Model-Based Task Planning System for a Space Laboratory Environment

Sung-Do Chi, Bernard P. Zeigler, and Francois E. Cellier

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

ABSTRACT

This paper describes the design of a model-based autonomous planning system that will enable robots to manage a space-borne chemical laboratory. In a model-based planning system, knowledge is encapsulated in the form of models at the various layers to support the predefined system objectives. Thus the model-based approach can be considered as an extended planning paradigm which is able to base its planning, control, diagnosis, repair, and other activities on a variety of objectives-related models. We employ a *System Entity Structure/Model Base* framework to support autonomous system design through the ability to generate a family of planning alternatives as well as to build hierarchical event-based control structures. The model base is a multi-level, multi-abstraction, and multi-formalism system organized through the use of system morphisms to integrate related models.

1. INTRODUCTION

Planning is a subfield of artificial intelligence that has yet to find a coherent theory. Recently, it has been extended a broader range of problems and techniques than the classical planning problem. Planning can be defined as the design process for selecting and sequencing actions to transform the world from a given state to a desired state. Conflicts between subgoals must be resolved so that by the time one comes to execute an action in the plan, one can be sure that this action will be executable[1,2,3]. Planning is, also, viewed as a reasoning about how actions affect the world[4]. In case-based planning, a system learns how to plan by synthesizing new plans from existing ones and analyzing how they interact with the world[5,6].

Faced with the overwhelming complexity of planning, a model-based approach strives to integrate key ideas in existing planning paradigms; hierarchical abstraction, time constraints, and dynamic memory retrieval[1,7,8]. In the model-based approach, knowledge is encapsulated in the form of models that are employed at the various hierarchical abstraction layers to support predefined system objectives. Thus the model-based approach can be considered as an extended planning paradigm which is able to base its planning, control, diagnosis, repair, and other activities on a variety of objectives-related models[9,10,11].

The work described in this paper derives from research related to telerobotics. The project is to develop a technology that will allow carefully designed and specially constructed laboratory robots to perform many routine tasks in a Space laboratory under remote supervision from the ground. Therefore the robots must be able to perform simple operations autonomously, and communicate with the ground only at the task level and above. Such robots should be able to judge the adequacy of a proposed action plan on the basis of expectations of its effects on the laboratory, materials, instruments, etc. For this purpose, it is important that models at various levels of granularity can be automatically generated from a set of generic master models[12].

transformation of the entity subtree is aborted. *Retrieve* looks for a model first in working memory. If no model is found in working memory, the *retrieve* procedure searches through model definition files, and finally, provided that the entity is a leaf, in pruned entity structure files. A new incarnation of the *transform* process is spawned to construct the leaf model in the last case. Once this construction is complete, the main *transform* process is resumed.

The result of a transformation is a model expressed in an underlying simulation language such as DEVS-Scheme[14] which is ready to be simulated and evaluated relative to the modeler's objective. The fact that the *transform* process can look for previously developed pruned entity structures, in addition to basic model files, encourages PES reusability.

In model-based task planning and execution, knowledge is encapsulated in the form of models that are employed at the various control layers to support the predefined system objectives. Lower 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. Discrete event representation, facilitating event-based control, can be employed to map traditional dynamic to symbolic models[17,20].

Note than an autonomous system could in principle, base its operation, diagnosis, repair, planning, and other activities on a single comprehensive model of its environment. However, such a model would be extremely unwieldy to develop and lead to intractable computations in practice[12]. Instead, our architecture employs a multiplicity of partial models to support system objectives. As indicated, such models differ in level of abstraction and in formalism. The partial models, being oriented to specific objectives, should be easier to develop and computationally tractable[15]. However, this approach leads to sets of overlapping and redundant representations. Concepts and tools are need to organize such representation into a coherent whole. Morphisms[15,17,21] can connect models at different levels of abstraction so that they can consistently modified.

