Engineering Systems with Intelligence

Concepts, Tools and Applications

edited by

SPYROS G. TZAFESTAS

National Technical University of Athens,
Department of Electrical and Computer Engineering,
Athens, Greece

KLUWER ACADEMIC PUBLISHERS
DORDRECHT / BOSTON / LONDON

Library of Congress Cataloging-in-Publication Data

Engineering systems with intelligence : concepts, tools, and applications / edited by Spyros G. Tzafestas.
p. cm. -- (Microprocessor-based and intelligent systems engineering ; v. 9)
Papers presented at the European Robotics and Intelligent Systems Conference (EURISCON '91) held in Corfu, Greece, June 23-28, 1991.
Includes index.
ISBN 0-7923-1500-6 (HB : acid-free paper)
1. Intelligent control systems--Congresses. 2. Artificial intelligence--Congresses. 3. Expert systems (Computer science)--Congresses. I. Tzafestas, S. G., 1939- . II. European Robotics and Intelligent Systems Conference (1991 : Kerkyra, Greece)
III. Series.
TJ217.5.E54 1992
629.8--dc20 91-35794

ISBN 0-7923-1500-6

Published by Kluwer Academic Publishers,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Kluwer Academic Publishers incorporates
the publishing programmes of
D. Reidel, Martinus Nijhoff, Dr W. Junk and MTP Press.

Sold and distributed in the U.S.A. and Canada
by Kluwer Academic Publishers,
101 Philip Drive, Norwell, MA 02061, U.S.A.

In all other countries, sold and distributed
by Kluwer Academic Publishers Group,
P.O. Box 322, 3300 AH Dordrecht, The Netherlands.

Printed on acid-free paper

MODEL-BASED ARCHITECTURE FOR HIGH AUTONOMY SYSTEMS*

B.P. Zeigler, S.D. Chi and F.E. Cellier

AI-Simulation Group
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, 85721

Abstract

This paper presents the principles for design of autonomous systems whose behavior is based on models that support the various tasks that must be performed. We propose a model-based architecture aimed at reducing the computational demands required to integrate high level symbolic models with low level dynamic models. Model construction methods are illustrated to outfit such an architecture with the models needed to meet assigned objectives.

1. Introduction

Emerging from the control field, *intelligent control* is viewed as a new paradigm for solving control problems [1]. Its relatively narrow interpretation of the "control problem" does not include for example, the control needed by a system to diagnose and repair itself after significant insults to its physical structure. However, requiring greater degrees of autonomy from a system forces a more expanded view. Antsaklis *et al.*[2] developed a more inclusive framework for autonomous control systems. To achieve autonomy, artificial intelligence is one, among other approaches.

Although, criteria for artificial intelligence [3,4] implicitly include autonomy, robotics research is realizing that autonomy requires the integration of AI decision making components together with perception and action components [5]. Even some AI researchers state that problems in AI itself might be better solved by incorporating solutions from perception and motor control [6]. To include such diverse systems as planetary colonies, self-operating factories, and telerobotic laboratories along with autonomous land and space vehicles in the autonomy umbrella, we employ a definition [7]:

Autonomy is the ability to function as an independent unit or element over an extended period of time, performing a variety of actions necessary to achieve pre-designated objectives while responding to stimuli produced by integrally contained sensors.

Saridis [8] developed a three layer hierarchy (execution, coordination and management) for intelligent control [8] which is supposed to reflect increasing intelligence with decreasing precision. Antsaklis *et al.* [2] refine the hierarchy to an arbitrary number of layers, depending on the particular application. The coupling of control and information at

* Research supported by NASA-Ames Co-operative Agreement No. NCC 2-525, "A Simulation Environment for Laboratory Management by Robot Organization".

various layers characterizes the framework recently proposed by Albus [9,10,11]. Generalizing the earlier NASREM framework, Albus elaborates an architecture of seven levels, each of which has its own Task Decomposition and Execution, Sensory Processing, World Model, Global Memory, and Value Judgement components. Components at one level communicate with each other and with corresponding components at higher and lower levels. The Albus architecture is the most general and elaborate attempt to integrate perception, decision and action to achieve intelligence/autonomy. However, it leaves many details unspecified, particularly of interest here, the role of models.

The alternative architecture for autonomous robotics developed by Brooks [12] does not explicitly incorporate higher level planning and world model perception. This approach by-passes the computational burden required in the multi-layered architectures. Although achievements such as robotic "insects" are impressive, it is not clear whether such an approach "scales up" to systems which for example, operate sophisticated equipment and are able to detect, diagnose and repair faults in such equipment. However, the Brooks alternative makes clear that one major hurdle to be overcome in the architectural concepts of Albus, Saridis, Antsaklis, etc. is the heavy on-line computation times needed for higher level functions such as planning and diagnosing. Our model-based architecture seeks to reduce such on-line computation by off-line pre-compilation to produce simplified models individually matched to the tasks needing to be performed. Such models are to be attached to generic engines that can interpret them efficiently with reduced processing times.

In the proposed *model-based architecture*, knowledge is encapsulated in the form of models that are employed at the various control layers to support the predefined system objectives. The term "model" here refines the "world model" concept: "the system's best estimate of objective reality" [9]. It recognizes that an autonomous system must maintain models in a variety of formalisms and at various levels of abstraction. Lower control layers are more likely to employ conventional differential equation models with symbolic models more prevalent at higher layers. A key requirement is the systematic development and integration of dynamic and symbolic models at the different layers. In this way, traditional control theory, where it is applicable, can be interfaced with AI techniques, where they are necessary. Structure and behavior preserving morphisms from model theory [13,14,15] can connect models at different levels of abstraction so that they can be developed to be consistent with each other and can be consistently modified. An organized model base enables the agent to deal with the multiplicity of objects and situations in its environment and to link its high level plans with its actual low level actions through well-defined morphism mappings [16].

2. Model Formalisms

There are three major formalisms, or short-hands for dynamic system specification. Differential and difference equations employed to describe continuous systems have a long history of development whose mathematical formalization came well before the advent of the computer. Discrete event modelling is of more recent vintage and is finding ever more application to analysis and design of complex manufacturing, robotic, communication, and computer systems among others [17,18]. In contrast to continuous systems simulation, discrete event simulations were made possible by, and evolved with, the growing computational power of computers. Formalization of the models that underlay discrete event simulations is therefore a relatively new concern [13,19]. Formalisms that can capture both discrete and continuous behavior are also of increasing importance [20,21,22].

The classical AI knowledge representation schemes, such as rules, semantic nets, and frames, can be viewed as essentially symbolic formalisms for static models. They provide organizations for large bodies of facts with mechanisms for inferencing, access path development, and comparison based on symbol manipulation. New directions in AI, for example, qualitative modelling and planning, are finding it necessary to take time into account [23,24].

On the other hand, while differential/difference equation formalisms are numeric in character, this is not necessarily true of more recent dynamic systems formalisms. Discrete event and automata models for example, usually include state representations of a symbolic nature. Antsaklis *et al.* [2] refer to discrete event models that do not include a time base explicitly [25] as "logical", thereby emphasizing their symbolic nature. Narain [22] has developed a logic-based formalism intended to facilitate reasoning about the behavior of combined discrete/continuous models.

Qualitative physics has done much to bring qualitative formalisms for dynamic model specification to attention. AI researchers in qualitative physics envision an ideal formalism which captures commonsense knowledge of dynamic systems without the use of differential equations [26,27,28,29,30]. The ability of such models to deal with incomplete specification is emphasized, although the cost is heavy ambiguity and trajectory branching. Model-based diagnosis [29,31,32,33] is a related approach that seeks to support deeper reasoning using models than possible in conventional "shallow" expert system approaches.

