Robotica (1997) volume 15, pp 181–198. © 1997 Cambridge University Press

Discrete-event modelling, simulation and control of a distributed manipulation environment

Fazel Naghdy and Naeem Anjum

Department of Electrical and Computer Engineering, University of Wollongong, Northfields Ave., Wollongong, NSW 2522 (Australia) email: f.naghdy@uow.edu.au

SUMMARY

Conventional robotics has proved to be inflexible and non-generic. The concept of Distributed Manipulation Environment (DME) is introduced to overcome some of these shortcomings. This concept proposes a distributed approach to robotics and flexible automation. The work is concerned with modelling, simulation and event based control of DME. The modelling, conducted both at the atomic and the coupled level, is quite generic and provides a framework for static and dynamic behaviour analysis of DME systems. The simulation models serve as a mean of performance evaluation of the system on a computer before the actual implementation in real time. The event-based controller provides a simple and robust control scheme. The controller, itself, can be tested, validated and finely tuned through simulation before implementation. The feasibility of the modelling technique is demonstrated through a case study.

KEYWORDS: Distributed Manipulation Environment; Discrete-Event-Modelling; Computer Simulation; Event-Based control.

1. INTRODUCTION

Industrial robots were introduced in manufacturing industry as a flexible and intelligent tool for automatic manipulation. The experience gained over the last two decades has, however, shown that the employment of an industrial robot does not always produce a flexible automated system. An industrial robot is a flexible and programmable tool. The manipulation solutions offered based on an industrial robot are, however, mostly application oriented, non-generic and non-systematic. This has resulted in high complexity and high cost of systems developed based on industrial robots.

The solutions developed based on an articulated robot have normally proved unnecessarily complex to control, too difficult to integrate into a production line and too costly particularly for small to medium-size industries. Moreover, the approach has been radically different from the natural trends developed in the industry over the years and hence requiring specialised high-skilled personnel for its design, development and maintenance.

In the work conducted by Naghdy, a distributed approach to robotics has been proposed.[1,2] The aim has been to define and develop a robotics system which is more systematic, generic, flexible and economical than conventional systems. The theoretical basis of the proposed robotics is the concept of Distributed Manipulation Environment (DME). DME is a distributed and concurrent system formed by a network of Manipulation Modules that are entities capable of mechanical, informational, sensory and processing behaviour.

In order to control a DME to achieve a desired performance, its behaviour should be formally defined and modelled. Considering the nature of DME, it is possible to identify two different levels of modelling:

(a) The component level at which the behaviour of the individual manipulation modules working together is analysed and modelled.
(b) The functional level at which the interaction, coordination and sequence of the operation of the manipulation modules should be formally and systematically defined and modelled.

At level (a) the conventional analytical methods are used to describe the kinematics and dynamics of the manipulation modules. The modelling procedure for level (b) is fundamentally different from (a) and new tools and techniques are required. Study of the modelling of DME at this level, computer simulation of the model and its control system are the main issues addressed in this work.

A DME system is basically a Discrete Event Dynamic System whose operation can be specified as a chain of concurrent and sequential events. At present no general methodology exists to describe the dynamic behaviour of such a system. Researchers in the past have applied deterministic techniques such as Min-Max Algebra,[3] Finitely Recursive Process[4] and stochastic approaches like Markov Chains.[5] Queuing Networks[6] to model the discrete event dynamic systems. In this work Discrete Event System Specifications formalism, proposed by Ziegler, is employed.[7,8] The simulation models are implemented using Simscript which is a discrete event simulation package.

The Discrete Event System Specifications formalism does not only capture the dynamics and concurrent behaviour of the system but also provides a formal basis for specifying the dynamic model within the discrete event simulation environment.

In the course of the paper the concept of DME will be introduced. The DEVS approach and its application to DME will be described. The methodology developed for

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

discrete event modelling AND simulation of DME will be reviewed and the performance of the methodology will be demonstrated through a case study.

2. BACKGROUND

Discrete event modelling and simulation play a fundamental and important role in understanding and developing complex discrete event dynamic systems (DEDS). Such systems can be found today in a wide variety of technological areas such as flexible manufacturing, assembly and production lines, traffic systems, computer and communication networks, etc.[9]

Since a DEDS is a man-made system rather than a natural/physical one, no physical law exists for it, as they are for Continuous Variable dynamic systems (CVDS). The natural limits of materials and ergonomics are then the only factors that may constrain a DEDS system configuration. System complexity can, therefore, easily explode in a combinational fashion making performance analysis and optimization a difficult job.[10]

The existing analytical approaches developed for performance evaluation of a system are often insufficient and inapplicable to complex discrete event dynamic systems. These methods are based on unrealistic assumptions and many simplifying approximations. The resulting inaccuracies, owing to these assumptions and approximations, make the result unacceptable for complex DEDX of practical interest. In such a situation discrete event modelling and simulation become the most important tools for:[11]

- Understanding the behaviour of the system.
- Testing the operation of the system during development.
- Getting estimates of the performance measures of the system.
- Making improvement and modification in the system before it actually goes into service.

Such analysis and study of a system may otherwise be too costly, impractical or even impossible to be conducted directly on the system.

Most of the manufacturing and automation applications have a DEDS structure at some level of description.[12] Discrete event modelling and simulation thus may be applied to these systems not only at the design stage but also in handling of tasks during daily routine operations. Some applications of discrete event modelling and simulation have already been successfully implemented in industry while others are finding their ways very rapidly. In this section three applications of discrete-event modelling and simulation in real time will be briefly reviewed.

The first application, known as Short Term Planning (S-Plan) system, has been developed for the UK paper and board industry.[13] This system performs the function of factory wide control that includes the integration, control and management of the whole of the business and production process. It is being currently used in Stoneywood mill at Wiggins Teape. The Stoneywood mill is one of the largest and most complex of the UK paper mills and has a manufacturing capacity of 70,000 tones per annum of high value added paper.

The S-Plan at Stoneywood controls the whole of the manufacturing process on a 24 hour a day basis. It translates orders into programmes of work for each of its 40 production centres. Generally orders for a period of six weeks are planned. There might be as many as 1000 orders being processed on the shop floor. The system also accepts feedback dynamically from the shop floor concerning the status of all the operations and updates itself rapidly in the event of any break down, change of resources or sudden change in the production pattern. It is also capable of providing information about bottle necks, slack period in the current production plan, and the effects of overtime, sub-contracting and new orders, etc., to management. The benefits obtained by using S-Plan include:[14]

- Finishing productivity up by 25%
- Site output up by 20% and
- Customer complaints reduced from 27/1000 to 9/1000.

The second application of Discrete event modelling and simulation is an advanced robot-controlled instrumentation for Fluid Handling Laboratory (FHL) in a semi-autonomous environment.[15] FHL is part of Life Saving Module (LSM) of NASA's Space Station Freedom (SSF) project that will serve as a platform to conduct long term scientific experiments in space. The LSM of SSF is aimed to carry out experiments related to space medicine, gravitational biology, genetics and biochemistry. FHL will also play an important role for many experiments being planned in manufacturing and biotechnology.

The FHL has been modelled and simulated by discrete event methodology in assigning responsibilities to an organized group of robots for routine handling of fluids. Its design is based on the discrete event sequence of units' operations to be carried out to bring a real process from one initial state to a desired one. For example, the operation of a water sterilising unit used in FHL may be specified as a chain of three events of filling a bottle with water, placing it in a heating spiral and removing it when the required temperature has reached. Scheduling of transition from one state to another is based on time-to-next-event values obtained from trajectories of the dynamic model. Each unit operation is associated with a set of sensors for detecting its initializing and goal states.

The last real-time application of discrete event modelling and simulation is concerned with a comprehensive computer-aided manufacturing system simulation tool.[16] This tool has been reported by the author as one of the largest applications of discrete event modelling and simulation in industry. It is currently being used at Wright Peterson Air Force Base, USA, as a part of Integrated Computer Aided Manufacturing Program (ICAM). The ICAM performs simulation of complex manufacturing processes and provides information about work flow within the factory, raw material inventories, shipping of the finished goods, etc. It includes a graphic language for representing systems, a database for

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

maintaining system configuration and a simulator for analysing system performance.

3. DISTRIBUTED MANIPULATION ENVIRONMENT

Distributed Manipulation Environment (DME) proposes a distributed approach to robotics and flexible automation.[1,17] In the context of this work a distributed manipulation environment is an inherently distributed, intelligent, and programmable system that has been hardened in mechanics and control software to perform a specific type of manipulation. The basic unit of DME is a manipulation module which is a stand-alone, autonomous and intelligent unit. A manipulation module is capable of mechanical, informational, sensory and processing behaviour and is loosely coupled to other manipulation modules in the system. Each manipulation module has mechanical links for connection to other manipulation modules, electronics links to communicate with them and sensory links to sense and realise its environment as shown in Figure 1.

There are different types of manipulation modules defined to perform different tasks. These manipulation modules, based on the type of actuation they perform, can be categorised as shown in Figure 2.

An ensemble of manipulation modules and standard/non-standard hard automation components produce a Distributed Manipulation Environment which is configured and linked (hardened) to perform a specific task. Each manipulation module can be programmed individually. All these modules communicate locally and coordinate their operations on the basis of information received from their neighbours and a priori knowledge on the required task. Such a coordination is vital to avoid conflict and undue competition for available resources. The efficiency of the whole system depends on this coordination rather than the capabilities of the individual manipulation modules.