Fishwick[22] has extended process abstraction concepts and implemented a simulation system that is able to switch between levels within simulation runs. However, although the need for multiple levels of abstraction has been recognized in mainstream AI, there has been little consideration of the importance of the morphism concept to this issue. But just as tools are needed to enable agents to plan, diagnose, and reason effectively with respect to particular objects, so tools are needed to organize the various models that support such planning, diagnosis, and reasoning. 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. Such a model base is a special case of multifacetted model base management[15].

The SES/MB framework provides an ability to develop model-based planning systems. It can supports:

(1) multiplicity of partial models to support system objectives (multifacetted modelling).
(2) integration of dynamic and symbolic models at different layers (hierarchical architecture).
(3) multi-abstraction to integrate related models (system morphism).
(4) execution, control, diagnosis, and repair (event-based control).

(5) selection/retrieval of initial/modified planning models (pruning and reusability).

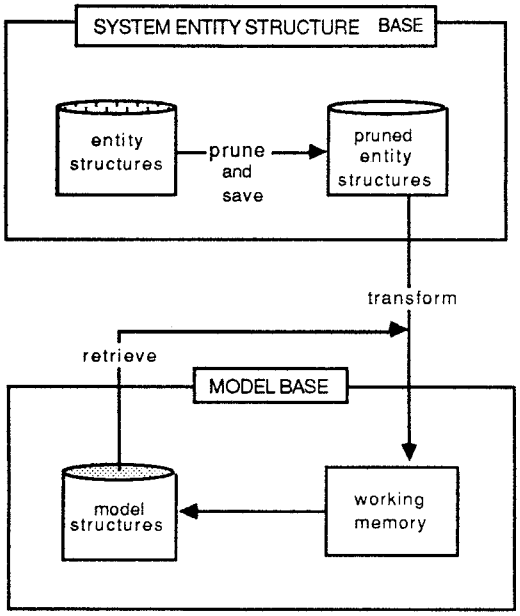

Fig. 1. The System Entity Structure/Model Base (SES/MB) Environment

3. SPACE-BORNE LABORATORY

As an example of a model-based architecture of autonomous planning system, let us consider a robot-managed space-borne laboratory environment. Such a laboratory requires design of advanced robot-controlled instrumentation. For example, handling fluids in orbit will be essential to many of the experiments being planned in manufacturing and biotechnology. However, the micro gravity conditions of space necessitate radically different approaches to fluid handling than common on earth. As experience in space accumulates, approaches and instrumentation will likely undergo continual modification, enhancement, and replacement. Thus robots for managing such equipment must be highly autonomous and flexible so that constantly changing environments can be accommodated[23].

In designing the robot models, we assume that necessary mobility, manipulative and sensory capabilities exist so that we can focus on task-related cognitive requirements. Such capacities, the focus of much current robot research, are treated at a high level of abstraction obviating the need to solve current technological problems.

The laboratory environment is constructed on the basis of object-oriented and hierarchical models of laboratory components within DEVS-Scheme. Laboratory configurations are generated by pruning the entity structure knowledge representation. The laboratory is designed to be as generic as possible by employing hierarchical event-based control logic, which will be discussed later. However, as stated, the focus will be upon fluid handling in microgravity which presents a variety of problems that are unique to space.

Fig. 2 illustrates the approach taken. The entity structure for SSL (Space Station Laboratory) decomposes this entity into structure knowledge part and experiment (goal)

knowledge part; STRUCTURE and EXPERIMENT. The former is for the execution structure; hardware, software, instrumentation, etc. The latter concerns how to achieve given goals. Each of the entities will have one or more classes of objects (models) expressed in DEVS-Scheme to realize it.

The EXPERIMENT has 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 The laboratory STRUCTURE is decomposed into SPACE and OBJECTS. For a full explanation see reference [17].

Each OBJECT is specialized into ROBOT and EQUIP. And each ROBOT is decomposed into MOTION, SENSE, and BRAIN. The EQUIP is a generic entity for the laboratory equipment which is modeled much the same as the ROBOT. However, EQUIP has no BRAIN and its MOTION and SENSE subsystems are always passive. Note that OBJECTS are also defined as a *multiple entity* . Any number of instances may be generated for such an entity, and with the pruning discussed earlier, we can have any desired number of ROBOTs and EQUIPs in the laboratory.

