Lozano-Pérez, T. and Wesley, M.A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570.

Lozano-Pérez, T. February 1983. Spatial planning : a configuration space approach. *IEEE Transactions on Computers*, C–32(2):108–120.

Malcolm, C.A. (1987). *Planning and Performing the Robotic Assembly of Soma Cube Constructions.* Master's thesis, Department of Artificial Intelligence, University of Edinburgh.

Malcolm, C.A., Smithers, T., and Hallam, J. (1989). *An Emerging Paradigm in Robotic Architecture.* DAI Research Paper 447, Department of Artificial Intelligence, University of Edinburgh.

Peshkin, M.A. and Sanderson, A.C. December 1988. The motion of a pushed, sliding workpiece. *IEEE Journal of Robotics and Automation*, 4(6).

Selke, K.K.W., Shen, H.C., Deacon, G.E., and Pugh, A. (1991). A strategy for sensors and rules in flexible robotic assembly. *International Journal of Production Res.*, 29(2):279–291.

Shen, H.C., Selke, K.K.W., Deacon, G.E., and Pugh, A. (1987). A sensor-data driven, rule based strategy for error recovery in robotic assembly. In *Proceedings of NATO Advanced Research Workshop on Kinematic and Dynamic Issues in Sensor based Control.*

Srinivas, S. (1976). *Error Recovery in Robot Systems.* PhD thesis, California Institute of Technology.

Taylor, R.H. (1976). *The Synthesis of Manipulator Control Programs from Task-Level Specifications.* Report AIM 282, Stanford University.

Bibliography

Albus, J.S. (1981). *Brains, Behavior, and Robotics.* BYTE Publications Inc.

Brooks, R.A. (1982). *Symbolic Error Analysis and Robot Planning.* AI Memo 685, MIT.

Brooks, R.A. (1986). *Achieving Artificial Intelligence Through Building Robots.* AI Memo 899, MIT, Artificial Intelligence Laboratory.

Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23.

Brooks, R.A. (1989). A robot that walks: emergent behaviors from a carefully evolved network. In *Proceedings of IEEE Robotics and Automation Conference*, pages 692–696.

Brost, R. April 1986. Automatic grasp planning in the presence of uncertainty. In *Proceedings of IEEE International Conference on Robotics and Automation.*

Donald, B.R. April 1986. Robot motion planning with uncertainty in the geometric models of the robot and environment : a formal framework for error detection and recovery. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1588–1593, San Francisco.

Donald, B.R. (1988). Planning multi-step error detection and recovery strategies. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 892–897.

Donald, B.R. February 1990. Planning multi-step error detection and recovery strategies. *International Journal of Robotics Research*, 9(1):3–60.

Gini, M. and Gini, G. (1983). Towards automatic error recovery in robot programs. In *Proceedings of 8th International Joint Conference on Artificial Intelligence*, pages 821–823.

Gini, M. and Smith, R.E. (1986). Reliable real–time robot operation employing intelligent forward recovery. *Journal of Robotic Systems*, 3:281–300.

Lozano-Pérez, T. and Brooks, R.A. April 1985. *An Approach to Automatic Robot Programming.* AI Memo 842, MIT.

which can be used to investigate appropriate low level Behavioural Modules and their linking to increase the competence of the system.

7 CONCLUSIONS

The work in this paper looks at increasing the reliability of a Behaviour-based robotic assembly system by providing appropriate alternatives when it is deemed necessary. These decisions are taken by the system designer, and can be determined by the use of local information and sensory data available in the Behavioural Modules.

The variation and uncertainty present in the robot system can be dealt with completely in the on-line system. The decomposition of the system into separate planning and execution systems provides the advantages of simplifying the planning process which no longer needs to reason about robot motions and can plan instead for the more abstract part motions, and the execution system can deal with the translation of these part motions into robot motions as required. This type of system also provides the on-line Behavioural Modules with the advantage of easily changing both the strategies which are employed, and the sensors which are utilised, thus making the most of new technology when available. These can be slotted into the system to provide a new interpretation of particular part motions.