The distributed approach employed in DME not only simplifies the overall design and development of the system but is also independent of the number of manipulation modules used in the system. This number can vary from one system to another, depending on the functionality of the system. An existing DME can be

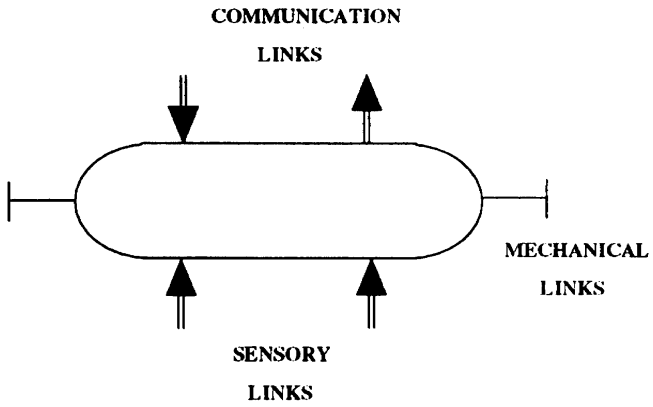


Fig. 1. Manipulation Module: A Building Block of DME.

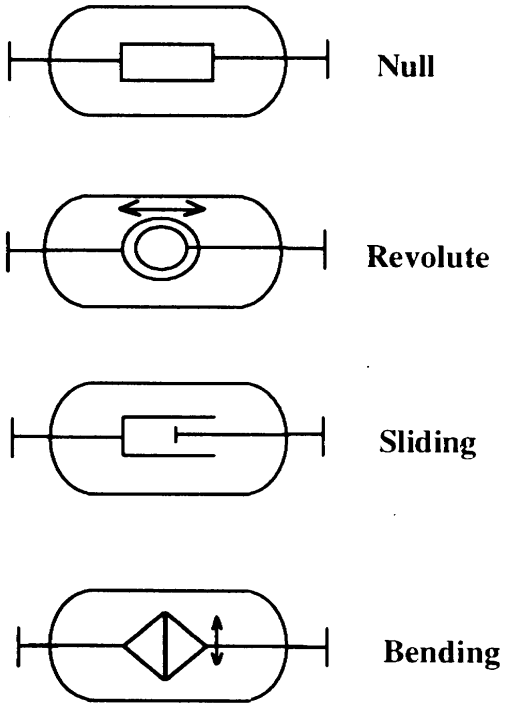


Fig. 2. Types of Manipulation Modules.

easily modified to perform a different task by adding or removing manipulation modules and changing their configurations.

The result is a highly modular and flexible design, reconfigurable at minimum effort and cost. This is in contrast to the conventional robotics where radical modifications in the overall system are required to accommodate any change in the task being carried out by the system.

4. DISCRETE EVENT MODELLING OF DME

A DME system is inherently discrete and can be viewed as a dynamic system with discrete state space and piecewise constant state trajectories. The time instants at which input and state transitions occur are usually irregular. This is similar to the general definition given by Ramadge of a discrete event dynamic system.[18]

The operation of a DME system is specified as a chain of concurrent and sequential events. These events in DME are the state transitions of the manipulation modules and the entities being manipulated.

The discrete event modelling of DME is expressed by DEVS formalism that focuses on the instant changes of a set of variables as the result of event happenings.[8,19] This generates time segments that are piecewise constant but usually spaced unequally as the time intervals between event occurrences are not constant. The modelling of DME takes place at two different levels. They are called atomic and coupled models and are defined in the following sections.

4.1 Discrete event system specifications formalism

Discrete event system specifications (DEVS) formalism was proposed by Ziegler in 1976.[7] The DEVS formalism provides a formal basis for specifying discrete events

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

dynamic systems that are amenable to mathematical manipulation for their behaviour analysis.

The DEVS is a structure as

$$\langle U, Y, S, \delta, \lambda, ta \rangle \quad (1)$$

where

- U is the input set
- Y is the output set
- S is the state set
- δ is the transition function
- $\lambda: S \rightarrow Y$ is the output function
- $ta: S \rightarrow R^{+}_{0,\infty}$ is the time advance function where $R^{+}_{0,\infty}$ is the set of non-negative Reals with ∞ adjoined.

The transition function δ can further be divided as follows:[8]

- $\delta_{\text{int}}: S \rightarrow S$ is the internal transition function and
- $\delta_{\text{ext}}: Q \times U \rightarrow S$ is the external transition function that is applied to the events in the input signals with Q defined as

$$Q = \{(s, e) \mid s \in S, 0 \leq r \leq t_a(s)\}$$

The above DEVS model, like a system theoretical model, may be considered as a black box that contains a process and produces outputs in response to inputs. A DEVS model, however, differs from the classical system theory in its inclusion and emphasis on the concept of an event.[20] An event can be either internal or external. An internal event is defined by δ_{int} and occurs when some conditions of its occurrence are fulfilled. The external events are the input stimuli and are modelled by δ_{ext}.

Furthermore, time in DEVS model does not increase continuously or in fixed steps. It progresses from one event to another so that the system time gets incremented in chunks of variable size.

4.2 Atomic model

The building block of DME is the manipulation module. The atomic model (AM) in DME, therefore, specifies the manipulation behaviour of a single manipulation module and can be defined as:

$$AM = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle \quad (2)$$

where

- $X = \{\text{Sensory}_{\text{inp}}, \text{Message}_{\text{inp}}\}$; the set of inputs.
- $S = \{s_1, s_2, \ldots, s_n\}$, the sequential state set of a manipulation module. The parameters that determine the DEVS state of a system depend on the specific task being carried out and hence may vary from one application to another. Typically, a pair (q, x) that relates the current state (q) of a manipulation module to the current input value (x) would represent state s_i in the sequential state set S.
- $Y = \{\text{Control-Signal}_{\text{out}}, \text{Message}_{\text{out}}\}$; the set of outputs.
- $\delta_{\text{int}} = S \rightarrow S$; the internal transition function. It specifies the next state to which the system will transit after the lapse of the time, defined by the time advance function, provided no external event occurs in the meantime.
- $\delta_{\text{ext}} = Q \times S \rightarrow S$; the external transition function that describes the system behaviour under the action of an input and
  $Q = \{(s, e) \mid s \in S, \ 0 \leq r \leq t_a(s)\}$; the total set that relates a sequential state (s_i) and the time elapsed (e_i) in that state.
- $ta = S \rightarrow R^{+}_{0,\infty}$; the time advance function. It defines the time for which the system remains in a given state before it undergoes the next internal transition provided that there is no change in the inputs in the meantime.
- $\lambda = S \rightarrow Y$; the output function that is used to generate an external output.

A manipulation module may be in active or passive phase at any instant of time. In the passive phase the time advance function, *ta,* is infinity and the manipulation module is locked to a physical location s_i. The manipulation module will stay in such a state indefinitely until an external event influences it. The internal transition function needs not to be defined for this state.

Under the influence of an external event, the manipulation module switches to an active state that is governed by the internal transition function and the time advance function. Once active, the manipulation module undergoes internal state transitions. All these internal state transitions are spontaneous in nature, i.e., the time advance function, *ta,* is zero, and the manipulation module is then said to be in a transitory state. The internal transition function and the time advance function are both defined by the dynamics of the manipulation modules.

An external transition is defined based on external events linked to each manipulation module. Depending on the type of manipulation module and the sensory linked attached to it; the external events may be generated by a sensor or triggered by another manipulation module as a request to carry out a task. An external event generated by a sensor linked to a manipulation module can interrupt the transition of manipulation module and force it to a passive state of s_{int}. The next state taken up by the system when influenced by δ_{ext} depends on the present state, external input and the time that has elapsed in the current state.

The functions required in this model can be defined as rules or algorithms that can be easily simulated. Figure 3 shows an atomic model in state s for an elapsed time e. The remaining time τ, after which the next internal transition would take place, can then be easily calculated as $t_a(s) - e$.

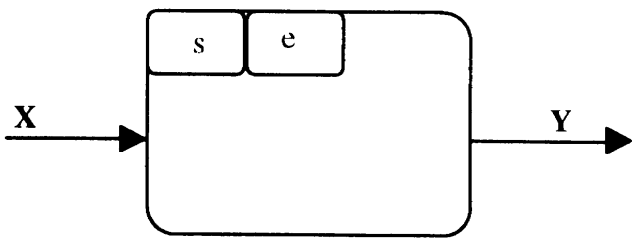


Fig. 3. An Atomic Model.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

4.3 Coupled model

A coupled model defines the manipulation behaviour of an ensemble of manipulation modules that are collaborating to carry out a certain task. In practice a coupled model (CM) describes how the atomic models of several manipulation modules can be combined to form a new model. It contains the following information:

- The set of manipulation modules operating in the coupled model
- The influences of each manipulation module
- The set of input ports through which external events are received
- The set of output ports through which external events are sent
- The coupling specification defining the external input coupling and the external output coupling. In this part, the manipulation modules whose input or output ports are connected to the input/output ports of the coupled model are identified.