Each Robot's Cognition system (BRAIN) is decomposed into a SELECTOR as controller and MPUs (Model Plan Units) as components. The MPU is specialized into HMPU, MMPU, and LMPU corresponding to the EXPERIMENT abstraction hierarchy just mentioned. HMPU is a high level MPU which manages action planning of MMPUs. Each of which perform the same function with respect to the lower model LMPUs. The latter employ event-based control logic to interact with the environment. These three types of MPU are placed at the same level in our entity structure but conceptually they reflect the hierarchical decomposition structure of the EXPERIMENT models.

As illustrated in Fig. 2, more than one SES may exist in the entity structure base, each one representing a family of models for its root entity. In principle, every entity might have its own SES but this would lead to extreme fragmentation of the encoded knowledge. PESs are saved as they are developed for each of the SESs and available for convenient reuse as illustrated in Fig. 3. See reference [17] for more discussion.

4. MODEL ABSTRACTION HIERARCHIES AND HIERARCHICAL PLANNING/EXECUTION STRUCTURE

This section briefly reviews our approach to model abstraction hierarchies, hierarchical planning structures, and hierarchical event-based control.

Models of intelligent agents must represent not only a decision making component, but also the model of the real system the decision maker uses to arrive at its decisions. Such modelling is based on homomorphic preservation of the input-output behavior where inputs are operation commands to the system and outputs are responses of finite-state sensors attached to the system to observe its state. Selection of controls and sensors must reflect the operation objectives. An atomic DEVS model abstracts incremental micro-state transitions from the continuous model and replaces them by time windows taken for macro-state transitions (which correspond to crossing of sensor thresholds). A coupled DEVS model abstracts the behavior of the composition of lower level DEVS models.