Traditional discrete event formalisms incorporate uncertainty in the form of stochastic processes. Modern views recognize kinds of uncertainty that cannot be represented in a probabilistic manner such as ambiguity, lack of relevant factual knowledge, and deliberate use of linguistic imprecision. Fuzzy set theory is the predominate framework for representing such uncertainty (although, it is not the only one). Incorporating fuzzy set mechanisms into models is drawing increasing attention [34,35].

For autonomous systems design, any, and all, of such formalisms are potentially applicable. Conventional automatic control theory makes heavy use of differential/difference equation models to capture continuous interaction with the world at the lower physical level ("physical" here includes aerodynamic, chemical, biological, ecological and other processes). Planning and scheduling of manufacturing and other operations rely on simulation of discrete event models. It is reasonable to seek to achieve high autonomy by retaining such existing model formalisms and associated simulation engines as a basis, introducing newer symbolic and qualitative formalisms as needed.

The event-based control methodology [15,36] provides an approach to process and device control which is especially tuned to the constraints of high-autonomy applications. It provides a relatively simple and robust real-time control layer that can be homomorphically linked to higher level symbolic reasoning layers.

3. Model Base Development

Figure 1 presents a model-based autonomous system architecture to integrate decision, action, and prediction components. The architecture features a model base at the center of its planning, operation, diagnosis, and fault recovery strategies. In this way, it integrates AI symbolic models and control-theoretic dynamic models into a coherent system.

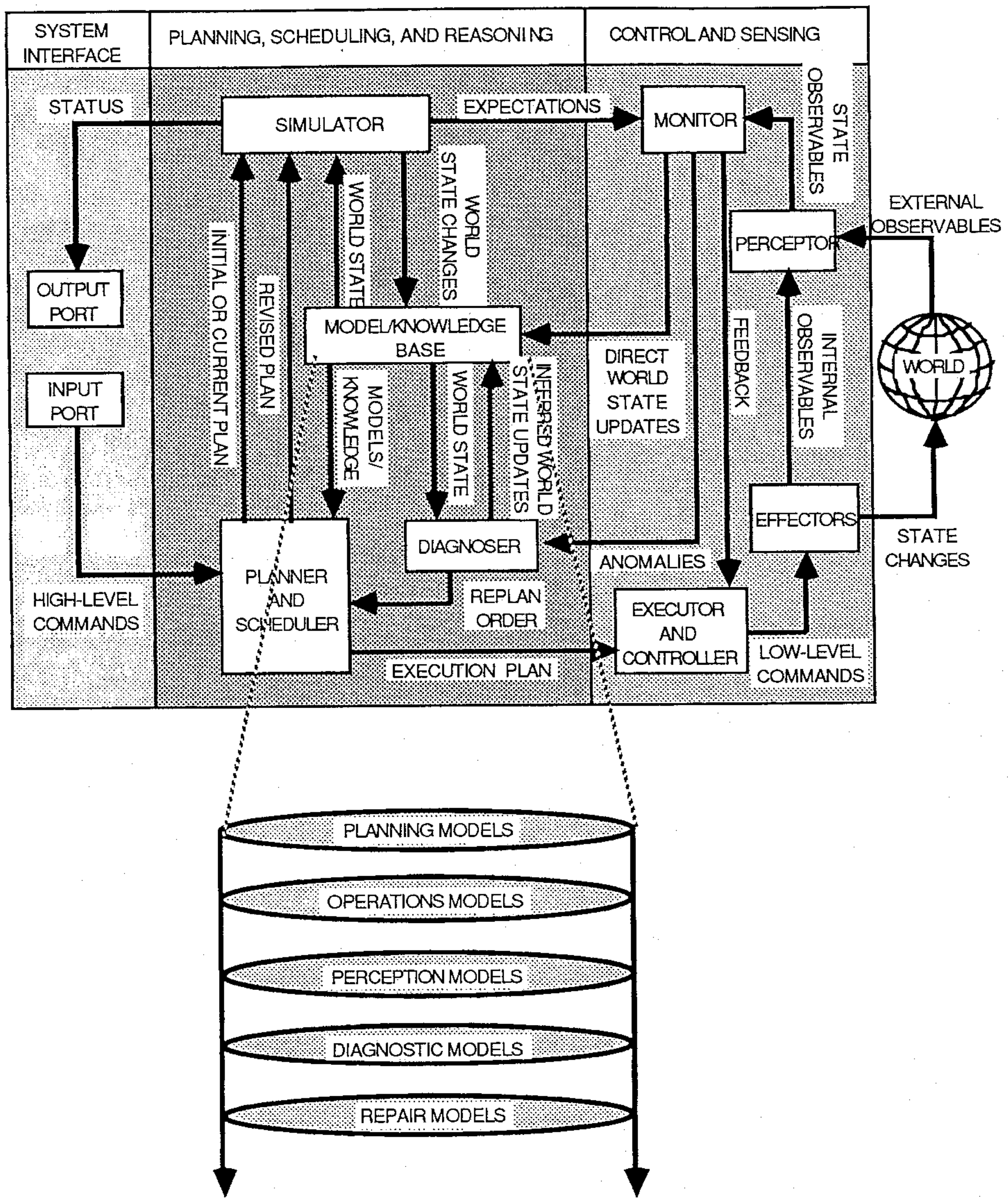


Figure 1. Autonomous System Architecture and its Model Base Class

Approaches to develop various autonomous component models for planning, operation, diagnosis, and fault recovery have been developed in their respective research fields so that there are many overlaps as well as inconsistencies in assumptions. In an integrated system, such components cannot be considered independently. For example, planning requires execution, and diagnosis is activated when abnormalities are detected during execution.

Planning is defined as "Reasoning how to achieve a given goal". It employs a suitable model to map from initial and goal states to a correcting sequence of states within a normal operation envelope. Once a plan is set up, it should be faithfully executed. "Execution with verification" maps from the input command and its expected normal responses to success/failure. As long as execution is successful, it continues until achieving the goal. But, if it fails, the diagnosis function will be activated. Diagnosis is defined as "Discovering the cause of the failure". It maps from observed symptoms to plausible abnormalities. Having identified the causes, the autonomous system should be able to recover, i.e., plan a path from the faulty state of the real system to a normal state on the original plan.

3.1 Model-based Planning

There are two major approaches in task planning: one considers planning as searching and the other considers planning as a representation problem. The former deals with the initial planning problem where no prior experience is employed. In contrast, the latter, so-called case-based planning, views planning as remembering, i.e., retrieving and modifying existing plans for new problems [37].

Figure 2 depicts a planning approach to build autonomous execution structures, where planning is viewed as a pruning operation which generates a candidate structure. In our model-based approach, planning is achieved by pruning a System Entity Structure (SES) to select Pruned Entity Structures (PESs) from alternatives [39]. The PESs are in turn transformed into simulation model structures for execution. The non-experienced initial planning, which means the pruning of SES alternatives, can be achieved by using a rule-based approach. Every action (or state) node has several rules associated with system constraints, pre-conditions, and post-conditions. The resultant PES is saved with an index into an entity structure base (ENBASE) for reuse. In contrast with non-experienced planning, the experienced planning is done by retrieving PESs from the ENBASE. The planner first retrieves a plan that might be used to achieve a given goal, or generates a new trial plan from partial plans if no existing plan is suitable. This candidate plan is then projected forward via simulation by attaching component models in a model base (MBASE), where low level planning is embedded.

Once an execution model is synthesized, a lower level planner produces a goal table that is a list of 4-tuples: state, goal, command, and time-to-reach-goal. From these, a time optimal path from an initial state to a goal state is readily derived. Since discrete event models embody timing it is natural to base optimal sequencing on predicted execution time. The planner works by developing paths backward from the goal until the given initial states (possible starting states of the given system) are reached[15,40].