A coupled model is formally described as

$$CM = \langle D,\ C,\ \text{Select} \rangle \tag{3}$$

where

- $D = \{\text{MM_A, MM_B, MM_C}, \ldots, \text{MM_N}\}$; is the set of component manipulation modules
  $C = \{C_1,\ C_2,\ C_3, \ldots, C_n\}$; is a set of ensembles that describes an individual manipulation module including its working relationships with other manipulation modules in the coupled model.

For each α in C

- $C_\alpha = \{M_\alpha,\ I_\alpha,\ Z_{\alpha,\beta}\}$ where
  - M_α is the atomic model of the manipulation module α
  - I_α is the set of influences of α and

for each β in I_α,

- $Z_{\alpha,\beta}$ is the interface map of α with its influence manipulation modules in the system
- The Select has rules or algorithms used to determine which manipulation module is allowed to carry out the next event.

A coupled model may contain any number of manipulation modules. The manipulation modules present in the coupled model are independent and operate synchronously or asynchronously. At many occasions during system operation, these manipulation modules would also be required to work concurrently. During the concurrent operation of manipulation modules, the DME system modes to a state that is the same resultant state reached if all the manipulation modules would have operated serially.

Similar to the atomic model, the coupled model is built for simulation through writing various algorithms. A coupled model consisting of two manipulation modules, M_α and M_β, is shown in Figure 4.

4.5 DME system model

A DME system model is modular and hierarchical in nature and may be constructed recursively from its component models, each being atomic or itself a coupling of atomic models. The atomic models are coupled to produce a coupled model according to a coupling specification that varies for different applications. The coupled model thus obtained may in turn be used as a component model to be coupled with other component models in a larger multi-component system to give rise to a modular and hierarchical model construction. It is, therefore, not necessary that a coupled model should contain atomic models only as its constituents. Hence as the system size grows and the number of interacting sub-systems increases, more and more coupling layers are added to the overall system model as shown in Figure 5.

In a larger multi-component system, the coupled model when used as a component model, would have the following structure.

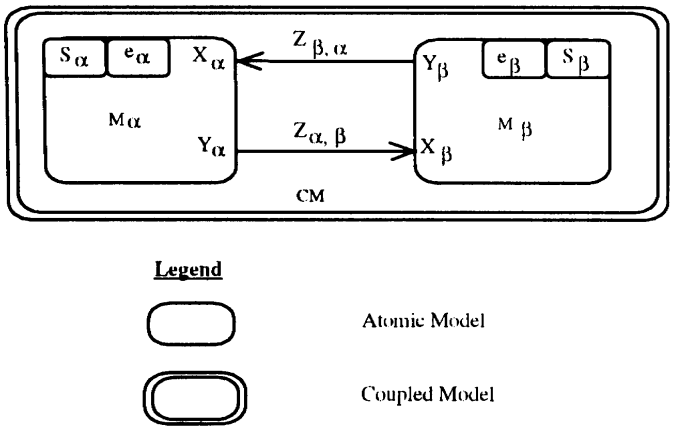
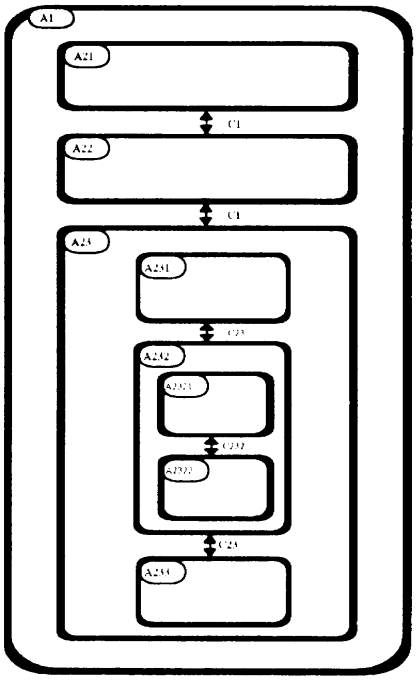


Fig. 4. A Coupled Model.



LEGEND

A1	Overall System Model
A21, A22, A231, A2321, A2322, A233	Atomic Models
A23	A Coupled Model of A231, A232 and A233
A232	A Coupled Model of A2321 and A2322
Cn	stands for Coupling Specifications

Fig. 5. A Modular DME System.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

- INPUT = The cross product of input sets of those manipulation modules which have their influencers outside the coupled model
- OUTPUT = The cross product of output sets of those manipulation modules that act as influencers outside the coupled model
- STATES = The cross product of manipulation modules' state set
- TRANSITION FUNCTION = The resultant of the transition functions of all individual manipulation modules in the coupled model, i.e., resultant = $\delta_n(q_n, \ldots \delta_2(q_2, \delta_1(q_1, x_1)) \cdots)$
- OUTPUT FUNCTION = The resultant of the output functions of all individual manipulation modules in the coupled model

The state of the overall DME system, Q_{system}, at any instant of time is a vector of total states of all its components. The system moves into a new state whenever one or more manipulation modules in a component model move to a new state. This system state can be specified as

$$Q_{\text{system}} = (\cdots (s_m, e_m) \cdots)$$

The above expression suggests that at any given instant t, each component, m in the system has been in state s_m for the lapsed time of e_m. Since the time advance in state s_m is given by $t_{am}(s_m)$, the component m is scheduled for an internal transition at time $t + (t_{am}(s_m) - e_m)$. The next system state transition will now occur at a time which is the minimum of these scheduled times. Thus if the minimum of the residual time $(t_{am}(s_m) - e_m)$ over the components m is τ, the next transition will occur at time $t + \tau$.

The overall discrete model of a DME system is, therefore, a coupling of a set of component models. The component models assist to create a distributed hierarchically structured model of the system identical to the actual system. They also simplify the task of development, debugging and maintenance of the whole model.

5. DISCRETE EVENT SIMULATION OF DME

The simulation of a DME system at functional level is quite different from that of electrical networks or mathematical models derived from system characteristic equations. The discrete event simulation methodology for DME focuses on events which move the system from one state to the next, and assumes that nothing of importance takes place between the two consecutive events.[21] In this process the following points are of interest:

- The prior state of DME
- The time of occurrence of the current event
- Selected time advance function which is in fact the minimum of the time advance functions of all the events that may occur in the current state
- The set of constraints, if any
- The next state of DME

Let a DME system be initialized to state s_0 at time t_0. This initialization can be expressed as

prior state = unspecified
time = t_0
selected time advance = unspecified or ϕ
constraint set = null
next state = s_0

As this system evolves intime under events, its behaviour at time t and in state s will be represented as

prior state = s
time = t
selected time advance = t_a
constraint set = c
next state = s_n = unspecified

The next state s_n of the system and new system parameters will be obtained as follows.

a) On the basis of s and t_a, perform simulation to get s_n, the next state.
b) Calculate all possible $t_a(s_n)$ for the internal events that may occur in state s_n.
c) Find $t_a(s_n)_{\min}$, the minimum of all $t_a(s_n)$ calculated in (b).
d) Set the parameters for the calculation of state s_{n+1} as follows:
   time = $t + t_a(s_n)_{\min}$
   selected time advance = $t_a(s_n)_{\min}$
   prior state = s_n
   next state = to be calculated
   constraint set = c'

Steps a, b, c and d are then repeated again from this state to find out the next state and the system parameters.

The discrete event simulation models of DME are based on discrete event mathematical models, i.e. atomic and coupled models. The simulation behaviour is incorporated into the mathematical model using the same set of functions and parameters specified in the atomic and coupled models.

Both the atomic and coupled simulators have static as well as dynamic behaviour.[22,23] The static behaviour is expressed by defining the models in a proper format and in appropriate modules. The format mainly depends on the software implementing the simulation. The dynamic behaviour of the simulator is governed by the transition functions as explained in the next two sections.

5.1 Atomic simulator

An atomic model describing the manipulation behaviour of a single manipulation module was defined by (1). Since this model cannot be further decomposed, S_{AM}, the simulator assigned to this atomic model has the following structure and is intended to represent a segment of the domain under study.

S_{AM}
operation
data
interfaces

Operations operate on and manipulate the *data* to incorporate dynamics into the simulator while *interfaces* are used to communicate with the environment.

The atomic simulator has two state variables and three storage cells as shown in Figure 6. The state variables are

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

S and t_L. S represents a sequential state while t_L indicates the time of the last event. The three storage cells are denoted by t_N, e and τ and respectively store information regarding the time of the next event, elapsed time since the last state transition and the time remaining for the next state transiiton to occur. Taking t as the global time, the contents of the storage cells are well defiend as

$$t_N = t_L + t_a(s) \tag{4}$$

$$e = t - t_L \text{ and} \tag{5}$$

$$\tau = t_N - t \text{ or}$$

$$= t_a(s) - e \tag{6}$$

The simulator interacts with its environment that includes other simulators in a predefined manner via an input and an output port. Through the input port it receives input and undergoes a state transition using either external or internal transition functions. The dynamic behaviour of the simulator is expressed as follows:

a) Case of δ_{ext}, when receives an input (x, t)
where $x \in X$ and
t is global time
If t satisfies $t_L \leq t - t_N$

$$e = t - t_L$$

$$s_n = \sigma_{ext}(\mathrm{s}, \mathrm{e}, \mathrm{x})$$

$$\mathrm{t_L} = \mathrm{t}$$

$$t_N = t_L + t_a(s)$$

b) Case of δ_{int} when receives synchronisation signal to update state
If $t = t_N$

$$s_n = \delta_{int}(s)$$

$$t_L = t$$

$$t_N = t_L = t_a(s)$$

The simulator uses its output port to report t_N to the control module that manages the simulation time of the whole system.

