to assign heat sources to the dissipation facilities. In this example, we abstract the major characteristics of the thermal management problem in the form of dynamic blocks and multidimensional bins. The blocks represent the heat generated, and the bins provide the mechanism for dissipation of heat.

## 3.1 Blocks & Bins

The blocks and the bins have well defined properties. Blocks have two dimensions, and a growth rate. Bins have a given shrinking capacity. The blocks are generated in a deterministic or stochastic manner. The problem is to pack these blocks into the bins as they are generated in real time.

We can picture the real world as consisting of blocks and bins placed on a table. After being generated and put on the table, the blocks grow at different rates. The blocks are packed into the bins by the autonomous system. On being packed, the blocks start shrinking until they finally disappear. The shrinking capacity of the bins keeps changing as blocks get added or dissolved.

It is desirable to carry out this bin-packing in an "efficient" manner. The criteria for efficiency is for the users of the system to define. For example, it may be desired that the ratio of the number of blocks packed to the number of blocks arriving per unit time is as high as possible.

## 3.2 Knowledge Base

The Knowledge Base contains the following :

- A timestamped description of the state of the blocks and the bins with their respective sizes.
- Copy of commands issued by Executor.

- A set of bin-packing algorithms for the Planner to choose from.
- Aids to help Diagnoser analyse anomalies.
- Analysis of anomalies created by the Diagnoser.
- Heuristic information for the Planner to aid in selecting the appropriate strategy for bin packing.
- Statistics to evaluate the performance of the algorithm in use.
- Historical trace of state of bins and blocks.
- Simulation tools for the operator.

## 3.3 The Model

The role of the system components is described in what follows. The blocks are generated by a Block Generator (which is needed only to simulate a source for the blocks and is not part of the autonomous system). The Block Generator puts the blocks on the table where they keep growing until they are packed. On being packed, they begin to shrink at a rate depending on the bin they are placed into, until they finally disappear.

The Perceptor keeps giving a picture of the Real World to the State Space Estimator periodically. This picture may be a lower dimensional projection of the real world, and may be subject to noise corruption. The State Space Estimator interprets the picture to update the current world state in the Knowledge Base. The Monitor looks at the current world state and on detecting the presence of fresh blocks, it informs the Executor. The Executor uses the nominal plan specified by the Planner to determine which blocks to pack first, and where to put them (i.e. the bin chosen to pack the blocks and the position of the blocks within the bins).

The roles of different parts of the system are:

- CONTROL: Provides the interface for the operator to communicate with the system. It is used to
  - specify the criteria to be used in choosing an appropriate bin packing algorithm.

- inquire about the current state of the blocks and bins.
- input commands to try out new strategies for bin-packing.
- handle unexpected anomalies in the system which cannot be handled by the Planner.

- PLANNER: Specifies the information that the Executor needs to know to pack the blocks in the bins. It generates a nominal plan which is used by the Executor to decide:

  1. which block to pack next
  2. how to choose the bin for a given block
  3. the bin-packing algorithm (with parameters, if necessary)

  On a signal from the Diagnoser to Replan, it may change the packing strategy currently in use. To decide on how to alter the strategy it uses the description of the anomaly and heuristic information on how to handle it from the Knowledge Base. If it is unable to handle the anomaly, it informs the operator.

- SIMULATOR: Used by the operator to test new bin-packing strategies.

- DISPLAYS: Formats displays for the operator. There could be tabular descriptions of the blocks and bins or figures showing the bins with blocks placed inside.

- EXECUTOR: It chooses one or more blocks for being packed next based on the policy specified by the Planner. It executes the bin-packing algorithm and issues commands to the Effector to place the block in a bin (for fresh blocks) or repack a block(if the Monitor decided that the block was not properly packed). It informs the Real Time Simulator about the command it has issued.

- EFFECTOR: Issues the primitives received from the Executor to the real world. Thus it physically removes the blocks from the table and puts them in the bin (or moves blocks from one part of the bin to another, if they are being repacked).

- PERCEPTOR: Passes information about the physical attributes of the blocks and bins to the State Space Estimator. It has no interpretational ability.

- STATE SPACE ESTIMATOR: Interprets the data from the Perceptor to come up with a meaningful description of the state of the real world. It updates the state of the blocks and the bins in the Knowledge Base. If there are new blocks it informs the Executor. If a block disappears from a bin, or the size of one or more blocks changes, it informs the Monitor.

- REAL TIME SIMULATOR: Uses the last state description of the blocks and bins and the commands issued by the Executor to compute the new dimensions of all the blocks. It also determines which blocks will disappear due to shrinkage. It passes these (timestamped) Expectations to the Monitor.