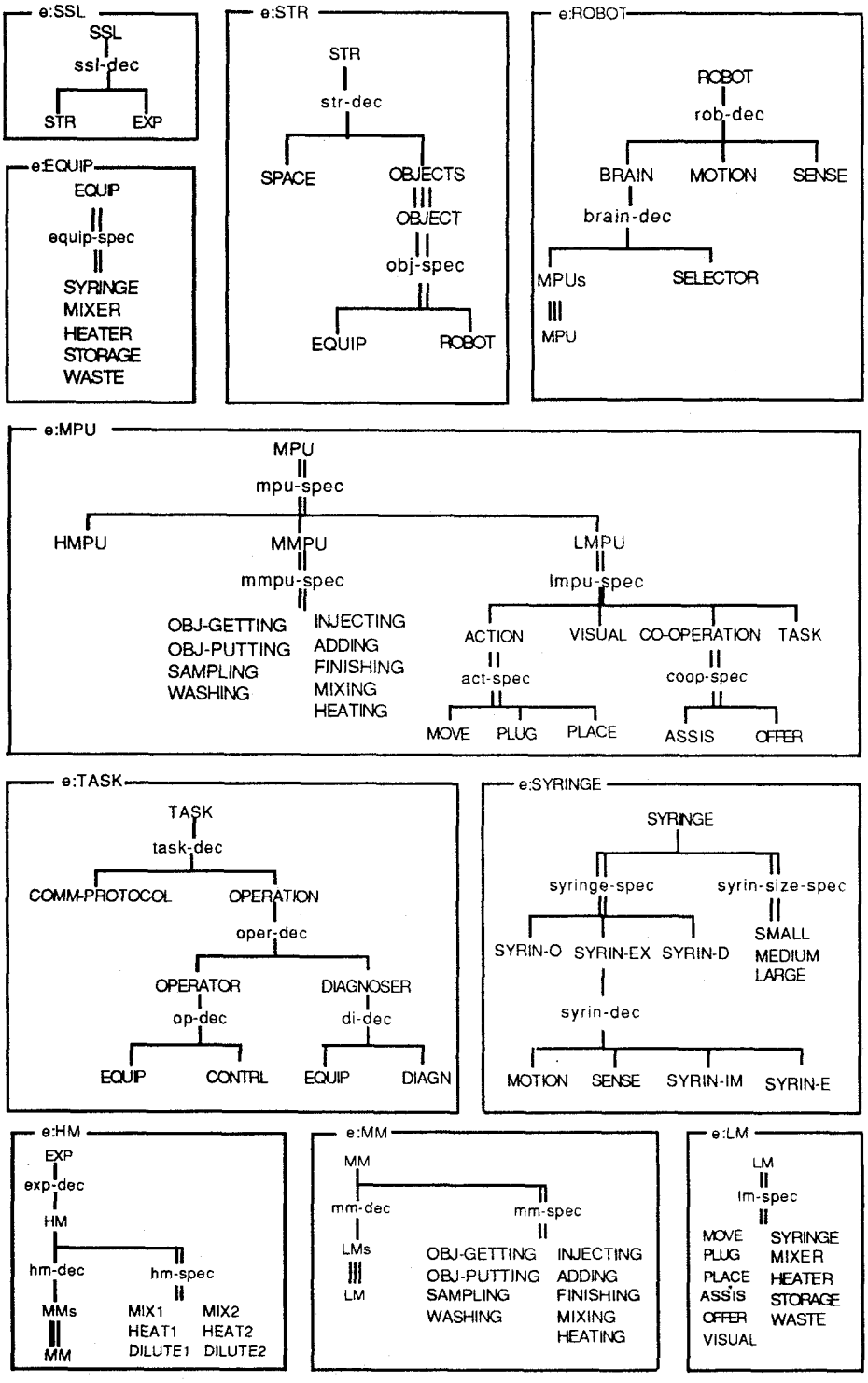

Fig. 2. Partitioned SES of Robot-Managed Chemical Laboratory

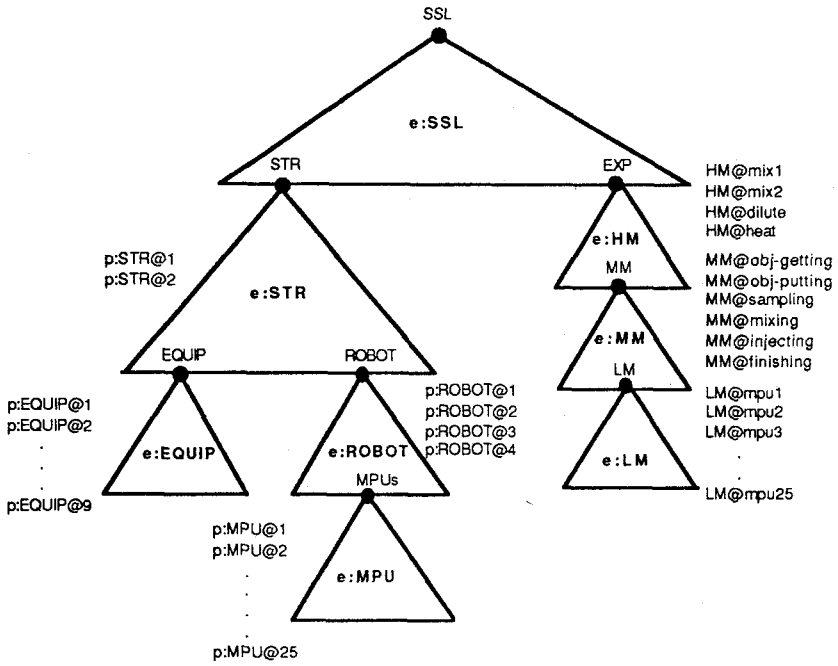

Fig. 3. Hierarchical Structure of Partitioned/indexed PES family for reuse

Fig. 4 represents abstraction related models. At the bottom, MB is a continuous state model of the system being controlled (the most refined model considered for it). Other models are related by abstraction, i.e., a form of homomorphic relation. ME is a discrete event model derived from MB, and MI-O and MI-D are two different abstraction models of MI (for operation and diagnosis, respectively) which in turn is an abstraction of ME. Each abstraction is governed by an underlying morphism. ME serves as the external model of each device, whereas MI-O and MI-D serve as the internal models used by the low level event-based control units (LMPU).