The system can be gradually built up from low level competence Behavioural Modules, which have been designed and tested to ensure the robust interaction between the hardware, software and the task itself, creating a system of higher competence as the levels increase. This type of partitioning allows the simplification of individual Behavioural Modules while not limiting the complexity of the system itself. As the system deals with the variation and uncertainty as it occurs within the on-line system, it is not necessary to remove the failure interpretation from the context in which it occurs. The diagnosis can take place within its context, and the testing of the system in the workcell ensures that the most likely solution can be determined.

The occurrence of possible error situations can be spotted due to the close relation between sensing, action, software, and hardware within the Behavioural Modules. These can also be recognised quickly, allowing the system to be immediately halted or perform alternative actions, rather than going on to perform inappropriate actions.

It is hoped that this type of Behaviour-based system architecture can be used to provide a reliable assembly system. Although initial experiments have been performed to examine the use of Exit States, the Ideal Execution Path and Alternative Execution Paths, the ideas have yet to be tested on a complex system. A useful basis for future experimentation has been built,

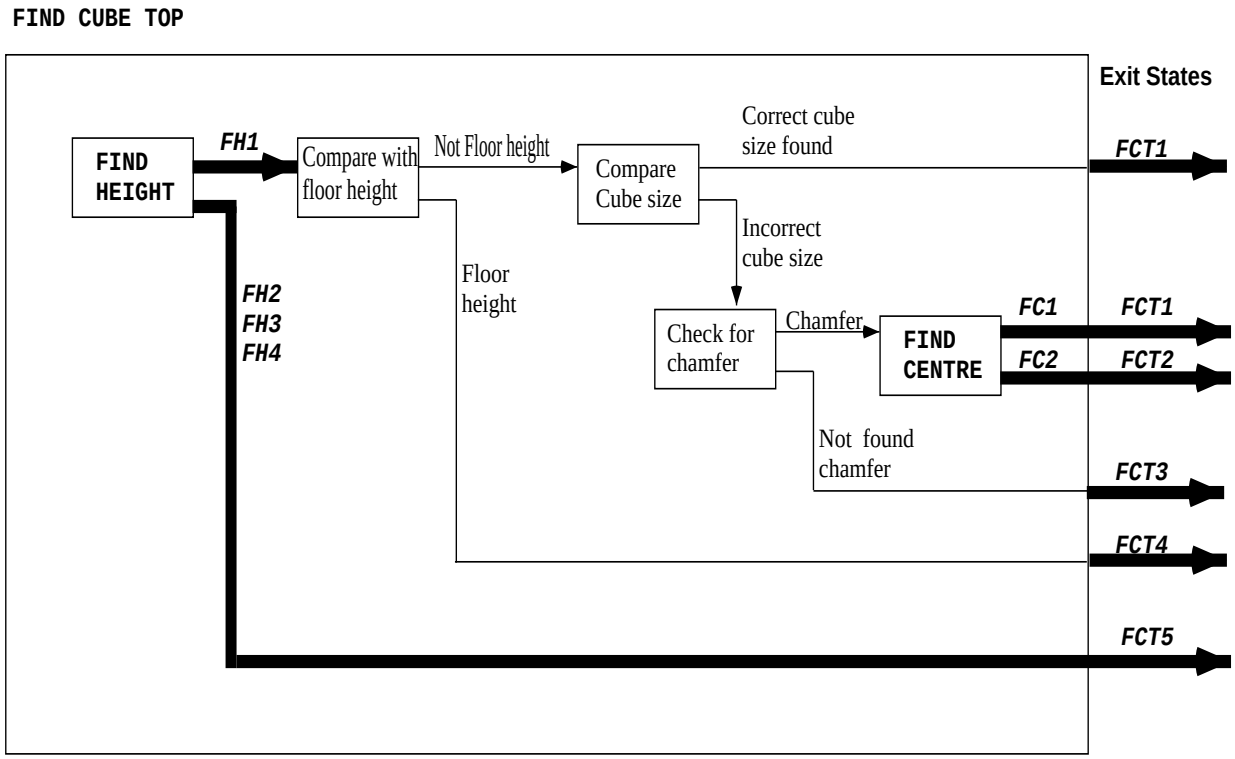

Figure 2: Find Cube Top Behavioural Module.

value is then checked locally to determine if the guarded motion did not encounter a Soma part and therefore ended up triggering the sensor at the level of the workcell floor. The value is also compared to see if it corresponds to a nominal expected height for the top of the cube. If it does, then the Behavioural Module is exited with Exit State FCT1. On the other hand, if the sensor has triggered at the height of one of the chamfers on the cube, a simple alternative Behavioural Module can be used to perform a search strategy to find the actual top of the Soma part, using the Behavioural Module **Find Centre**.