- MONITOR: Uses the Expectations computed by the Real Time Simulator to detect discrepancies in real time. If there is a difference between the real and the expected size of a block, or if the disappearance of a block happens too early or too late, it either requests the Executor to repack the block or alerts the Diagnoser about the problematic block.

- DIAGNOSER: On getting reports of serious anomalies from the Monitor, it tries to determine the cause of the anomaly and places its analysis in the Knowledge Base. For instance, if the Monitor reports that some block has not disappeared at the time it was supposed to, the Diagnoser will attempt to explain it by analysing the effect of other factors in the Real World which could prevent the disappearnce of the block. (There are aids in the Knowledge Base to enable the Diagnoser to know what to check in case a given anomaly is reported). It then signals the Planner to Replan.

# 4 Testbed

## 4.1 Assumptions

Throughout this implementation, the following points will be followed as guidelines:

1. The interface between the real world and the Autonomous System is through the observable information of the Perceptor. The Effector is the only component of the autonomous system which can carry out defined functions in the real world.

2. The real world is simulated by the appropriate data structure and a process which interfaces with the Perceptor and the Effector of the autonomous system. It also interfaces with the Generator and the Disturber, which are considered as exogenous entities.

3. All module interfaces are asynchronous.

4. Communication between modules is accomplished through message passing, using a common IPC facility. Each message is timestamped, and a copy of each one is kept in the Knowledge Base.

5. Unless it is clearly said otherwise, all times are logical, and provide a partial order involving related events.

## 4.2 Implementation

What follows is a description of the system modules and interfaces. Detailed specification of some of these modules can be found in appendix A.

- CONTROL

  FUNCTIONS - The main function of the control is to provide the interface necessary for the operator sitting in front of the terminal display. It will provide the communications capability for the operator to any of the modules of the system. The link with the Planner will be used for specifying the policies. It will talk to the Diagnoser to

provide resolution to the contingencies which can not be handled by the Planner.

It is able to carry out query functions on the Knowledge Base, and also to request specific simulations.

The system should be operable in a teleoperation mode, in which all decisions are taken by the operator. In this case, the Control will be linked directly to the Effector and the Perceptor will be connected to the Display.

INS - Operator command

OUTS - Inquiries or control commands.

- PLANNER

  FUNCTIONS - Specifies the information that the Executor needs to know for packing the blocks in the bins. It chooses the bin-packing algorithm to be used and specifies:

  - Policy regarding which block is to be packed next
  - Policy regarding which bin has higher priority
  - Any other parameters necessary for the chosen algorithm.

  INS - Inquiries or control commands

  OUTS - Strategy for bin packing

- EXECUTOR

  FUNCTIONS - Responsible for carrying out the policy defined by the Planner for the bin packing. It receives information from the Monitor and generates the action-type commands for the Effector.

  A copy of each command is sent to the Real Time Simulator, so that it can compute the expectations.

  INS - New block description;
  Minor mismatch report

OUTS - Commands to place (or remove) blocks in (or from) the bins. (It also puts statistical data for performance evaluation in KB)

- REAL TIME SIMULATOR (RTS)

  FUNCTIONS - Responsible for computing expectations for Monitor.

  INS - Request for expectations at a given time T.

  OUTS - Expectations at time T.

- EFFECTOR

  FUNCTIONS - Carry out the commands issued by the Executor. This is the only entity in the autonomous system which can change the real world.

  INS - Block placement and removal commands

  OUTS - Changes to real world

- PERCEPTOR

  FUNCTIONS - Gets information about the real world and gives it to the State Space Estimator. It has very little processing or interpretational capabilities, acting as a passive observer. It operates both continuously and under the control of the Monitor which may ask it to acquire a specific piece of information.

  INS - Requests for description of real world
  Picture of real world

  OUTS - Real world description

- STATE SPACE ESTIMATOR

  FUNCTIONS - Transforms a real world description into more meaningful specification of the real world state, using information in the knowledge base. e.g. interprets a two dimensional world view into a three dimentional state description.

  INS - Real world description

OUTS - Interpreted real world state description

- MONITOR

  FUNCTIONS - It provides the Executor with the block descriptions. It compares information regarding the expected versus the real world description. On finding discrepancies it checks the severity. "Minor" problems are reported to the Executor. "Major" problems are reported to the Diagnoser. It also interfaces with the Real Time Simulator to get the expected values for the world state.

  INS - Real world state description
  Expected world state description

  OUTS - Reports of mismatches between the real and expected world states.

- DIAGNOSER

  FUNCTIONS - It is responsible for analysing anamolies and generating actions for the Planner, Executor or the Display.