5.2 Coupled simulator

A coupled simulator is associated with a coupled model and has the same structure as that of the coupled model. It consists of a number of component simulators, each responsible for a component model in the coupled model. These component simulators may be the atomic simulators or again coupled simulators of some other component simulators.

Mathematically a coupled simulator is defined (2). Correspondingly, the coupled simulator, S_{CM}, will have the following structure:

S_{CM}
component simulators; atomic or coupled
coupling scheme
interfaces

A coupled simulator thus incorporates a coupling scheme in its structure to cater for the interfaces needed among its components. This coupling scheme is implemented by a coordinator which is responsible for coordinating and synchronising the component simulators within the coupled simulator and handling external events.[24] Figure 7 shows a coupled simulator for n componnent simulators. For each component model D_i in the coupled model, there is a corresponding component simulator S_i in the coupled simulator. Upon the receipt of an external event, the coordinator applies this input to all the component simulators attached to it which then behave as explained in section 5.1.

The dynamics of the coupled simulator is expressed as:

a) Case of δ_{ext} when coupled simulator receives an input (x, t)
if $t_L < t < t_N$
send input (x, t) to each component simulators
wait until all component simulators are done
$t_L = t$
$t_N = t_N(\min)$ where $t_N(\min)$ is the minimum of all t_N in S_{CM}

b) Case of δ_{int}
At $t = t_N$
Select component simulator S_j with $t_N(\min)$
Apply δ_{int} to S_j
send input $(x_{j,k}, t)$ to each influencee S_k of S_j where $(x_{j,k}, t)$ is the input to S_k
from S_j obtained by using output function.
wait until simulator j and all of its influences are done
$t_L = t$
$t_N = t_N(\min)$

For a coupled simulator, a situation may arise when two or more component simulators want to utilize the same single resource at the same time while the resource cannot be shared. This situation can be handled by some tie breaking mechanism, the details of which will depend on the implementing software.

The external interface of the coupled model consists of an input and an output port. It receives input and synchronization signals through the input port and informs the control module about $t_N(\min)$ through the output port. This interface structure is the same as that

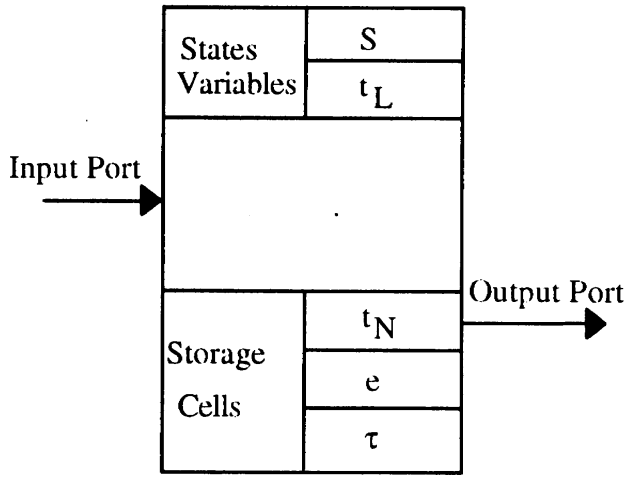


Fig. 6. An Atomic Simulator.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

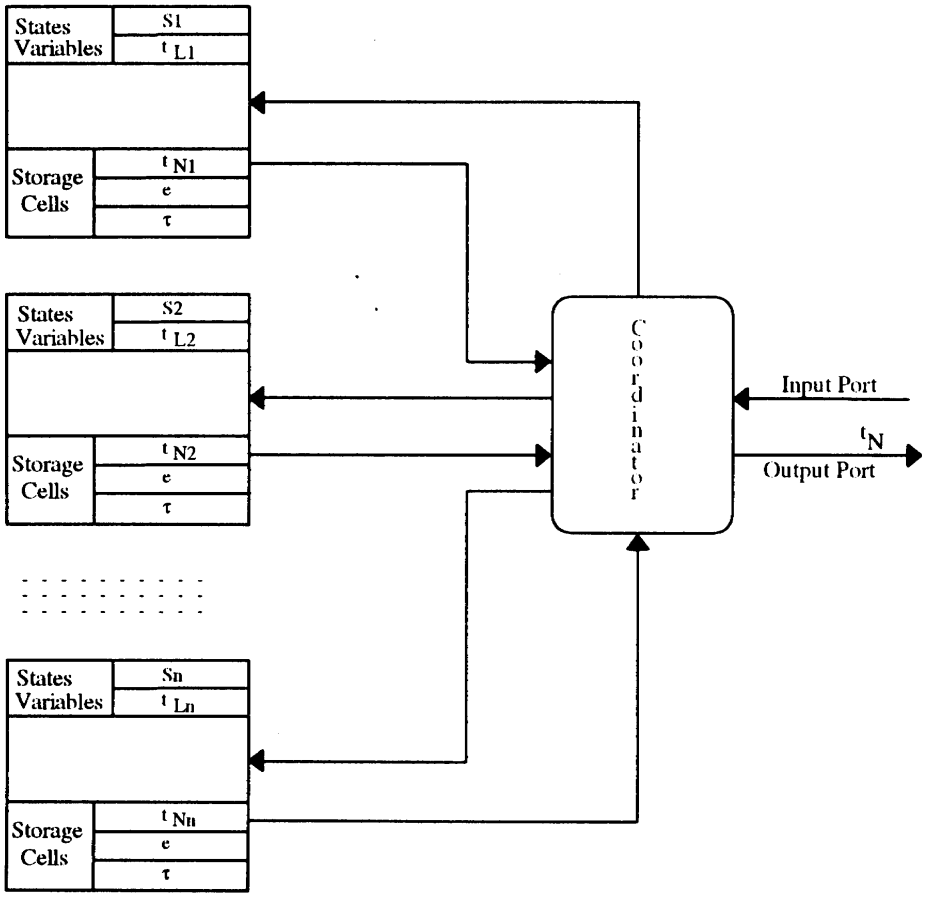


Fig. 7. A Coupled Simulator.

of the atomic simulator. Because of this, simulators may be combined in a modular and hierarchical fashion irrespective of the fact that a simulator at hand is an atomic or coupled in nature.

6. EVENT BASED CONTROL

In event-based control, each state transition of the system is associated with a definite time window.[25] This time window is determined by the discrete event model of the system and usually varies from one state to another. The sensors are assumed to respond to the controller within the time windows to confirm that the expected state transitions have occurred. The system moves from one state to another as long as the controller continues to receive the sensors' responses within the expected time windows.

The event-based control is produced based on a special case of DEVS model of a continuous state dynamic system. This model is referred to as Boundary Based Discrete Event.[26] The overall methodology to develop an event-based control model for a dynamic system is a three-step process:

(a) Work out the DEVS based discrete event model of the continuous state dynamic system to be controlled.
(b) From discrete event model of the system, develop a boundary-based discrete event model.
(c) Use boundary-based discrete event model to obtain event-based control model of the system.

6.1 DEVS model of a continuous state dynamic form

To obtain the discrete event model of the continuous dynamically system, the first assumption made is that the inputs to the system are piecewise constant time functions (e.g. sequences of a step function). Next, the continuous system is outfitted with a finite set of finite-states threshold-type sensors. These threshold-type sensors provide information about system state. For each state some of the sensors would be above the threshold while some others below. Thus the sensors divide the state space into a finite mutually exclusive state partitioning blocks. This is illustrated in Figure 8 in which a single 4-state threshold-type sensor divides the state space into 4 partitioning blocks.

The state of the system at a block is now represented by a part (q, x) where q is the current state of the dynamic system in that partitioning block and x is the current input. The discrete event model of the system will be the same as (2). The parameters δ_{int}, δ_{ext}, ta, λ are, however, defined based on this partitioning:

- $\delta_{\text{int}}(q, x) = (q', x)$ where q' is the system state at the next partitioning block
- δ_{ext} is the external transition function given as
- $\delta_{\text{ext}}((q, x), e, x') = (q'', x')$ where q'' is the state captured by the dynamic system after receiving a constant input x' for an elapsed time e given by $0 \leq e \leq ta(q, x)$. The system, however, remains in the same partitioning block. As a result no immediate output is produced by δ_{ext}.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

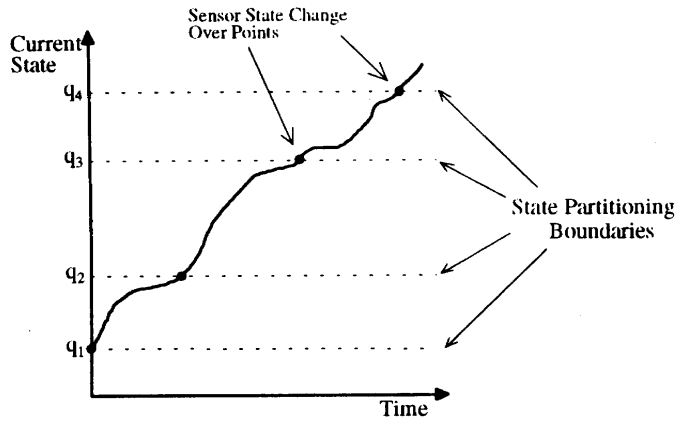


Fig. 8. Partitioning of State Space by a 4-State Threshold-Type Sensor.