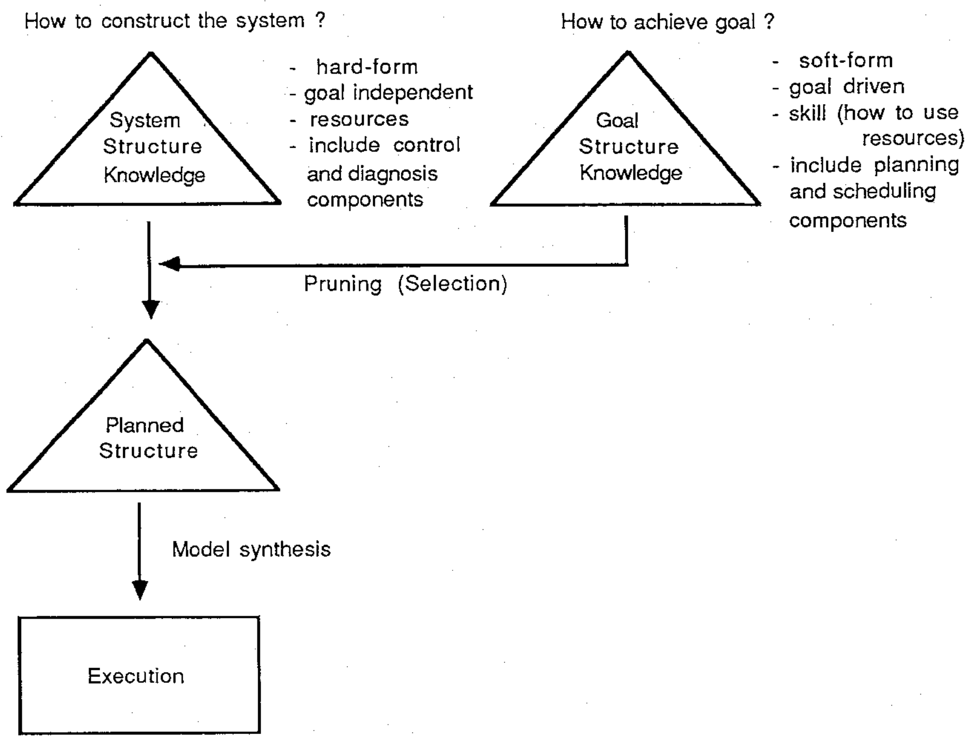

How to construct the system ?
System Structure Knowledge
- hard-form
- goal independent
- resources
- include control and diagnosis components
How to achieve goal ?
Goal Structure Knowledge
- soft-form
- goal driven
- skill (how to use resources)
- include planning and scheduling components
Pruning (Selection)
Planned Structure
Model synthesis
Execution

Figure 2. Planning Concept: Viewing Planning as Pruning the Goal Structure

3.2 Model-based Operation

The event-based control paradigm realizes intelligent control by employing a discrete eventistic form of control logic represented by the DEVS formalism [36,40]. In this control paradigm, the controller expects to receive confirming sensor responses to its control commands within defined time windows determined by its model of the system under control. An essential advantage of the event-based operation is that the error messages it issues can bear important information for diagnostic purposes.

The operational model used by event-based operation has a state transition table that is abstracted from a more detailed model that represents both normal and abnormal behavior. The state transition table keeps a knowledge of states, commands, next states, outputs, and time windows. The window associated with a state is determined by bracketing the time-advance values of all transitions associated with the corresponding states in the more detailed model. This divergence arises due to variations in parameters and initial states considered to represent normal behavior.

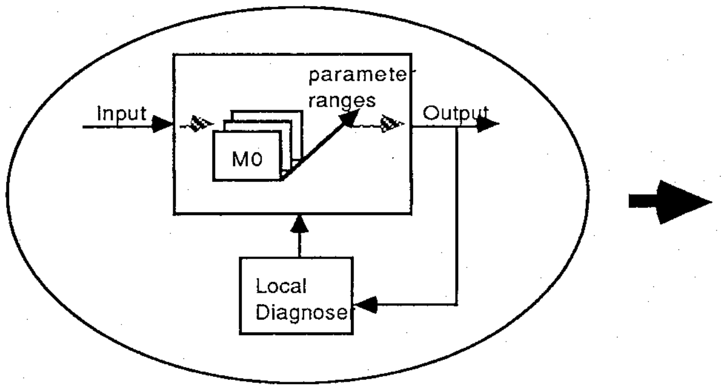
The operator uses the goal-table from the planner and the state-table of its operational model to issue commands to the controlled device. When proper response signals are received the operator causes the model to advance to the next state corresponding to the one in which the device is supposed to be. The operator ceases interacting with the device as soon as any discrepancy, such as a *too-early* or *too-late* sensor response, occurs and calls on an associated fault diagnoser.

3.3 Model-based Diagnosis

A model-based diagnoser assumes that its model of the system is correct, and looks for discrepancies between the behavior of its model and the behavior of the real system. If such a discrepancy has been detected, it assumes that the system has undergone some change which is considered a fault. In our approach, the diagnoser will make use of a fault model to match the observed anomalous behavior of the system. Diagnosis in the model-based architecture is performed by local and global diagnosers, to find single or multiple faults using knowledge of structure and behavior.

By local diagnosis (Figure 3(a)) we mean the diagnostic description of a component model: the local diagnoser looks for a fault that might have occurred within the currently activated model unit. Once the controller has detected a sensor response discrepancy, the diagnoser is activated. Data associated with the discrepancy, such as the state in which it occurred, and its timing, are also passed on to the local diagnoser. From such data, as well as information it can gather from auxiliary sensors, the local diagnoser tries to discover the fault that occurred.

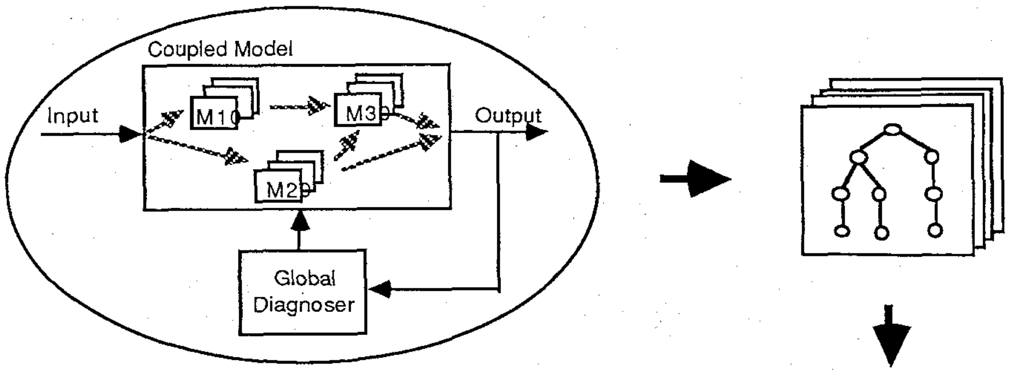
If the local diagnosis fails, the global (higher level) diagnoser is activated to analyze the enclosing coupled model (Figure 3(b)). The global diagnostic model is a cause-effect table obtained by symbolic simulation to generate all fault-injected trajectories that reach states exhibiting the detected symptom. Faults injected in such trajectories represent candidate diagnoses [41].



Output (Effect)	Model (Cause)
O1 O2 O3	M1 M2 M3

(a). Local Diagnosis

Figure 3. Diagnostic Model Generation



Output (effect)	Models (causes)
O1	M11 M32
O2	M22

Where $M_{i,0}$ = normal model and $M_{i,j}$ ($j > 0$) = fault model

(b). Global Diagnosis

Figure 3. Diagnostic Model Generation (continued)

4. Endomorphic System Concept

Endomorphism refers to the existence of a homomorphism from an object to a sub-object within it, the part (sub-object) then being a model of the whole [15]. As illustrated in Figure 4, in order to control an object, a high autonomy system needs a corresponding model of the object to determine the particular action to take. The internal model used by the system and its world base model are related by abstraction, i.e., some form of homomorphic (i.e., endomorphic) relation. The inference engine asks its internal model for the necessary information for interacting with the real world object. By "world base model" we mean the most comprehensive model of the world available to the system whether it exists as a single object or as a family of partial models in the model base.