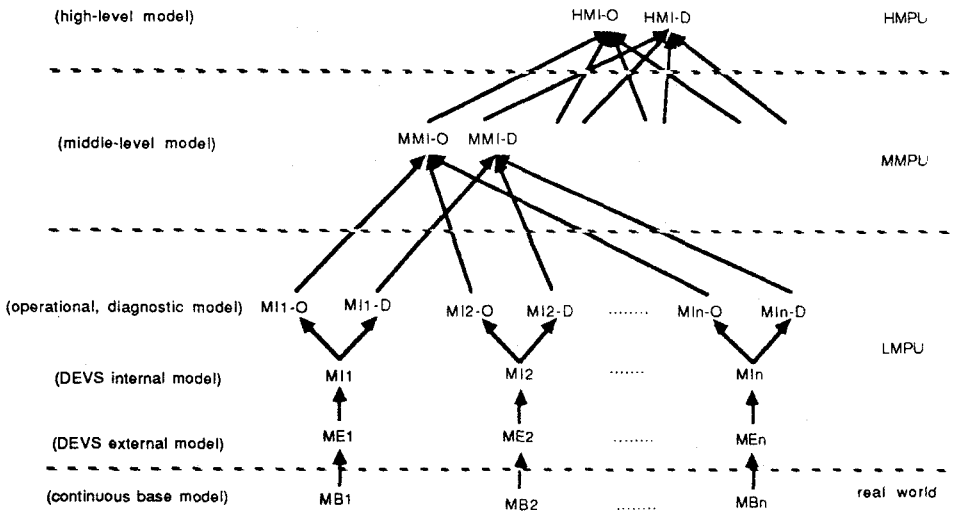

Fig. 4. Abstraction Related Models

Sets of MI-O and MI-D can be composed into higher level models (MMI-O and MMI-D) and abstracted to represent more global state transitions (used by MMPU). Likewise, HMI-O and HMI-D are the highest level models which represent global state transitions (used by HMPU). In this way, the higher level execution units use their own models, which are abstractions of compositions of lower level models, to control their subordinates. The abstraction process is done by aggregating states and windows of lower level models.

The levels of abstraction just illustrated can be formalized using system morphism concepts[15,17]. By generating a hierarchical structure with models of different levels of abstraction we can systematically integrate the highest level goal command with most refined dynamic models.

There are three levels of model plan units: high-level model plan unit(HMPU), middle-level model plan units(MMPU), and lowest-level model plan unit(LMPU). The task formulation planner interprets an external command and uses it to decide the necessary models. Consider external commands, for example, (mix ((liquid-A 100) (liquid-B 100))) and (mix ((liquid-A 100) (liquid-B 200))). Also suppose that liquid transporting syringes are of three different sizes as followings: small (≤100), medium (≤ 200), and large (> 200). In the former case, we need only one small size syringe to handle the liquid. But in the latter case, we need two different syringes, small and medium sizes. And, in both cases, two types of liquids, liquid-A and liquid-B, are involved.

Most laboratory procedures managed by a robot consist of sequences of common steps[24]. These building blocks are represented by the middle level models (MMs) and their associated execution units (MMPUs). The action flow graph (Fig. 5) contains the sequencing constraints imposed on these building blocks. The MM sequencing depends on the target MM as well as needed instruments, e.g. number and size of syringes. Once a target MM and necessary instruments are selected, the sequencing can be done by chaining MMs in backward and forward manner. Conflict resolution is done within the task formulation module. For example, if the target MM is *mixing*, and one syringe is selected, the resulting sequence is : *object-getting*, *sampling*, *fluid-injecting*, *washing*, *sampling*, *fluid-adding*, *mixing*, *washing*, *object-putting*, and *finishing*. On the other hand, if two syringes are selected, then the resulting sequence is : *object-getting*, *sampling*, *fluid-injecting*, *washing*, *object-putting*, *object-getting*, *sampling*, *fluid-adding*, *mixing*, *washing*, *object-putting*, and *finishing*. As already indicated, each MM in the hierarchy is decomposed into LMs.