- $ta(q, x)$ is the time advance function. It indicates the time required to cross the current partitioning block containing q, under an input x
- $\lambda(q, x)$ is the output function that generates the output of the system as it enters the state q' at the next partitioning block after the time given by $ta(q, x)$

The above discrete event model is thus related to the original dynamic system with a homomorphism that faithfully preserves a correspondence between the states of the discrete event model and the original system under the corresponding transitions and output operations. The aim is not to capture all the internal structure of the system but the input-output behaviour with a greater degree of accuracy.

The three stages of developing an event-based control model of a continuous dynamic system is illustrated in Figure 9.[26] The Boundary Based Discrete Event Model and Event Based Control Model also have structures similar to (2) with the parameters δ_{int}, δ_{ext}, ta, and λ redefined based on the partitioning illustrated in Figure 9.

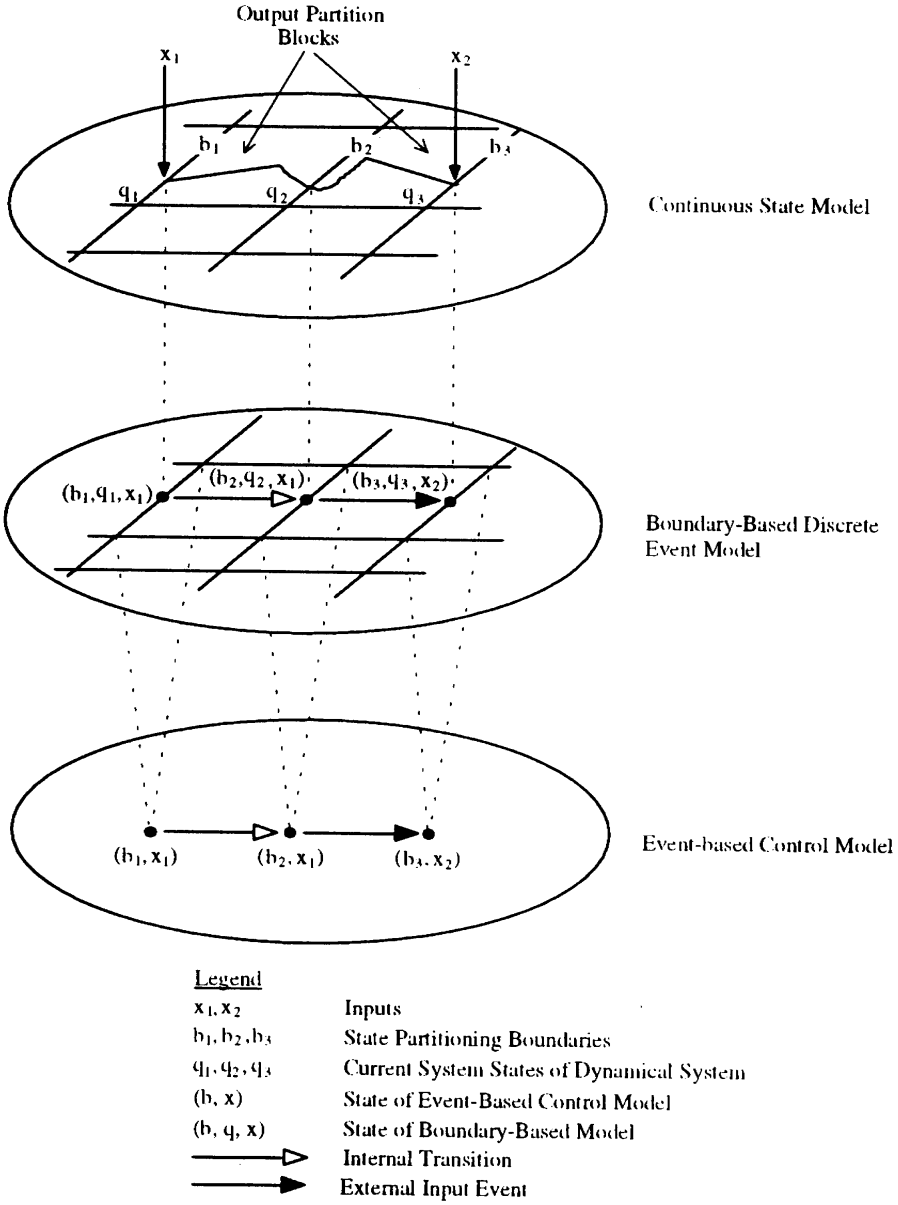


Fig. 9. Event-Based Control Model.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

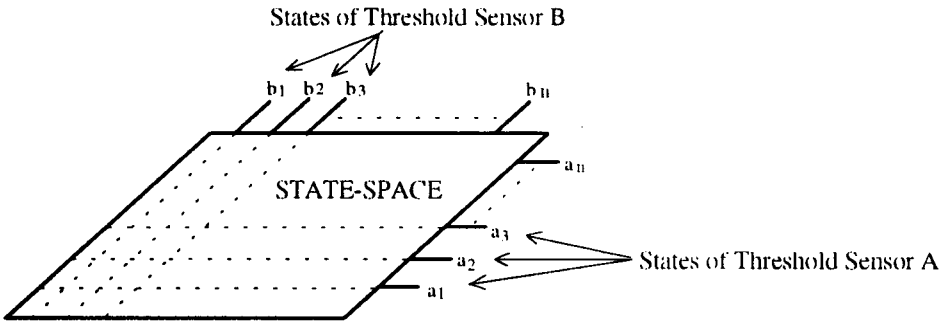


Fig. 10. State Partitioning Blocks.

6.3 Event-based control model for DME

Before developing an event-based control model for DME using the procedure described in the previous section, the specific nature of DME needs to be considered. A DME system has a distributed architecture. This distributed nature is also reflected in its control scheme. Hence an event-based control model will be developed for each sub-system. These event-based control models will run concurrently and communicate and coordinate with each other so that the overall system evolves in a smooth and disciplined manner.

Since the basic units of a DME system are the manipulation modules, the control of the system is achieved by controlling the manipulation modules. Each manipulation module in the system is, therefore, outfitted with a threshold-type finite-state sensor through finite set of bi-level sensors may also be used. All such sensors divide the state space into partitioned blocks forming a cellular grid structure as shown in Figure 10.

The size of the portioning block depends mainly on the specific job being carried out and indicates the range to be tolerated in the measurement of sensors. The threshold-type sensors are thus chosen according to the required width of the partitioning blocks. The crossing of the partitioning blocks in DME is considered as internal events and is modelled by internal transition function. The internal transition function along with the time advance function thus determines the time and state of the next boundary crossing. The changes in the input, e.g., speed control of a manipulation module, are modelled as external events and are implemented by external transition function.

The manipulation modules in a DME system work independently and cooperate with each other to achieve set goals. After transition to a new state, a manipulation module normally waits for a definite time so that the other manipulation modules in the system may respond accordingly. This waiting time may be zero or a function of the time advance functions of other manipulation modules in the system.

To model this waiting time, the notion of wait state is introduced. Each state of a manipulation module is followed by the wait state that is used to model its waiting time. Thus for the purpose of event-based control, the states of a manipulation module can be represented as shown in Figure 11.

This figure shows that after crossing a boundary, a manipulation module waits for a time of $ta(w_n)$ before it is activated again to reach the next boundary. The total time taken by the manipulation module from one boundary to the next is, therefore, the sum of $ta(w_{n-1})$ and $ta(s_n)$ where $ta(s_n)$ is the time for which the manipulation module remains active in state s_n under a given input.

The resulting control models of the manipulation modules are, therefore, interrelated with each other through the wait states and any malfunction of the system equally affects all these control models. The time windows for different states are determined by parameter variations of specific jobs being carried out, under normal operating conditions. These can be also calculated by a series of system simulation runs by varying the parameters within the range of their normal tolerance.

7. A CASE STUDY

The discrete modelling procedure mentioned for DME is applied to the system shown in Figure 12 to illustrate the application and potential of this technique. In this system a drilling task conventionally requiring a four-degrees of freedom robot is represented by a simple DME.