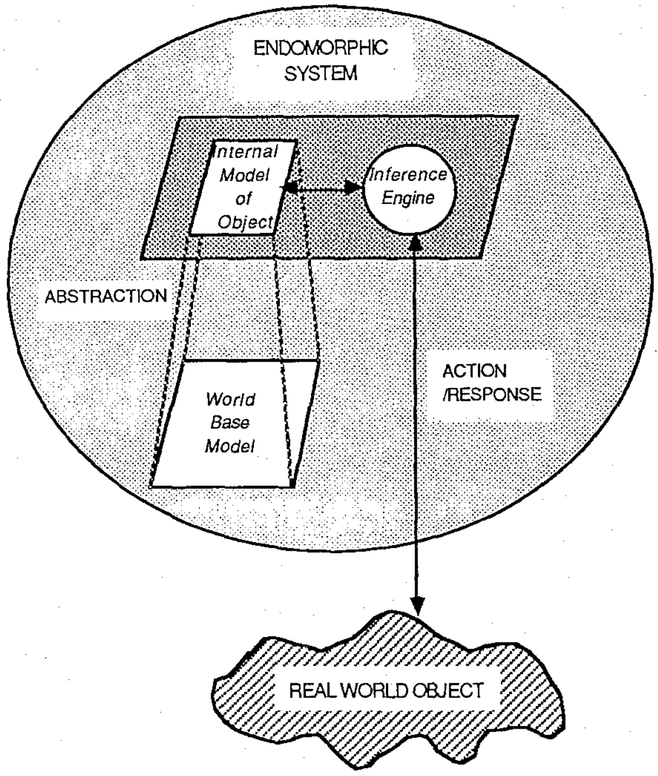


Figure 4. Endomorphic System Concept

5. Engine-based Design Methodologies

Typical expert systems comprise a domain-independent inference engine and a domain-dependent knowledge base. The inference engine examines the knowledge base and decides the order in which inferences are made. The engine-based modelling approach provides a clear separation between the domain-dependent model base and the domain-independent inference engine. It facilitates the automatic generation of a model base using endomorphisms. Figure 5 shows the engine-based modelling concept and examples of autonomous system components realized using the concept.

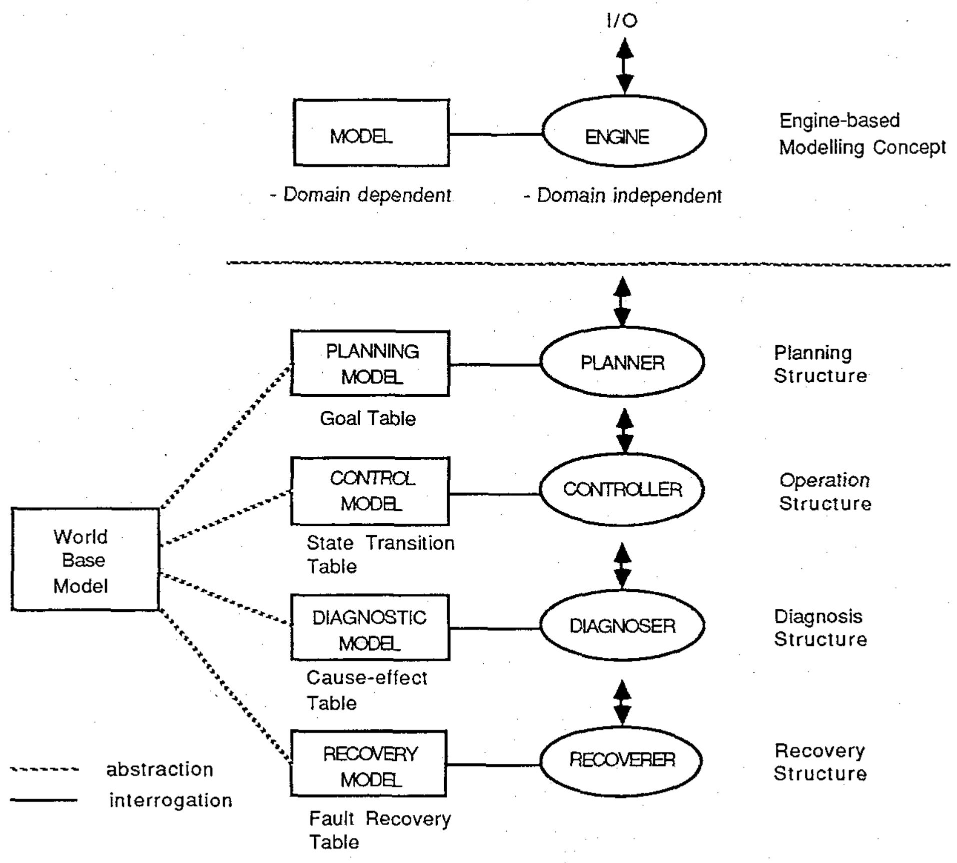


Figure 5. Engine-based Modelling Concept for Endomorphic System Design

6. Hierarchical Development of Intelligent Units

The autonomous system components described above have to be coupled within a unit in order to interact with each other. We use the term "Intelligent Unit" to denote the smallest unit that encapsulates all engine-based components as shown in Figure 6.

To cope with complex problems, an autonomous system requires multiple intelligent units coupled in a hierarchical fashion. Models in the hierarchy must have valid abstraction relations to each other. Figure 6 illustrates an autonomous system development based on hierarchical abstraction and integration. Intelligent units at the leaf nodes of the execution structure employ internal models directly abstracted from the world base model. Units at higher levels employ internal models that are abstractions of coupled models composed of immediately inferior internal models.