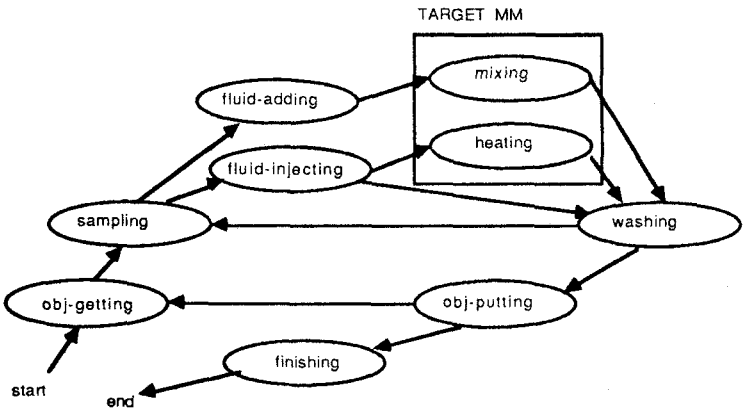

Fig. 5. Action Flow Graph

Once a sequence of MMs is derived to perform a task, it is assigned to a HMPU and an execution control structure is generated as shown in Fig. 6.

The LMPU is the lowest level in the hierarchy and employs event-based control logic for operation and fault diagnosis. In this control paradigm, the controller expects to receive confirming sensor responses to its control commands within definite time windows determined by its model of the system under control. This lowest level unit is consisted of a planner, controller, and diagnoser. The planner works by developing paths backward from the goal in the MM-O model until the given initial states (possible starting states of the controlled system) are reached. See references [15, 16, 20] for further details.

At the medium and high levels, each control unit has its own internal model and controller to supervise its subordinates. 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 top-down manner. Whereas, the done messages are gathered in bottom-up manner. There are three types (+,0,-) of done messages: + is for a *success*, - for a *failure*, and 0 for an *uncertainty* (not sure whether goal was reached). The last message may be due to the lack of relevant sensors or complex fault associated with other units in the hierarchy.