In this example, the Ideal Execution Path for the Behavioural Module followed the control path, FH1, 'Not floor height' and 'Correct cube size found'. The Alternative Execution Path followed the control path FH1, 'Not floor height', 'Incorrect cube size', and 'Chamfer'.

As with Albus's hierarchical approach to the use of servo-mechanisms (Albus, 1981), this method attempts to bring the system back to a correct state where the assembly can continue as originally planned. Whereas Albus's approach separates the control hierarchy and the sensory-processing hierarchy, using an internal world model to link them, this method links the sensors and task at the level of the Behavioural Module. The system does not need to know or reason about sensor types, which can be different depending on the chosen Behavioural Module.

Due to the design of the above type of system, it is able to contain the problem of variation and uncertainty in the on-line component of the system. This means that, unlike in many other systems, recovery is possible at different levels of the system, with no outright failure until the top of the hierarchy is reached. This allows the system great flexibility in dealing with the problem depending on what sensors and strategies are available to the system. An attempt can be made to rectify a particular failure at a low level, but if this does not succeed, it may still be possible to use an alternative at a higher level. If the **Pick-up-Method-2** Behavioural Module in Fig. 1 were not available at that level of the hierarchy to attempt to provide an alternative, the failure could be passed out as the Exit State DEMO-C, and the failure could be tackled at the level from which the Behavioural Module **Demonstration** was called.

6 FIND CUBE TOP EXAMPLE

An example of the above type of alternative Behavioural Module use can be found in an implemented example which was part of a system to assemble Soma parts into a large shape. Each Soma part consists of chamfered cubes glued together to form irregular shapes. The Behavioural Module described below is part of a larger strategy to locate the accurate position and orientation of such a part as it sits in the workcell.

The **Find Cube Top** Behavioural Module, seen in Fig. 2, utilises a guarded motion Behavioural Module, called **Find Height**, to return a point on the top of the Soma part above the surface of the workcell by moving vertically downwards over the part until a sensor on the tip of the gripper fingers is triggered. The Behavioural Module **Find Cube Top** is composed of the Behavioural Modules, **Find Height** and **Find Centre**. The rest of the Behavioural Module is composed from robot language code. If a valid height is returned, the Exit State FH1 is returned from **Find Height**. This

workcell being outwith its competence, or whether it partially completed the task, but then failed (such as a part falling from a gripper during a transporting motion). This is useful information which can be used to determine whether there are other Behavioural Modules which could cope with the unusual situation and bring the system back to the Ideal Execution Path. Alternative Execution Paths can also be entered into by information from the planner. The planning system does not know anything about the dimensions of the parts and the workcell, but the on-line system, has this knowledge and can determine if alternative Behavioural Modules should be used due to, for example, the height of an assembly becoming too great for the normal method of inserting a part. In Fig. 1, given the Ideal Execution Path following the control path PUM1-A, 'Data Correct', and P-A, an Alternative Execution Path is available through **Pick-up-Method-2** if the Exit State from **Pick-up-Method-1** is PUM1-B. If this Behavioural Module then has Exit State PUM2-A, then the task returns to the Ideal Execution Path and assembly can continue as before. An example of this type of alternative can be found if a simple touch sensor is normally used to locate a part in the workcell. If this fails to find the part in its nominal position, a vision system may be available which can scan a larger area and thus deal with a greater variation in the initial part positioning.