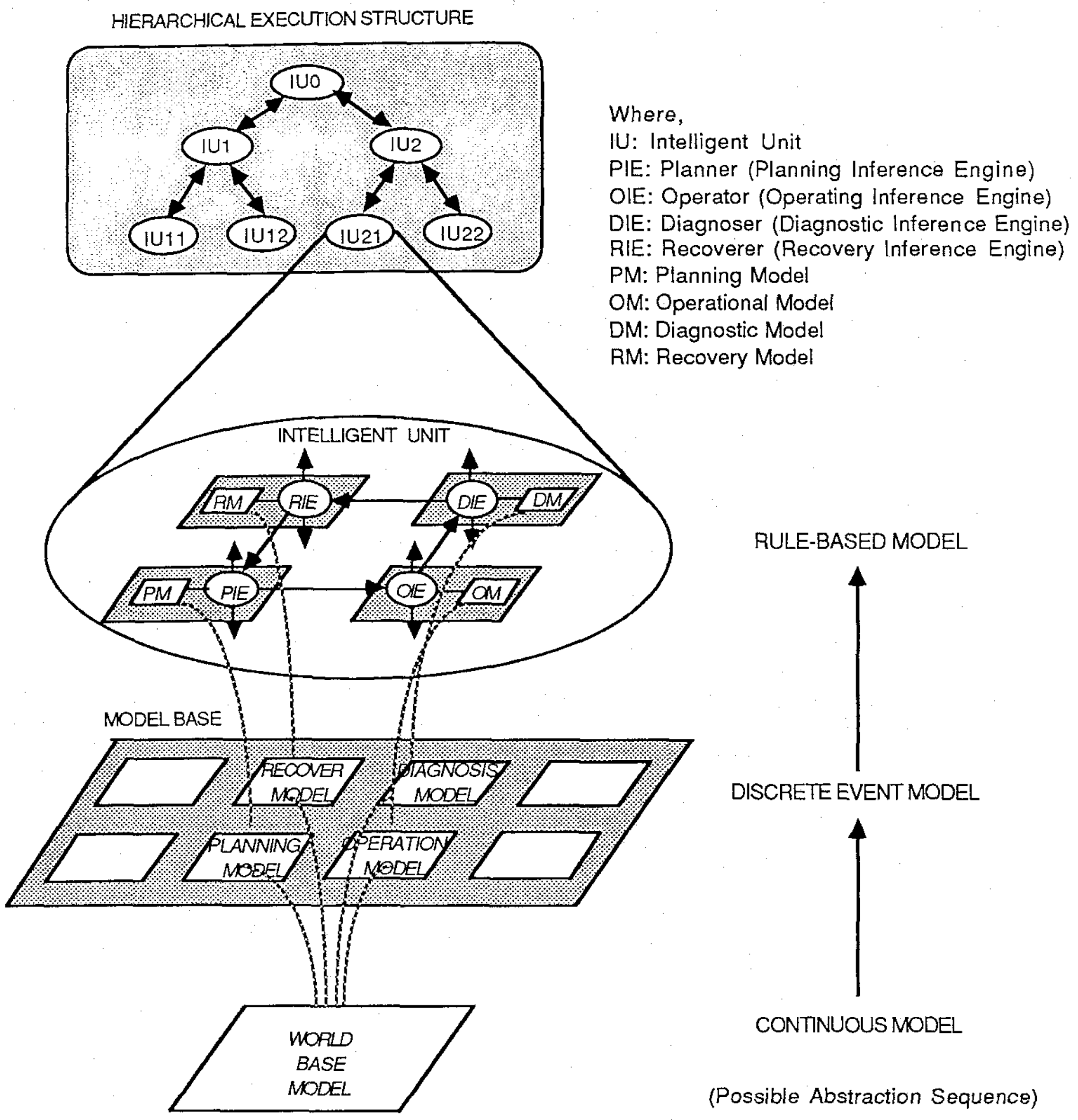
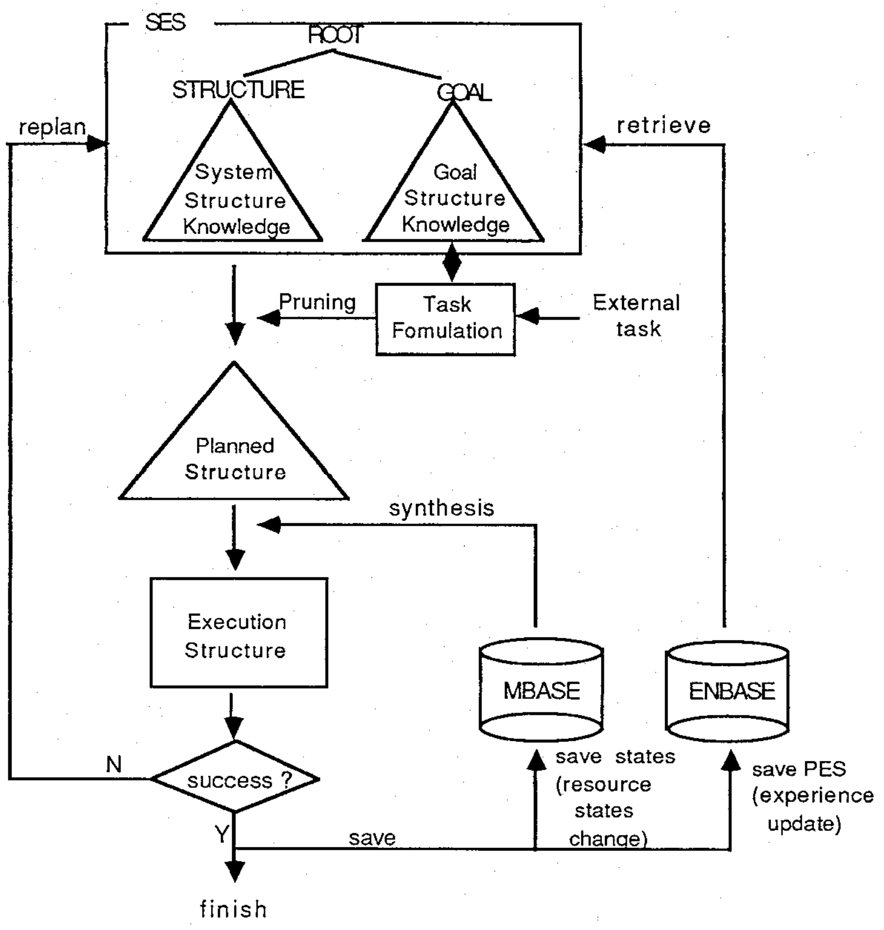


Figure 6. Hierarchical Abstraction and Integration of High Autonomy System

The overall methodology for autonomous system generation in a model-based simulation environment is shown in Figure 7. The task formulation module receives an objective. It then retrieves an SES from the ENBASE and generates a plan structure by using the goal sub-SES of the SES (for initial planning) or partitioned PESs (for experience-based planning). A simulation structure is obtained by synthesizing the models in MBASE, where models can exist in advance or be generated automatically.

The initial environment for generating an autonomous system can be obtained using the layered development concept of DEVS-Scheme [15]. The first layer is the Lisp-based, object-oriented programming system that provides the foundation on which the system is built. The next layer is the SES/MB system where the behavioral and structural models can be built and saved in the MBASE and ENBASE, respectively. Models in MBASE are base models as well as internal models abstracted from base models.



Where,
SES: System Entity Structure
PES: Pruned Entity Structure
MBASE: Model Base
ENBASE: Entity Structure Base

Figure 7. Autonomous System Generation Methodology

Phase I : Plan generation

Once the basic environment is built, the next phase is the planning structure generation. When receiving a goal command, the task hierarchy is generated in a top-down fashion (task decomposition) as shown in Figure 8(a).