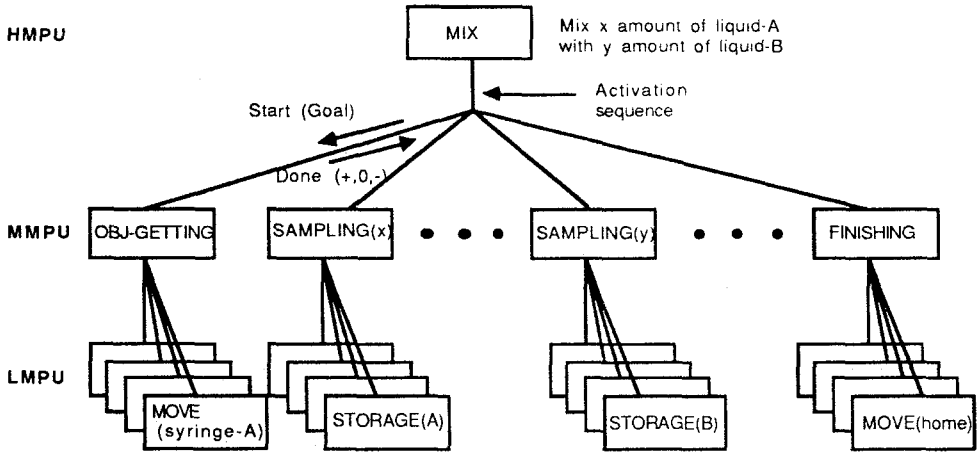

Fig. 6. Mixing Example of Hierarchical Event-based Control Structure

5. AUTONOMOUS SYSTEM GENERATION

The SES/MB framework is used to generate a control structure with a given goal. In other words, once we have specified the available resources (structures) and their behavior characteristics (models), we have to specify how to allocate resources to accomplish a goal. In our SES/MB framework, the planning is viewed as a pruning operation which generates a candidate structure. As shown in Fig. 4, the STRUCTURE entity generates possible configurations of an execution structure to be pruned. On the other hand, the EXPERIMENT entity represents the possible alternative goals. Fig. 7 depicts a methodology for generating autonomous systems.

(1) Load the system entity structure (SSL) which organizes all available domain knowledge including experiment knowledge and structure knowledge;
(2) Interpret natural language-like task goal to map into the predefined domain knowledge (SSL);

(3) Check ENBASE whether there are already existing plans (PESs) and if not then select necessary models and sequence actions by pruning to the goal knowledge part of the system entity structure (EXPERIMENT);
(4) Construct model structure by pruning the execution knowledge part of the system entity structure (STRUCTURE);
(5) Transform the model structure into an autonomous system architecture by synthesizing component models from the model base, MBASE.
(6) Test the correctness of the plan by simulation.
(7) In case the plan is not satisfactory, reprune (replan) the SES by issuing the repairing goal and repeat from step (3).
(8) Save the results for reuse; states of each model into MBASE and PES into ENBASE.

The reuse of existing plans, formulated as pruned entity structures, is similar in spirit to the case-based planning approach[5,6].

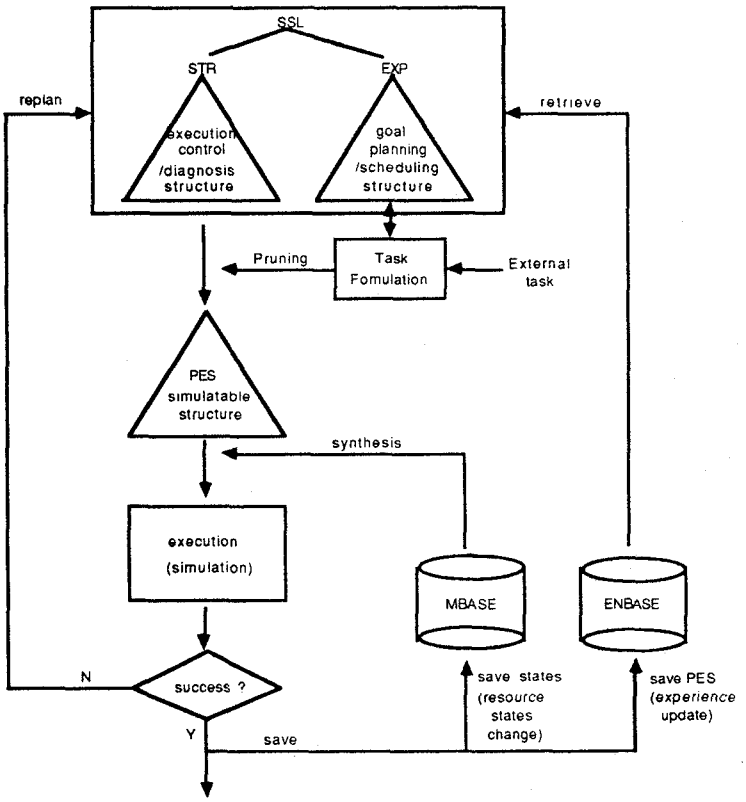

Fig. 7. Autonomous System Generation Methodology Using SES/MB

6. CONCLUSIONS

The key concepts and issues of advanced planning including execution (simulation) are categorized as follows:

(a) Abstraction-related hierarchical planning.
(b) Reusability of old plans.
(c) Time constraints (goal condition checking).
(d) Replanning, self-diagnostic and self-repairing.