Here X is a plate on which a number of holes are to be drilled at different locations. The manipulation modules MM_A and MM_B are linear actuators used to control the plate X. A composite manipulation module attached to a drill bit is represented by CM1. This coupled model consists of a linear manipulation module MM_C and a rotary manipulation module MM_D and is capable of drilling holes in the plate at required positions. It is assumed that MM_A and MM_B move in steps rather than continuously and that all the points where holes are

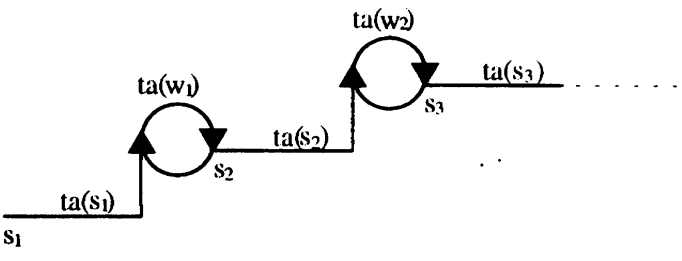


Fig. 11. Wait States Associated with a Manipulation Module.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

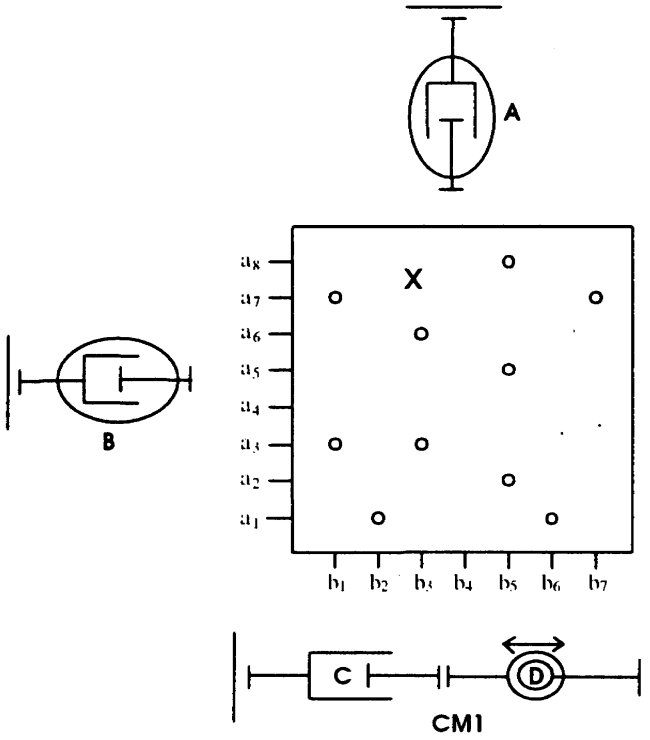


Fig. 12. DME Drilling System.

to be drilled are achievable. Thus to have a fine control over the whole work-space, MM_A and MM_B should have adequately large number of small steps. The accuracy of these manipulation modules is also important as the precision of the whole system depends on it.

Once the drilling head is located at the desired position through manipulation of MM_A and MM_B, CM1 is activated to drill a hole. The block diagram illustrating the interaction of the manipulation modules in this system is shown in Figure 13.

The DEVS model of the drilling system is provided in Appendix A. The task data including work-space size, the number of holes to be drilled, the location of the holes on the plate, system constraints, and similar information is fed into the task planner. The task planner, based on the system configuration and job accomplishment scheme, generates event labels for each manipulation module.

The scheduling algorithm employed in the selection scheme aims to minimise the manipulation time for the whole system in order to increase the productivity and reduce the cost. Genetic Algorithm (GA) is used to carry out this optimization. On the basis of the information produced by GA, the task planner generates event labels for different manipulation modules in the system.

7.1 Simulation Model of Drilling System

The discrete event model of drill is composed of 4 atomic models (MM_A, MM_B, MM_C and MM_D) and one coupled model CM_1 consisting of MM_C and MM_D. The atomic simulators, corresponding to these atomic models, have been expressed as processes. Each process contains a code specific to the nature of the manipulation module represented by the atomic model in the simulation model. The coupled model CM_1 is expressed by a coupled simulator that is also implemented as a process, cm1.sim. It contains code regarding how MM_C and MM_D are coupled and behave when called by the system.

The process objects of the drill model enter into the simulation at an explicit time by the occurrence of some specific events. They become active either immediately or at a prescribed activation time. Each time a process is activated, it executes statements representing changes to the system state and then is terminated. Figure 14 shows the evolution of the simulation model through these processes.[27]

The information regarding the number and locations

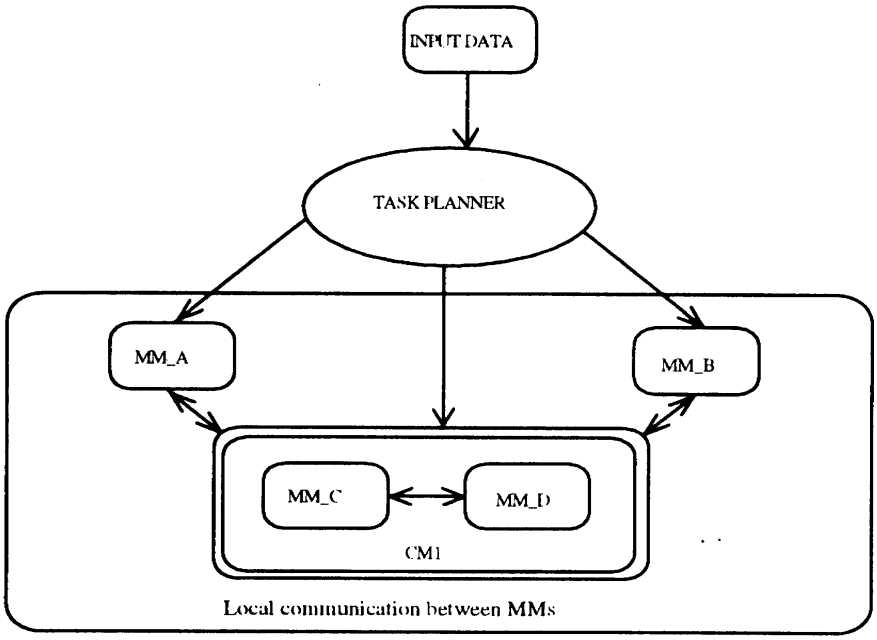


Fig. 13. Manipulation Modules in Drilling Model.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

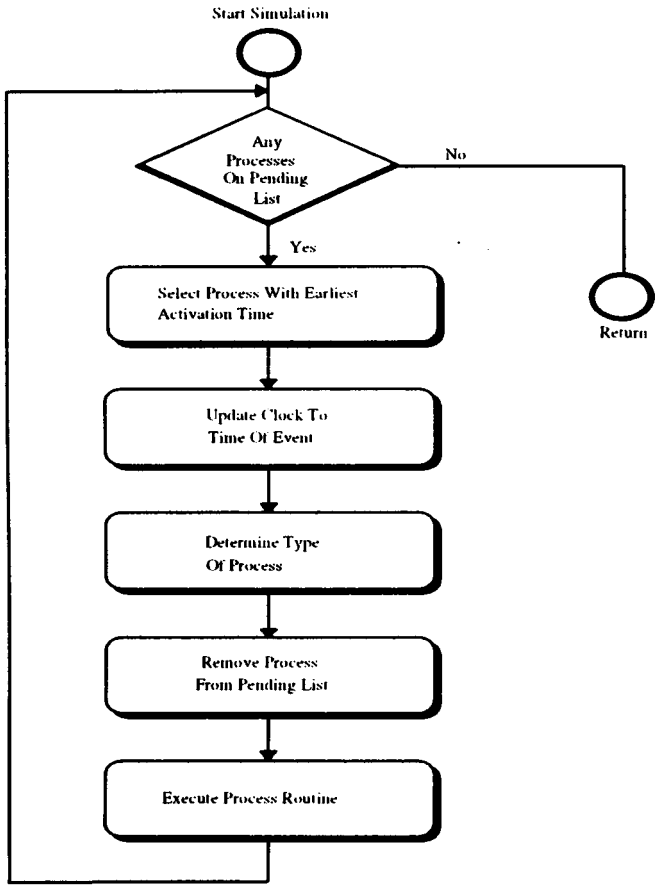


Fig. 14. Evolution of Simulation Model by Processes.

of holes on the plate is contained in two files *h.dat* and *v.dat.* these files act as a source of external events. For this example, data for 6 holes on the plate is contained in *h.dat* and *v.dat.* By changing data in these files, any number of holes can be drilled. The location of the holes can also be controlled. The speed of the manipulation modules and system simulation can also be set at the beginning of the task. Before starting drilling, marking of the points on the plates where holes are to be drilled according to the data contained in *h.dat* and *v.dat* is made. When a hole has been drilled at a marked position, it is also indicated.

The system state at various instants of times during the drilling process is illustrated through Diagrams B1 to B6.

7.4 Event-based control model of drilling system

The approach developed for the event-based control of DME is applied to the drilling system to obtain its control model which is then simulated on a computer. The DME model of drill hole system is expressed as

$$\text{Drill}\langle D,\ C,\ \text{Select}\rangle$$

and consists of 2 manipulation modules MM_A and MM_B and a coupled model CM1 which decomposes into manipulation modules MM_C and MM_D. The event-based control model for each manipulation module is developed taking into consideration the other manipulation modules in the system. These control models of all manipulation modules then collectively constitute the event-based model of the system.

7.4.1 Event-based control model for MM_A. To develop an event-based model for MM_A, a finite-state threshold-type sensor is attached to it that divides its state space into partitioning blocks. As the boundary of a new partitioning block is reached during the movement of MM_A, sensors send a confirmative signal to the controller. During normal operation the controller, therefore, always knows when the new partitioning block is reached by MM_A and it then issues appropriate control commands in order to either place MM_A in a wait state for a time given by *ta*(wait state) or to move it to the next desired boundary. Figure 15 shows the input events and the state partitioning of the work-space with respect to MM_A.