The extent to which the system follows the Ideal Execution Path can be taken as a measure of the suitability of the Behavioural Modules chosen to perform a task. Extensive deviation to Alternative Execution Paths shows that the Behavioural Modules on the Ideal Execution Path are not the best suited for the task, whatever the criteria for their original choice. If many deviations are made from the Ideal Execution Path onto an Alternative Execution Path from a normally reliable system, this may also indicate that, for example, a feeder has moved out of position and needs resetting. The system can still be kept running while the problem is investigated, using the Alternative Execution Path Behavioural Modules to prevent failure until the source is located. Alternative Execution Paths may consist of Behavioural Modules, or may merely perform local robot commands, depending on the extent of the deviation from the Ideal Execution Path. Alternative Execution Paths are incorporated by the system designer to bring the system back to a state where it can continue without external interference. The Alternative Execution Path may simply retry the original Behavioural Module which returned a failed Exit State if it is possible that it may succeed in the correct task if repeated. The Alternative Execution Path may instead employ a different sensor type if available, or may use different strategies.

5 EXECUTION PATHS

The Ideal Execution Path and the Alternative Execution Paths are defined control paths through the hierarchy of the Behavioural Modules. These paths can be followed by using the Exit States of Behavioural Modules plus local control decisions. The Ideal Execution Path describes the best route, according to the system designer, through the Behavioural Modules performing the assembly. Even though the Behavioural Modules are designed and tested to be reliable, it can be demonstrated that they are not always perfect at performing the assembly task, and therefore, alternative Behavioural Modules, those on an Alternative Execution Path, can be included to restore the system to the Ideal Execution Path without recourse to the planning system. Alternative Execution Paths can be used when Behavioural Modules on the Ideal Execution Path fail to correctly perform their task due to the actual workcell conditions being outwith the competence of the modules.

The Ideal Execution Path describes a control path through the Behavioural Module hierarchy, determined by Exit States from Behavioural Modules and other local information available to the system from the plan. In the example in Fig. 1 the Ideal Execution Path for the task performed by the Behavioural Module, **Demonstration**, can be designated as following the control route through **Pick-up-Method-1**, with Exit State PUM1-A, through the local robot code with control path 'Data Correct', and through the Behavioural Module **Place**, with Exit State P-A. In this instance, the returned Exit State, DEMO-A from **Demonstration** would signify that the task had been successfully completed. In another call of the Behavioural Module in a different context, the Ideal Execution Path may be designated as following the control route through **Pick-up-Method-1**, with Exit State PUM1-B, and through **Pick-up-Method-2**, with Exit State PUM2-B, leading to a returned Exit State of DEMO-C from **Demonstration** if the task is successfully completed. In any assembly, an Ideal Execution Path can be identified by the designer of the system which reflects the best way in which the assembly can be performed. The criteria for this can be diverse, ranging from time taken to perform the Behavioural Module, to the ease of using alternative strategies on failure.

If the Ideal Execution Path Behavioural Modules fail, or the workcell conditions are such that they are no longer the best modules available for the task, other Behavioural Modules may be available which can cope with the problem encountered. The Exit States can show whether the Behavioural Module could not attempt the task due to the conditions in the physical

Behavioural Module. As Behavioural Modules can be reusable in different situations, the context in which they are called plays an important part in determining whether the correct Exit State has been returned for one particular call. The concept of Exit States will be explained further in the next section.

4 EXIT STATES

The Exit States of a Behavioural Module reflect its internal control structure, with only one Exit State returned for each of its calls. The expected Exit State may be different depending on the context in which it is called. This information, passed up through the hierarchy in one parameter, can be used by the calling higher level Behavioural Module to determine if that low level Behavioural Module has successfully completed its assigned task. In the example in Fig. 1, the Behavioural Module, **Demonstration**, can have Exit State DEMO-A, DEMO-B, or DEMO-C, depending on which route has been taken within the Behavioural Module. The Exit States for a particular Behavioural Module are determined from local data, sensor data, and from Exit States returned from lower level Behavioural Modules. As Behavioural Modules can be general purpose and used in different parts of the system for different tasks, these Exit States prove useful to show what actually happens within the Behavioural Module in any one calling.