Building on the basis of the SES/MB framework as implemented in DEVS-Scheme, we have extended the ability of our knowledge/model base tools to support model-based design of high autonomy systems through the ability to generate a family of planning alternatives and to build a hierarchical event-based control structure. We have developed the methodology for the autonomous system generation by integrating the execution structure (resources structure) and planning structure (resources allocation structure). The methodology exhibits solutions to points (a), (b), (c) and the self-diagnosis part of (d).

As described above, the model-based approach, employing multi-abstraction, multi-level, and event-based control logic, has been demonstrated in the design of a robot-managed space-borne laboratory environment. We have applied our planning system to fluid handling under microgravity conditions. Although much of the methodology has been implemented, much work remains to complete and verify the working system. Also we will seek to extend the system to include self-repair capability and replanning using concepts from the case-based approach.

7. ACKNOWLEDGEMENTS

This research is supported by NASA-Ames Co-operative Agreement No. NCC 2-525, "A Simulation Environment for Laboratory Management by Robot Organization" and by a McDonnell Douglas Space Systems Grant for Knowledge-Based Model Management.

8. REFERENCES

[1] J. S. Hawker and R. N. Nagel, "World Models in Intelligent control Systems," IEEE Int. Symposium on Intelligent control, 1987, pp.482-488.

[2] W.J. Hurst and Mortimer,J.W., Laboratory Robotics : *A Guide to Planning Programming and Applications,*VCH Publishers, 1987.

[3] S.Steel,"Topics an Planning", in lecture Notes in Artificial Intelligent, J.Siekmann eds. Springer Verlag, 1987.

[4] D. E. Wilkins, *Practical Planning*, Morgan Kaufmann Inc., 1988.

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

[6] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasing*, Lawrence Erlbaum Associates, Inc., Hillside, New Jersey, 1989.

[7] E. D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier North-Holland, Inc., 1977.

[8] 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.

[9] B.P. Zeigler, "High Autonomy Systems: Concepts and Models", *Proc. on AI, Simulation and Planning in High Autonomy Systems*, Tucson, March, 1990.

[10] B.P. Zeigler and S.D. Chi, "Hierarchical Systems Architecture for Artificial Intelligence", *Proc. on 34th Int. Soc. System Sciences Conf.*, Portland, July, 1990.

[11] B.P. Zeigler and S.D. Chi, "Model-based Architecture Concepts for Autonomous Systems", *Proc. on 5th IEEE Int. Symposium on Intelligent Control*, 1990 (in process).

[12] 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, March, 1990.

[13] T.G.Kim, "A knowledge-based Environment for Hierarchical Modelling and Simulation", Ph.D Dissertation, Univ. of Arizona, 1988.

[14] T.G.Kim, and B.P. Zeigler, "ESP-Scheme : A realization of system Entity Structure in a LISP Environment," *Proc. in 1989 SCS Eastern Multiconference*, March 1989, Tempa, Florida.

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

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

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

[18] B.P.Zeigler, "Knowledge Representation from Minsky to Newton and beyond", *Applied Artificial Intelligence*, vol.1, pp87-107, Hemisphere Pub. Co. 1987.

[19] J.W.Rozenblit and B.P.Zeigler, "Design and Modelling Concepts', *Encyclopedia of robotics*, John Wiley, N.Y.,1987.

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

[21] B. P. Zeigler, *Theory of Modelling and Simulation*, Wiley, NY, (Reissued by Krieger Pub. Co., Malabar, FL, 1985), 1976.

[22] P.A. Fishwick, "Abstraction Level Traversal in Hierarchical Modelling", In: *Modelling and Simulation Methodology: Knowledge Systems Paradigms* (Eds.: M.S. Elzas, T.I. Oren, and B.P. Zeigler), North Holland Pub. Co., Amsterdam, pp. 393 - 430, 1989.

[23] 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, 1988, pp.299-309.

[24] W.J. Hurst and J. W. Mortimer, *Laboratory Robotics : A guide to Planning, Programming, and Applications*, VCH, 1987.