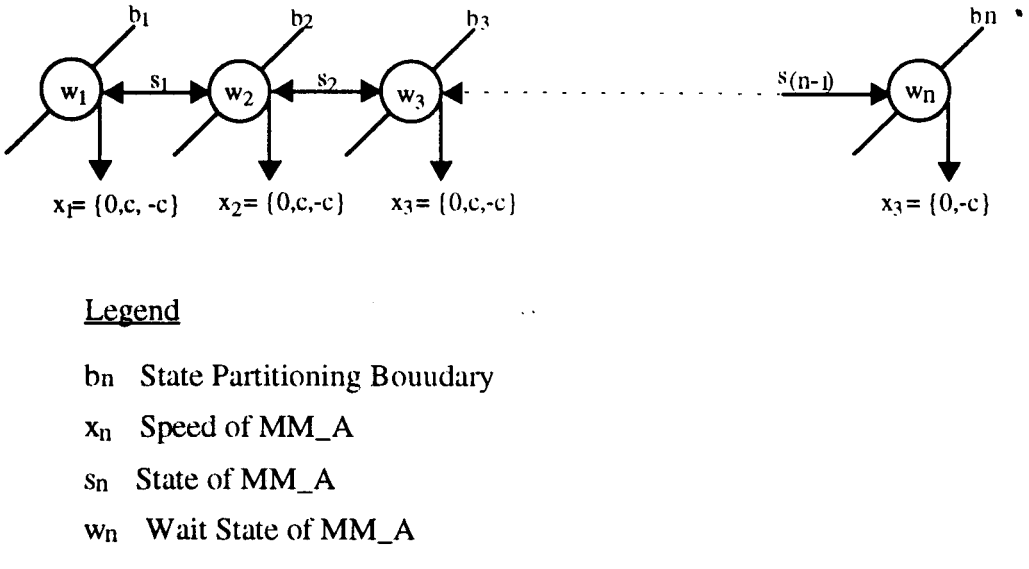


Fig. 15. State Diagram of MM_A for Event-Based Control.

State Diagram of MM_A for Event-Based Control.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

The event-based control model for MM_A is given as

Event-Based MM_A = $\langle X_A, S_A, Y_A, \delta_{intA}, \delta_{extA}, \lambda_A, ta_A \rangle$

where

- X_A = A set of external inputs; the same as before
- $Y_A = A$ set of outputs; the same as before
- $S_A = B_A \times X_A$
- $\delta_{intA}(b_{a1}, x_a) = (b_{a2}, x_a)$; where b_{a2} is the next boundary crossing
- $\delta_{extA}((b_{a1}, x_{a1}), 0, x_{a2}) = (b_{a1}, x_{a2})$
- $\lambda_A(b_{a1}, x_{a1})$ = output function
- $ta_A(c) = (t_2 - t_1)$ with $t_2 > t_1$ and $t_2, t_1 \in R^+_{0,\infty}$
- $ta_A(w_n) = (w_{t2} - w_{t1})$ where
- $w_{t2} = \max\{ta_A(w_n)\}$ and $w_{t1} = \text{Min}\{ta_A(w_n)\}$

The event-based control models for MM_B, CM1, MM_C and MM_B are similar to MM_A.

7.4.2 Simulation of event-based controller of drilling system. The event-based control models for drilling system are simulated using Simscript to generate control/actuation signals for MM_A, MM_B, CM1, MM_C and MM_D. These control signals activate the manipulation modules to move to the next partitioning block. It is assumed that during the normal operation:

- MM_A takes 1 unit of time to move the plate by a distance of (plate_height/($Y_{max}+1$)). The variable Y_{max} is the maximum of the y-coordinates of the locations of all the holes to be drilled.
- MM_B takes 1 unit of time to move the plate by a distance of (plate_width/($X_{max}+1$)). The variable X_{max} is the maximum of the x-coordinates of the locations of all the holes to be drilled.
- MM_C takes 1.5 unit of time to locate the drill bit on the plate while MM_D takes 1.5 unit of time to drill a hole.

The time windows in this example are assumed to have a width of ±10% of the ideal time required to cross the partitioning blocks. In case a sensor response is not received within its time window, the whole system operation is halted and a diagnostic message telling which manipulation module has caused the problem is issued. This is shown by following three simulation results in which control signals for the first three holes are produced.

A successful operation of the system is illustrated in Diagram C1 (Appendix C) in which the controller receives all the sensor responses within time windows and hence issues actuation signals to manipulation modules to complete the task.

Diagram C2 shows the case when MM_A does not work properly and the sensor response is received by the controller before the time window. The controller then issues the error message and stops generating of actuation signals to manipulation modules for the rest of the job. In Diagram C3 another situation is indicated in which CM1 malfunctions and the sensor response is received by the controller after the time window. Finally, Diagram C4 (Appendix C) shows the relationship of the control signals of MM_C and MM_D with the control signal of their coupled model, CM1.

8. CONCLUSION AND FUTURE WORK

According to definition, a DME is a concurrent and modular system in which manipulation modules operate asynchronously to carry out the required task. In order to develop a systematic method to analyse and design a DME and ultimately to control its operation a mathematical model defining the behaviour of DME is required. The DME behaviour is in essence discrete and cannot be modelled using conventional continuous-time methods.

The discrete event modelling methodology employed in this work to model DME matches closely the characteristics of this system particularly reflecting its concurrent and hierarchical structure and discrete nature. The modelling procedure developed is generic and provides a unified framework for static and dynamic behaviour analysis of any type of DME system. This methodology describes the system under study in terms of system states and events; representing the states of manipulation modules at different time intervals and the transitions from one state to another as the result of events.

The developed methodology systematically models the whole DME system according to its physical and logical components considering both atomic and coupled levels. The atomic model describes the behaviour of an individual manipulation module. The coupled model, on the other hand, defines the interaction of an ensemble of manipulation modules operating together to perform a specific task.

The main focus in the discrete event simulation of DME was to develop a generic methodology to systematically define the simulation modules for DME based on the Discrete Event model developed for it.

The simulation model consists of an integrated network of concurrent, communicating and asynchronous processes representing the behaviour of the manipulation modules as defined by their discrete models. The simulation model also addresses the computational and communicational aspects of the manipulation modules. The functions and parameters defined in the simulation model are identical to the discrete event mathematical model of the system.

The simulation model also has exactly the same structure as the mathematical model of the system. It is also decomposed into atomic and coupled simulators representing the behaviour of the atomic and the coupled models respectively. A coupled simulator, likewise, consists of a number of component simulators that may be atomic in nature or again a coupled simulator of some other component simulators. Both the atomic and the coupled simulators are implemented as processes in the simulation algorithm.

The methodology developed to build simulation models is again generic and can be applied to DME systems of any type or configuration. This is clearly evident from the simulation model of the case study as all the prismatic manipulation modules use the same code defining the behaviour of a typical prismatic manipulation module. This is also true for the revolute

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

modules. The modification and expansion of an existing DME system will thus be more efficient. It will also make the design task easier and remove a great load from the shoulders of the system designer.

The development of the event-based controller for DME was an important aspect of this work. This type of control is one level above conventional control approach which enhances the operation of the individual actuators in DME. The major role of this layer of control is to provide a systematic method for sequencing and scheduling the operation of the actuators.

The event-based controller works in an expectation-driven manner. It receives information related to commands and expected response time and windows from the system model. The controller then issues commands to the system under control to move it from one state to another as long as it receives the proper response signals.

An event-based controller primarily contains concurrent, self contained and loosely coupled control models that are derived from the discrete event model of the system. The controller itself can then be expressed as a discrete event model which may be tested, validated and finely tuned through computer simulation prior to its real-time implementation.

The error messages generated by the event-based controller in the events of malfunctioning of one of the components provide important information for diagnostic purposes. It will greatly reduce the time and efforts to bring the system back to the normal operating mode.

The mathematical modelling developed for DME in this work will pave the way for a more systematic work on this concept. The work conducted in this project can be considered as a preliminary nature towards this end.

The models developed in this work were validated through computer simulation and the results were found very encouraging. The real-time validation of the developed algorithms and logic was not possible at the moment due to unavailability of a system appropriate for such experimentation.

The discrete models developed in this work were produced manually. This task is quite time consuming and cumbersome particularly as the size of the system increases. In addition, a great deal of attention and efforts is required to avoid errors. This situation becomes even worse for event-based control models.

A computer-based tool generating discrete models for the system and its control from a given schematic diagram of the system can speedup the process significantly. The simulation models may also be generated in this process and simulated on the computer.

Another possible direction of the work can be to implement the discrete event simulation models and event-based controller on a parallel computing platform such as transputer. The inherent parallelism and distributed nature of discrete event models lend themselves well for such an approach. This will speed up the simulation and control processes and simplifies the code development.