A Behavioural Module designed to perform a guarded motion may be called on one occasion to deliberately bump into a surface, yet on another occasion may be expected to move to a specific point without any interference. The Behavioural Module can be designed such that its Exit States will be different when both calls correctly perform their assigned task.

If, on examining the Exit State of a Behavioural Module, it is found that the module has failed to perform the task required of it in the context of its calling, it is then possible to use this information to try to prevent the system from automatically falling into an error situation which requires human intervention. Using a combination of Exit States from lower level Behavioural Modules, and sensory and planner information particular to the calling Behavioural Module, the Ideal Execution Path and Alternative Execution Paths for a system can be determined. The use of these to provide a more reliable system can be seen in the next section.

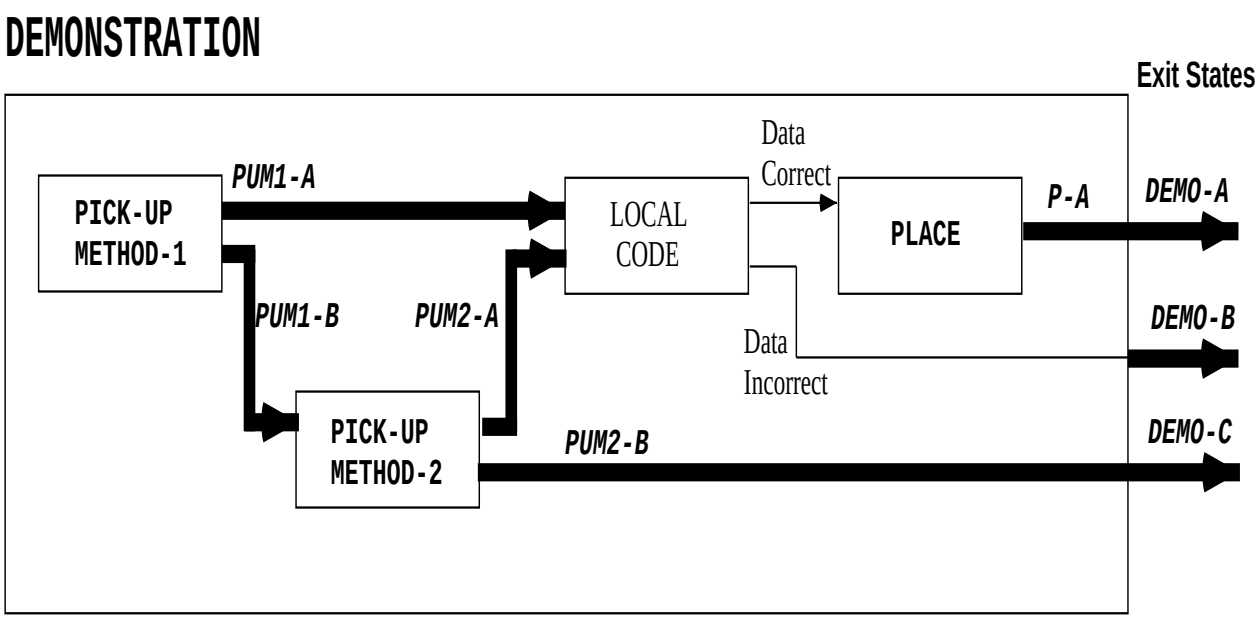

Figure 1: An example of a Behavioural Module.

Built in a hierarchical fashion, each Behavioural Module can be composed of any combination of other Behavioural Modules, robot commands in a robot control language, and purely computational modules. The Behavioural Modules can be designed by implementing basic low level competences and using these as building blocks to construct higher level Behavioural Modules with correspondingly higher competences. A height determining Behavioural Module can be composed of a series of guarded motion Behavioural Modules. Figure 1 shows an example Behavioural Module **Demonstration**, which is composed of the lower level Behavioural Modules called **Pick-up--Method-1**, **Pick-up-Method-2**, **Place**, and some robot language code. This diagram is used throughout the rest of the paper to illustrate the ideas presented.