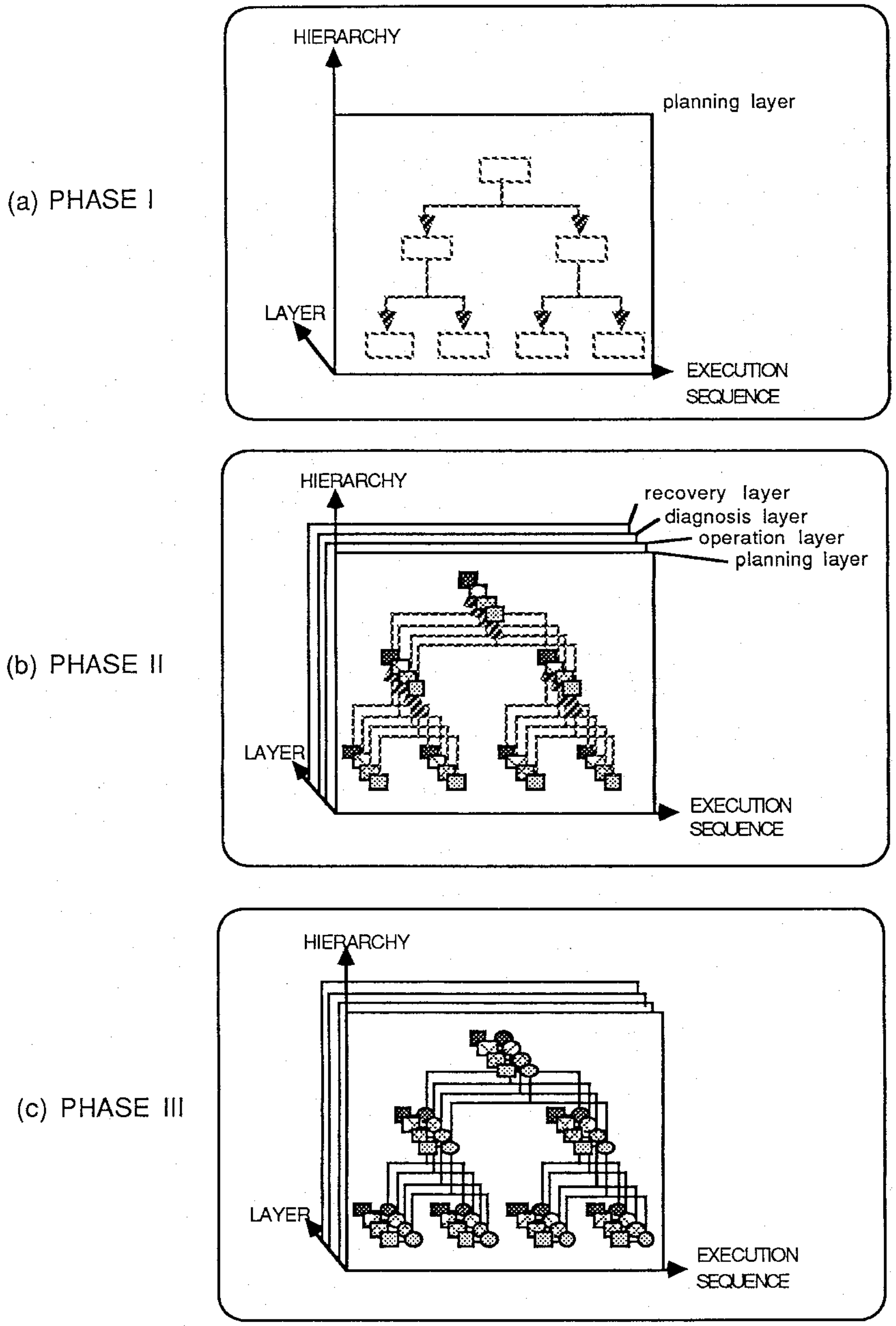


Figure 8. Autonomous System Generation Procedure

Phase II : Model construction

The next phase is the model base construction illustrated in Figure 8(b), where the necessary models can be retrieved from MBASE or automatically generated from the lower level models. This multi-layered hierarchical model generation and abstraction can be done in a bottom-up fashion. The resultant structure represents the domain dependent knowledge base structure.

Phase III : Engine attachment/integration

By attaching domain independent engines such as a planner, an operator, a diagnoser, and a recoverer which are able to interrogate corresponding models, we have a multi-agent structure. Now by coupling those agents, we can obtain the autonomous system architecture shown in Figure 8(c).

7. Example: Robot-managed Space-borne Laboratory

As an example of a model-based architecture of an autonomous system, let us consider a robot-managed space-borne laboratory environment [42]. Figure 9 illustrates the approach taken. The entity structure for the SSL (Space Station Laboratory) decomposes this entity into a structure knowledge part and an experiment (goal) knowledge part; STRUCTURE and EXPERIMENT, respectively. The former is for the execution structure; hardware, software, instrumentation, etc. The latter describes how to achieve given goals (see also Figure 2). Each of the entities will have one or more classes of objects (models) expressed in DEVS-Scheme to realize it.

The EXPERIMENT has a three level hierarchy; high-level models (HM), middle-level models (MM), and low-level models (LM). Each level is designed for experimental knowledge representation, which is used to resolve a given task into necessary component models.

The laboratory STRUCTURE is decomposed into SPACE and OBJECTS. For a full explanation see reference [39]. Each OBJECT is specialized into ROBOT and EQUIP. Each ROBOT is decomposed into MOTION, SENSE, and BRAIN. The EQUIP is a generic entity for the laboratory equipment which is modeled in much the same way as the ROBOT. However, EQUIP has no BRAIN and its MOTION and SENSE subsystems are always passive.

Each Robot's Cognition system (BRAIN) is decomposed into a SELECTOR used as the controller and MPUs (Model Plan Units) used as components. The MPU is specialized into HMPU, MMPU, and LMPU corresponding to the EXPERIMENT abstraction hierarchy mentioned earlier.

There are three levels in the model plan unit: high-level model plan units(HMPU), middle-level model plan units(MMPU), and lowest-level model plan units(LMPU). The HMPU first formulates a given task (command from the earth base) and then divides it into lower level units, MMPUs. Here actions are sequenced (planned) by chaining necessary actions in forward/backward manner starting from the given target action which is assigned from the task formulation process. The MMPUs in the hierarchy are divided again into LMPUs. To control its lower level units, the MMPU has abstracted states and

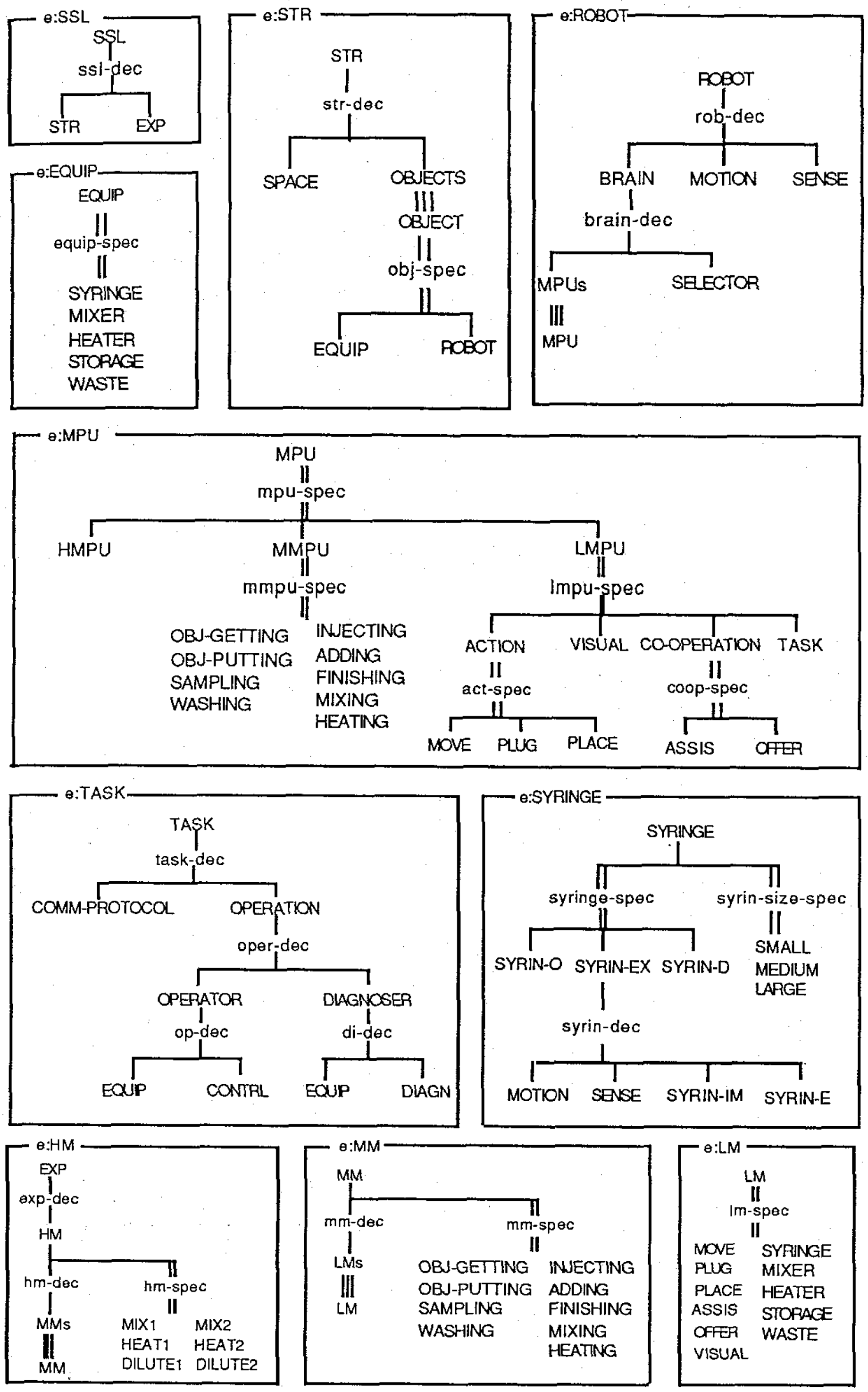


Figure 9. Partitioned SES of Robot-Managed Laboratory

time windows of its lower level LMPUs. The LMPU is the lowest level in the hierarchy which employs an event-based control logic for operation and fault diagnosis. The HMPU manages such activation sequences (MMPUs). Each action unit (MMPU) is divided into smallest action units (LMPUs).