  INS - Description of anomaly

  OUTS - Generates actions to take care of anomaly

- SIMULATOR

  FUNCTIONS - At the request of the operator it is responsible for carrying out the simulation of the real world. The basic inputs for the simulations may be obtained from the KB or as Whatif's from the operator. The outputs and results of the simulation are displayed as well as stored in the KB. They may also be sent to the Monitor at the request of the operator.

  INS - Operator requests

  OUTS - Results of simulations

# A Some Implementation Details

## A.1 Real World

It is represented by a monitor comprised of the following:

DATA STRUCTURES:

- LastUpdateTime
- Blocks not yet packed, represented by a list whose nodes contain: BlockNo, InitHeight, InitWidth, TimeGen, GrowRate
- State of the two bins, for each bin – a record with: BinHeight, BinWidth, BinDissolvRate, list of blocks currently in the bin. Each node of the list of blocks contains: BlockNo, Height, Width, Coord1, Coord2, TimeBin

PROCEDURES:

- add-table-list (block-description)
- add-bin-list (BlockNo, Height, Width, Coord1, Coord2, TimeBin, BinNo)
- del-table-list (BlockNo)
- del-bin-list (BlockNo, BinNo)
- copy-RW-list ()
- change-table-list (block-description)
- change-bin-list (block-description, BinNo)
- update-RW-blocks ()

## A.2 Generator

This block is not properly part of the Autonomous System.It is in charge of generating new blocks and introducing them to the Real World. It works at a certain rate, which can be changed through the keyboard.From time to time (After a given number of clock interruptions), the Generator will create random values (inside a given range) for: InitHeight, InitWidth, GrowRate. The current logical time is assigned to TimeGen. These values are associated to the next available BlockNo. The tuple formed by (BlockNo, InitHeight, InitWidth, TimeGen, GrowRate) is then used as a parameter for the add-table-list procedure.

## A.3 Perceptor

Works both, continuously (at a given rate), or on demand from the operator.

On activation :

1. Issues copy-RW-list
2. Delays until receiving the answer to the request
3. Forwards the copy to the Screen

PROCEDURE :

Reports the description of part or the whole real world, along with the time of observation.

## A.4 Monitor

Works continuously to provide control over the real world state. For that, Monitor uses a "retry-flag" for each block. When a block is generated, its retry-flag is OFF. A first problem with a block will be treated by the Monitor, and will cause its retry-flag to become ON. A second problem with the same block can then be identified, and will be considered as a serious anomaly, thus causing a signal to be sent to the Diagnoser.

PROCEDURE :

Receives expected world state (at time T) from Real Time Simulator. Receives the real world state (at time T) from State Space Estimator. Compares the two states.

If there is a mismatch for block BlockNo,

- Then
- If the retry-flag for that block is off
- Sends Redo(BlockNo, BinNo) to Executor
- Else
- Sends (BlockNo, BinNo) to Diagnoser
- Else
- Updates real world description in KB (removing blocks which
- disappeared, for instance).

## A.5 Executor

PROCEDURE 1 :

If Redo, then:

1. removes BlockNo from nominal real world (KB);
2. issues a Remove(BlockNo, BinNo) command to Effector;
3. retrieves original request for BlockNo from KB;
4. marks this block as retried;
5. places a copy of original request in list of blocks waiting to be packed by Executor (exec-list).

If New, then: Add incoming blocks to exec-list.

PROCEDURE 2 :

Whenever exec-list is not empty,

1. decides which block is going to be packed next, which bin it should go in, and where in the bin it should be placed
2. deletes chosen block from exec-list, and

3. issues Place command to Effector.

A copy of Executor's Ins's and Out's must go to whoever will evaluate the system performance.

## A.6 Effector

On receiving a Place command :

1. Computes TimeBin = f(CurrTime)
2. Computes Height = f(InitHeight, GrowRate, CurrTime, TimeGen)
3. Computes Width = f(InitWidth, GrowRate, CurrTime, TimeGen)
4. Computes BinDissolvRate = f(BinCapacity, AreaPack)
5. Computes AreaPack = f(AreaPack, Height, Width)
6. Issues delete-table-list(BlockNo)
7. Issues add-bin-list(BlockNo, Height, Width, Coord1, Coord2, Time-Bin, BinNo)

On receiving an EraseTable command :

- Issues delete-table-list(BlockNo)

On receiving an EraseBin command :

- Issues delete-bin-list(BlockNo, BinNo)

## A.7 Diagnoser

PROCEDURE :

Can do one of the following :

- Send a signal to Planner, in order to change policy/algorithm.

- Erase the block's description from the expected and real state and signal the Executor to redo packing for the original block.
- Cause an alarm to be displayed to the operator reporting the malfunction.