Emphasis is placed on using information directly from the real world as much as possible, leading to a tight linking of the software, hardware, and the task. Behavioural Modules are designed, implemented, and tested in the real workcell, thus ensuring that they are correctly adapted to the chosen environment. As far as is possible, the variation and uncertainty present in the task for which the Behavioural Module is designed is dealt with inside the module by using variation-reducing strategies devised by the designer and by correctly adapting sensors to the task.

As the planner provides only part motions, and does not dictate how these part motions will be performed (unlike some systems where each robot motion is sent down from the planner), the execution system has complete freedom to interpret these part motions according to what Behavioural Modules are available. This type of system can provide the flexibility to change old strategies with new and improved ones without changing the interface to the planner. The Behavioural Modules can be designed to limit the complexity of the individual Behavioural Modules without limiting the complexity of the system. This means that full use can be made of introducing new sensors into the system or designing new strategies to perform a sub-task. The hierarchical structure of this type of system also allows Behavioural Modules to be slotted in to the execution hierarchy to provide alternative competences if these are required. Sections 4 and 5 look at how appropriate information can be made available to direct the control path to these alternatives when necessary.

Each Behavioural Module is designed to allow the use of sensors to closely monitor its performance while performing its task. There is no concept of success or failure within a Behavioural Module, but instead there are a number of Exit States which correspond to the control paths available in the

separate information processing functions. A move is made away from collecting and processing data to form a single representation of the world in which the robot operates, to integrating sensor information at the lowest possible level to direct more responsive behaviour. This allows real time control of the modules which can respond to internal action priorities and to external world constraints (Brooks, 1986a; Brooks, 1986b; Brooks, 1989).

2 THE SOMASS SYSTEM

An example of a Behaviour-based robotic assembly system is provided by the SOMASS system (Malcolm, 1987; Malcolm et al, 1989). This system provided a basis for experimental work to test the ideas which are examined in this paper. The SOMASS system provides a complete assembly system for building Soma shapes from the planning stage through to the actual assembly by an Adept robot. The SOMASS system is divided into two parts: the planning system and the execution system. The planner does no explicit reasoning about the variation or uncertainty present in the real world of the assembly workcell, but instead leaves the problem of dealing with this to the on-line execution system. The assembly task is planned in terms of part motions which are realised in the robotic assembly cell by Behavioural Modules which are designed and engineered to cope with a certain amount of variation in the workcell. The planner uses some implicit assembly rules, such as putting the parts into the assembly vertically downward, which help to increase the reliability of the system. Since the earliest system employed no external sensing, it relied on variation-reducing motion strategies to increase the reliability within the Behavioural Modules in the on-line system.

An experimental system was developed in this framework to examine the use of sensors to reliably link the component Behavioural Modules. The original SOMASS planner which provided part motions, was retained, but the Behavioural Modules and the structure of the on-line system were changed to incorporate the ideas in this paper. The work described here examines the structure of this type of system, which includes sensors to allow decision making and choice of future actions.

3 BEHAVIOURAL MODULES

The Behavioural Modules in this work are regarded as modular units which combine hardware and software to reliably perform a task in the real world.

of the failure. This leads to problems in knowing how much information is needed to uniquely determine what has occurred as, in different contexts, similar sensor readings may represent different problems. It is only possible to diagnose what has been specifically included in the knowledge base.

Worst Case Analysis systems concentrate on producing plans to provide systems which will never fail, given the constraints imposed on the system by the designer. This method tries to guarantee that the plans which are produced will always work. Much use is made both of modelling the workcell and mathematically analysing the task, leading to computationally expensive systems. Research into this type of system ranges from fine motion planning (Taylor, 1976; Brooks, 1982) to planning the task in configuration space (Lozano-Pérez & Wesley, 1979; Lozano-Pérez, 1983). Later work expands the use of configuration space to compliant fine-motion strategies (Lozano-Pérez & Brooks, 1985). Donald (Donald, 1986; Donald, 1988; Donald, 1990) examines the problem of the failure of such plans when the goal state cannot be reached. Work has also been done on modelling the mechanics of manipulation (Brost, 1986; Peshkin & Sanderson, 1988). By the very nature of this type of work, it looks at dealing with the worst possible scenario which could be faced by the system, and attempts to find a plan which will work in these tight constraints. This precludes many strategies due to the necessity to overestimate the tolerances in the workcell, which will work in the real workcell, and often rejects many which are useful and efficient.