At each level, the intelligent unit has its own internal models and corresponding engines to supervise its sub-component units. There are two types of messages in the hierarchy, goal (command) and done (response) messages. The goal is divided into a set of subgoals in a top-down manner. The done messages are gathered in a bottom-up manner. There are three types (+,0,-) of done messages: + is for a *success*, - for a *failure*, and 0 for an *unknown*. The unknown message may result due to the lack of available sensors or a complex fault associated with other units in the hierarchy. Each of the refined units is developed on the basis of the event-based control logic. Thus each lowest level unit has a pair of controller and local diagnoser model/engine to handle the normal and abnormal condition of its objective model, respectively.

As a concrete illustration, let us set up a mixing experiment (i.e., mix x amount of liquid-A with y amount of liquid-B) in the space-borne laboratory environment. To perform such a task, a robot must identify a syringe required to sample a liquid-A, then bring that syringe to the identified storage in which liquid-A is stored, and perform the sampling from the storage, and so on.

By either pruning the SES (no prior experience) employing the rule-based approach or retrieving existing PESs (experienced) in the EXPERIMENT part of the SES (see Figure 9), we can obtain a simulatable execution structure as depicted in Figure 10, where each MPU block shows its model name and its goal, respectively.

The execution is achieved by employing event-based control logic where it compares the expected time from the operational model with the observed time from the sensory response of the real system to be controlled. In this mixing example, after several successful executions, i.e. MOVE, PLACE, MOVE, and PLUG, suppose the actual filling time during the fill-it operation in SYRINGE LMPU is 12 seconds, but the operational model indicates 8 to 10 (minimum time is 8 seconds and window time is 2 seconds) as its normal timing, then we are in a fault situation and the detected fault timing is *too-late*.

Once the controller detects the fault, then the local diagnoser is activated, where it can use two types of sensory information: indicator (expected sensor, timing, phase) and vision (liquid-level, needle-angle, tube-constriction). If the sensory feedback information indicates: sensor =*full-sensor*, timing = *too-late*, phase =*filling*, angle = *normal* (i.e. 89 < angle < 91), constriction = *normal* (i.e. constriction < 10), and level < 100 (where 100 is the full level value for the small size syringe), then the local diagnosis in the SYRINGE unit would be "*Syringe is leaky*".

Note that our desired sub-plan of the SAMPLING unit is to sample a small amount of liquid-A from the storage-A. However, suppose a robot picks up a large size syringe instead of a small size syringe for some reason such as "arm motor abnormal" or "mobile motor abnormal" or "by obstacle", etc. If such an abnormal condition was not out of the detectable range within its action (model) unit, then such an abnormality would be propagated until the filling action where timing should be *too-late* because the actual syringe is the large size contrary to the intention. In this case, the sensory information would look as follows: sensor =*full-sensor*, timing = *too-late*, phase =*filling*, angle =

normal, constriction = *normal*, and level ≥ 100. Therefore the local diagnoser for the SYRINGE unit would not be able to conclude a fault. It would activate the global diagnoser in the SAMPLING unit. The global diagnoser uses the global diagnostic table that is compiled from the symbolic simulation of every possible fault case. It would conclude that the detected fault is caused by 1) position is changed during moving action to syringe desk so that robot position is in bad angle to pickup the large syringe during picking-up action or 2) position is changed during moving action to the liquid-A storage so that robot position is in bad angle to plug it in during plugging action, and 3) combination of both cases. Among those possibilities, suppose case 1 has a higher priority (the priority would depend on the system requirements, objectives, etc.) than the other possibilities, then the recoverer would set up the new goal, for example, "remove obstacle, or check power line, and then continue remaining plan by picking the small syringe up.

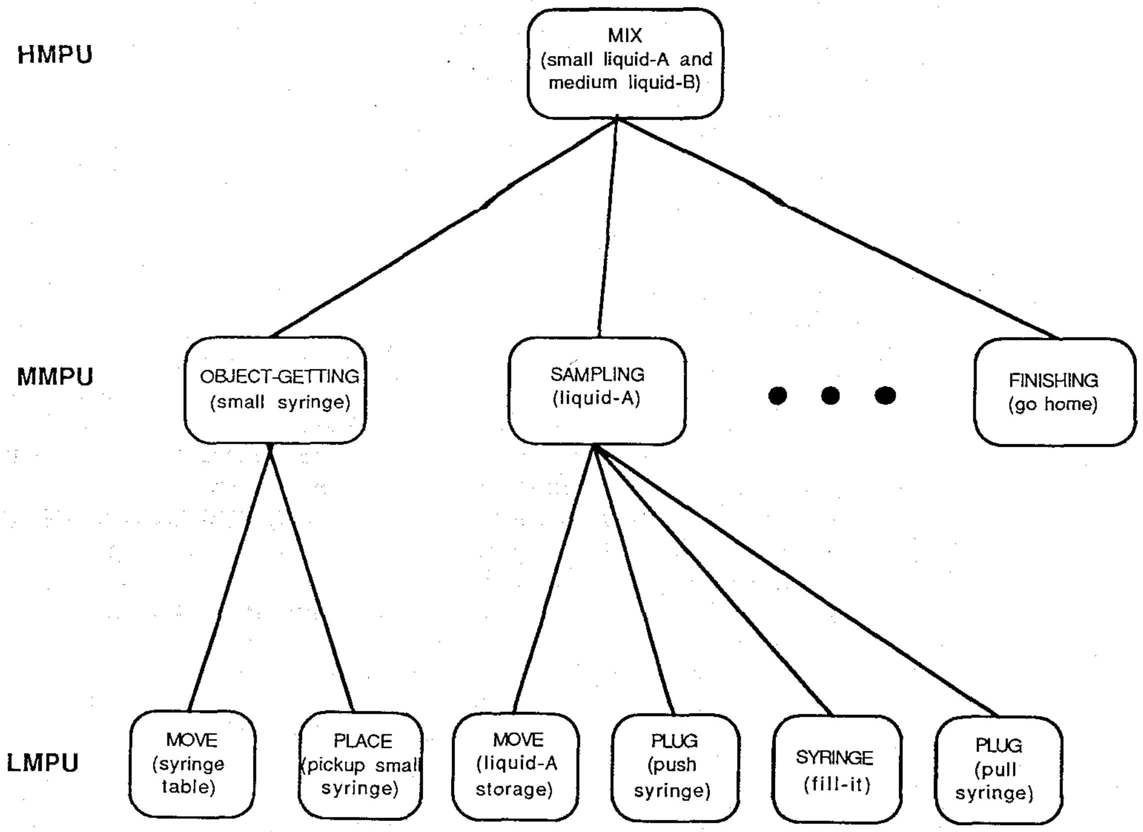


Figure 10. Mixing Example of Hierarchical Execution Structure

8. Conclusions

The main characteristics of the proposed architecture are as follows:

1. The time-based formalism (DEVS) provides coherent integration between symbolic and numeric models.

2. The SES/MB environment provides a hierarchical modularity, reusability, and testability.
3. Model-based deep reasoning supports a powerful diagnostic capability.
4. The endomorphism concept supports intelligent unit design and a consistent model base.
5. The engine-based design builds a domain-independent architecture instantiated with compiled models for application under real time constraints.
6. The intelligent unit encapsulates various autonomous components such as planning, operation, diagnosis, and recovery coherently to cope with complex problems.
7. Model-based planning supports the reusability of experienced plan structures.
8. The event-based control logic guarantees robustness, reduced sensor complexity, and increased diagnostic capability.