Competent Execution Systems, one type of which, the Behaviour-based approach, is used as the foundation of the work presented in this paper, plan for a simplified ideal world and rely on the run-time component of the system to reliably perform the assembly and deal with the variation and uncertainty. These systems rely on making information available when the system needs it and modularising the assembly process to enable useful incorporation of sensory data at appropriate points. Examples of this type of system architecture can be found in (Albus, 1981), and work on Information Space, which partitions the sensory input into areas which have associated actions to keep the assembly on the correct path, is described in (Shen et al, 1987; Selke et al, 1991). The main research problems associated with this method are finding useful modular boundaries, determining the modular architecture of the system, and deciding how to integrate high level control to achieve long term goals.

The Behaviour-based approach to programming robots attempts to decompose the task into reliable task achieving modules rather than into the

1 INTRODUCTION

A major problem with the use of robots to perform assembly tasks is the presence of variation and uncertainty in the workcell. Variation in part dimensions and positioning, and the uncertainty[2] introduced into the robot system when using sensors to more accurately measure workcell and part variation, can combine to produce an error situation which requires human intervention to restart the system after an unsuccessful assembly attempt. This has led to tightly engineered robotic systems which, like specialised hard automation systems, are specific to a particular application and thus lose the generality and flexibility originally promised by the introduction of robots to assembly.

Research into the failures of robotic assembly systems due to variation and uncertainty in the system has led to the development of several strategies designed to produce robot systems which can perform reliable assemblies. These strategies range from incorporating sensors specifically designed to catch a particular recurring error, through automatic monitoring systems, to manipulation strategies designed to reduce the occurrence of errors. The methods can be divided into three categories,

1. Error Detection, Analysis, and Recovery Strategies.
2. Worst Case Analysis.
3. Competent Execution Systems.

Error Detection, Analysis, and Recovery systems approach the problem of providing reliable systems by allowing errors to occur during the running of the assembly and attempting to perform actions to recover the system from an error situation (Srinivas, 1976). The assembly system is run until a failure is detected, a diagnosis is made of the cause of the problem, and recovery actions are planned for and performed. Much research has gone into automating the above process, with concentration focusing on introducing sensing to monitor the workcell in order to detect the failures, and setting up knowledge bases both to determine what has occurred and to decide upon the best course of action (Gini & Gini, 1983; Gini & Smith, 1986). This requires the comparison of expected and actual sensor values, and the extraction of enough data from the on-line system to determine the cause

[2] This term is defined differently in this work from its normal use in literature where sensors are usually seen as reducing uncertainty.

BEHAVIOUR-BASED ROBOTIC ASSEMBLY SYSTEMS: RELIABILITY OF BEHAVIOURAL MODULES.

Myra S. Wilson.[1]
Department of Artificial Intelligence,
Edinburgh University,
5 Forrest Hill,
Edinburgh, EH1 2QL.

23rd ISIR Conference,
Barcelona.
October 1992.

Abstract

This paper examines a method of improving the reliability of a Behaviour-based robotic assembly system. Robotic assembly systems in general suffer from the problem of variation and uncertainty build up which causes failure to complete the assembly correctly. The Behaviour-based approach to robotic assembly looks at decomposing the task into modular units, called Behavioural Modules, which reliably perform the assembly task. Even so, these Behavioural Modules can still fail due to the workcell conditions being outwith their competence. This paper examines a method of using appropriate localised information to provide alternatives in the event of such a failure. This introduces the concept of *Exit States* (which inform the system of which internal control path was taken through a Behavioural Module), the *Ideal Execution Path* (the best control route through the hierarchy of Behavioural Modules), and *Alternative Execution Paths* (an alternative control route containing Behavioural Modules which can be used in the event of failure).

Keywords: Robotic Assembly, Reliability, Behaviour-based, Exit States, Ideal Execution Path, Alternative Execution Path.

[1]The author wishes to acknowledge the financial support of SERC and the supervision of Dr T. Smithers and C.A.Malcolm during the development of this work.