The framework has been implemented in the space-borne laboratory system as a proof of the concept.

References

[1] A. Meystel, "Intelligent Control: A Sketch of the Theory," *J. Intelligent and Robotic Systems*, vol.2, No.2&3, pp.97-107, 1989.

[2] P.J. Antsaklis, K.M. Passino and S.J. Wang, "Towards Intelligent Autonomous control Systems: Architecture and Fundamental Issues", *J. Intelligent and robotics systems,* Vol.1, No.4, pp. 315-342.1989.

[3] A. Newell, "Putting it All together," In: *Complex Information Processing: The Impact of Herbert A. Simon* (eds: D. Klahr, K. Kotovosky), Lawerence Erlbaum, Hillsdale, NJ, 1988.

[4] M.A. Fischler and O. Firshein, *Intelligence, The Eye, The Brain, and The Computer,* Addison-Wesley Pub. Co., Reading, MA, 1987.

[5] T. Kanade, A roundtable discussion: Present and Future directions: trends in Artificial Intelligence, OE Reports, SPIE, September, 1989.

[6] B. Chandrasekaran, A roundtable discussion: Present and Future directions: trends in Artificial Intelligence, OE Reports, SPIE, September, 1989.

[7] NASA, *The Space Station Program*, NASA Pub., 1985.

[8] G.N. Saridis, "Intelligent Robotic controls", *IEEE Trans. on Auto. control.*, AC-28, No.5,1983.

[9] J.S. Albus, "The Role of World Modeling and Value Judgment in Perception", Proc. IEEE Conf. on Intelligent Control, September, Philadelphia, PA, 1990.

[10] J.S. Albus, "A Theory of Intelligent Systems", Proc. IEEE Conf. on Intelligent Control, September, Philadelphia, PA, 1990.

[11] J.S. Albus, "Hierarchical Interaction Between Sensory Processing and World Modeling in Intelligent Systems", Proc. IEEE Conf. on Intelligent Control, September, Philadelphia, PA, 1990.

[12] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", J. of Robotics and Automation, RA-2, pp. 14--23, 1986.

[13] B.P. Zeigler, *Theory of Modelling and Simulation*, New York, NY : Wiley, 1976 (reissued by Krieger Pub. Co., Malabar, FL,1985).

[14] B.P. Zeigler, *Multifaceted Modelling and discrete Event simulation,* Academic press, 1984.

[15] B.P. Zeigler, *Object-Oriented simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic systems,* Academic Press, 1990.

[16] B.P. Zeigler, C.J. Luh, and T.G. Kim, "Model Base Management for Multifacetted Systems", Proc. AI, Simulation and Planning in High Autonomy Systems, pp. 25-35, IEEE Press, Tucson, March, 1990.

[17] Ho, Y. "Editors Introduction", *Special Issue on Dynamics of Discrete Event Systems, Proceeedings of IEEE*, Vol. 77, No. 1, 1989.

[18] Garzia, R.F., M.R. Garzia, and B.P. Zeigler, "Discrete Event Simulation", *IEEE Spectrum*, December, pp. 32--36, 1986.

[19] Nance, R.E., "A Conical Methodology: A Framework for Simulation Model Development", *Proc. Conf. on Methodology and Validation*, SCS Pubs. San Diego, pp. 38--43, 1987.

[20] Praehofer, H., "System Theoretic Formalisms for Combined Discrete-Continuous System Simulation," Special Issue on Modelling and Simulation for High Autonomy Systems, *Int. J. Gen. Sys.* (to appear).

[21] Q. Wang and F.E. Cellier, "Time Windows : An approach to Automated Abstraction of Continuous-Time Models into Discrete-Event Models," *Proc. on AI, Simulation, and Planning in High Autonomy Systems,* Tucson, 1990.

[22] Narain, S. "An Approach to Reasoning about Hybrid Systems", Special Issue on Modelling and Simulation for High Autonomy Systems, *Int. J. Gen. Sys.* (to appear).

[23] Allen, J.F., "Toward a General Theory of Action and Time," *Artificial Intelligence*, Vol. 23, pp. 123--134, 1984.

[24] S.A. Vere, "Planning in Time : Windows and Durations for Activities and Goals", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 3, pp. 246-267, May, 1983.

[25] Zhong, H. and M.J. Wonham, "Hierarchical Coordination," *Proc. 5th EEE Symp. on Intelligent Control*, Philadelphia, PA, pp. 8--14, 1990.

[26] Kuipers, B.J., "Qualitative Simulation", *Artificial Intelligence*, pp. 289--338, 1986.

[27] Kuipers, B.J., "Qualitative Reasoning with Causal Models in Diagnosis of Complex Systems", In: *Artificial Intelligence, Simulation and Modelling* (eds. L.A. Widman, K.A. Loparo, and N. Nielsen), J. Wiley, NY, pp. 257--274, 1989.

[28] Y. Dekleer, "Qualitative Physics, A Personal View", In: *Readings in Qualitative Physics* (eds. D. Weld and Y. dekleer), Morgan Kaufman, Palo Alto, 1989.

[29] Iwasaki, Y., "An Integrated Scheme for Using First-Principle Physical Knowledge for Knowledge-based Simulation", *Proceedings of Fourth AAAI Workshop on AI and Simulation*, pp. 57--59, 1989.

[30] R. Rajagopalan, "The role of Qualitative Reasoning in Simulation", In: *Artificial Intelligence in Simulation* (eds. G.C. Vansteenkiste, E.J.H. Kerckhoffs, & B.P. Zeigler), SCS Pub., San Diego, CA, 1986.

[31] M.R. Gensereth, "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence*, Vol. 23, pp. 411-436, 1984.

[32] Y. De Kleer and B.C. Williams, "Diagnosing multiple faults", *Artificial Intelligence*, Vol. 32, pp. 97 - 130, 1987.

[33] R. Reiter, "Theory of diagnosis from first principles", *Artificial Intelligence*, Vol. 32, pp. 57 - 95, 1987.

[34] Fishwick, P.A., "Fuzzy Simulation: Specifying and Identifying Qualitative Models", Special Issue on Modelling and Simulation for High Autonomy Systems, *Int. J. Gen. Sys.* (to appear).

[35] G. Chien, "Dynamic System Modeling and Simulation in Product Design", Master Thesis, Illinois Inst. of Tech., 1989.

[36] B.P. Zeigler, "DEVS Representation of Dynamical Systems:Event-Based Intelligent Control", IEEE proc. Vol.77, no.1, Jan.1989. pp.72-80.

[37] K.J. Hammond, *Case-Based Planning*, Academic Press, 1989.

[38] N.J. Nilsson, *Principles of Artificial Intelligence*, Tioga Pub. Co., Palo Alto, CA, 1981.

[39] S. D. Chi, B. P. Zeigler, and F. E. Cellier, "Model-based Task Planning System for a Space Laboratory Environment" *SPIE Conference on Cooperative Intelligent Robotics in Space*, Boston, Nov., 1990

[40] S.D. Chi and B.P. Zeigler, "DEVS-based intelligent Control of Space Adapted Fluid Mixing", Proceeding of 5th conf. on Artificial Intelligence for Space Applications, May, 1990.

[41] B.P. Zeigler and S.D. Chi, "Symbolic Discrete Event System Specification", *Conf. on AI, Simulation and Planning in High Autonomy Systems*, Florida, April, 1991 (revised version will be submitted to IEEE Expert System).

[42] B.P. Zeigler, F.E. Cellier, and J.W. Rozenblit, "Design of a Simulation Environment for Laboratory Management by Robot Organizations", *J. Intelligent and Robotic Systems*, Vol.1, pp. 299-309, 1988.