References

1. F. Naghdy and P. Strickland, ''Distributed Manipulation Environment'' *Int. J. Comp. Integ. Manufacturing* **2,** No. 5, 281–289 (1989).
2. F. Nahdy and J. Touminen, ''A more generic approach to robotics'' *Proc. Intl. Conf. on Robotics for Competitive Industries* (1993) pp. 405–412.
3. M. Akian, G. Cohen, S. Gaubert, R. Nikoukhah and J.P. Quadrant, ''Linear systems in (Max, +) algebra'' *Proc. of 29th Conf. on Decision and Control,* Honolulu (1990) pp. 151–156.
4. G. Cohen, D. Dubois, J.P. Quadrat and M. Viot, ''A linear system theoretic view of discrete event processes and its use for performance evaluation in manufacturing'' *IEEE Trans. on Automatic Control* **AC-30,** No. 3, 210–220 (1985).
5. H.A. Taha, *Operations Research; An Introduction* (Macmillan Publishing Comp., USA, 1992).
6. R. Bronson, *Theory and Problems of Operations Research* (McGraw Hill, USA, 1989).
7. B.P. Ziegler, *Theory of Modelling and Simulation* (John Wiley and Sons, USA, 1976).
8. B.P. Ziegler, *Multifaceted Modelling and Discrete Event Simulation* (Academic Press, USA, 1984).
9. M.J. Denham, ''Survey of discrete-event control'' *Proc. of IEE Colloquium on Modelling, Simulation and Control of Discrete Event Systems,* London, UK (1989) pp. 1/1–2.
10. Y.C. Ho, ''Dynamic of discrete event systems'' *Proceedings of the IEEE* (1989) **Vol. 77,** No. 1, pp. 3–6.
11. R. Righter and J.C. Warland, ''Distributed simulation of discrete event systems'' *Proceedings of the IEEE* **77,** No. 1, 99–113 (1989).
12. C.M. Ozerrent and A.S. Willsky, ''Observability of discrete event dynamic systems'' *IEEE Trans. on Automatic Control* **35,** No. 7, 797–806 (1990).
13. G.F. Bryant, ''The use of discrete event simulations in factory wide control'' *IEE Colloquium on Modelling, Simulation and Control of Discrete Event Systems,* London, UK (1989) pp. 2/1–4.
14. E. Gillespie, ''CIM at the Stoneywood Mill of Wiggins Teape'' *Proc. CIM Conf.,* Birmingham, UK (1989) pp. 2/1–4.
15. H.S. Sarjoughian, F.E. Cellier and B.P. Ziegler, ''Hierarchical controllers and diagnostic units for semi-autonomous teleoperation of a fluid handling laboratory'' *Proc. 9th Annual Int. Conf. on Computers and Communication,* Scottsdale, USA (1990) pp. 795–802.
16. R.F. Garzia, M.R. Garzia and B.P. Zielger, ''Discrete event simulation'' *IEEE Spectrum* 32–36 (1986).
17. J. Touminen and F. Naghdy, ''Computations aspects of Distributed Manipulation Environment'' *Proc. 4th Australian Transputer & Occam User Group Conf.* (1991) pp. 167–172.
18. P.J.G. Ramadge and W.M. Wonham, ''The control of discrete event systems'' *Proc. of IEEE* (1989) **Vol. 77,** No. 1, pp. 81–98.
19. A.I. Concepcion and B.P. Ziegler, ''DEVS formalism: a framework for hierarchical model development'' *IEEE Trans. Software Engng* **14,** No. 2, 1228–241 (1988).
20. P.A. Fishwork and B.P. Zielgler, ''Creating qualitative and combined models with discrete events'' *Proc. of 2nd Annual Conf. on AI, Simulation, and Planning in High Autonomy Systems* Gainesville, USA (1991) pp. 306–315.
21. B.P. Ziegler and S. Chi, ''Symbolic discrete event system specification'' *Proc. of 2nd Annual Conf. on AI, Simulation, and Planning in High Autonomy Systems,* Gainesville, USA (1991) pp. 130–141.
22. M.A. Pollastachek, ''Design of a library for simulation'' *Proc. of 5th Israel Conf. on Comp. Systems and Software Engng.,* Herzlia, Israel (1991) pp. 43–49.
23. M. Cheirotti, J.W. Rozenblit and W. Jacak, ''A framework

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

for simulation design of flexible manufacturing systems'' *Proc. of 1991 Winter Simulation Conf.,* Phoenix, USA (1991) pp. 1106.
24. B.P. Ziegler, *Object Oriented Modelling and Discrete Event Simulation* (Academic Press Inc. USA, 1991).
25. B.P. Ziegler, ''DEVS representation of dynamic systems: Event-based intelligent control'' *Proceeding of the IEEE* **77,** No. 1, 72–80 (1989).
26. C.J. Luh and B.P. Zielger, ''Abstracting event-based control models for high autonomy systems'' *IEEE Trans. on Systems, Man and Cybernetics* **23,** No.. 1, 421–54 (1993).
27. E.C. Russell, ''Simscript II.5 and Modsim II: A brief introduction'' *Proc. of Winter Simulation Conference,* Phoenix, USA (1991) 62–66.

APPENDIX A

DEVS model of the drilling system

The model of the drilling system is a coupled model based on MM_A, MM_B and CM1 as defined below:

$\text{Drill} = \langle D, C, \text{Select}\rangle$
$D = \{\text{MM_A, MM_B, } CM1\}$
$C = \{C_1, C_B, C_{CM1}\}$
$C_A = \{M_A, I_A, Z_{A,CM1}\}$
$M_A = \langle X_A, S_A, Y_A, \delta_{intA}, \delta_{extA}, \lambda_A, ta_A\rangle$
$I_A = \{CM1\}$
$Z_{A,CM1} = S_A \rightarrow X_{CM1}$ where
$X_{CM1} = \{(a, b) \mid a \in X_C, b \in X_D\}$
$C_B = \{M_B, I_B, Z_{B,C}\}$
$M_B = \{X_B, S_B, Y_B, \delta_{intB}, \delta_{extB}, \lambda_B, ta_B\rangle$
$I_B = \{CM1\}$
$Z_{B,C} = S_B \rightarrow X_{CM1}$
$C_{CM1} = \{M_{CM1}, I_{CM1}, Z_{CM1,A}, Z_{CM1,B}\}$
$M_{CM1} = \langle D1, C1, \text{Select}\rangle$ where
$D1 = \{\text{MM_C, MM_D}\}$
$C1 = \{C_C, C_D\}$
$C_C = \{M_C, I_C, Z_{C,D}\}$
$M_C = \langle X_C, S_C, Y_C, \delta_{intC}, \delta_{extC}, \lambda_C, ta_C\rangle$
$I_C = \{\text{MM_D}\}$
$Z_{C,D} = S_C \rightarrow X_D$
$C_D = \{M_D, I_D\}$
$M_D = \{X_D, S_D, Y_D, \delta_{intD}, \delta_{extD}, \lambda_D, ta_D\rangle$
$I_D = \{\ \}$
$I_{CM1} = \{\text{MM_A, MM_B}\}$
$Z_{CM1,A} = S_{CM1} \rightarrow X_A$
$Z_{CM1,B} = S_{CM1} \rightarrow X_B$ where
$S_{CM1} = \{(a, b) \mid a \in S_C, b \in S_D\}$

The individual atomic models of this drill example are given as

$M_A = \langle X_A, S_A, Y_A, \delta_{intA}, \delta_{extA}, \lambda_A, ta_A\rangle$
$X_A = \{v_1, v_2, \ldots, v_8\}$; speed set of MM_A such that $v_i \in \{0, +\alpha, -\alpha\}$ where α is a constant so that MM_A moves with a uniform speed.
$S_A = \{(q_a, v_i))$ where $q_a = \{a_1, a_2, \ldots, a_8\}$ (Figure 11)
$Y_A = \{a_1, a_2, \ldots, a_8\}$ (reaching at the desired position)
$\delta_{intA}(q_a, v_i) = (q'_a, v_i)$ where q'_a is the next state of MM_A
$\delta_{extA}\{(q_a, v_i), e, v_j) = (q_a, v_j)$ $(i, j \leq 8)$
$\lambda_A = \{a_1, a_2, \ldots, a_8\}$ (produces output)
$ta_A(s_i) =$ the time advance function

Manipulation modules MM_B, MM_C and MM_D have similar atomic models with the difference

$X_D = \{\text{rotation} \mid \text{rotation} \in \{0, \beta\}\}$ where β is a constant so that MM_D rotates with a uniform speed
$S_B = \{(q_b, v_i)\}$ where $q_b = \{b_1, b_2, \ldots, b_7\}$ (Figure 3.4)
$S_C = \{(q_c, v_i)\}$ where $q_c = \{\text{home_position, contact_with_plate}\}$ and
$S_D = \{(q_d, \text{rotation})\}$ where $q_d = \{\text{start, end}\}$

The last two expressions show that contrary to MM_A and MM_B whose state sets depend on the number and location of holes to be drilled, both MM_C and MM_D have only two states as they always perform a fixed job.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

APPENDIX B

Simulation of the drilling system

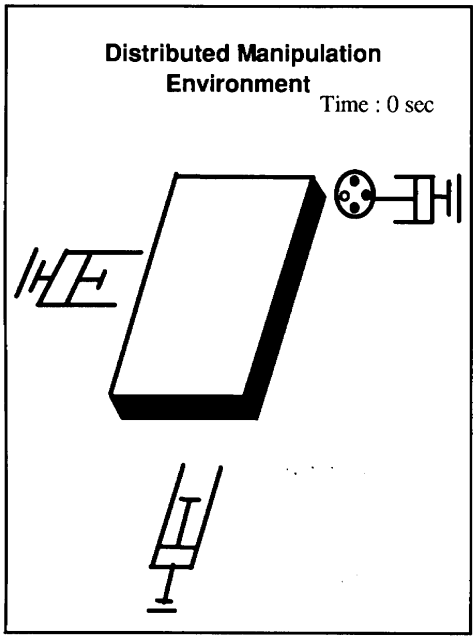


Diagram B1. Initial Set Up.

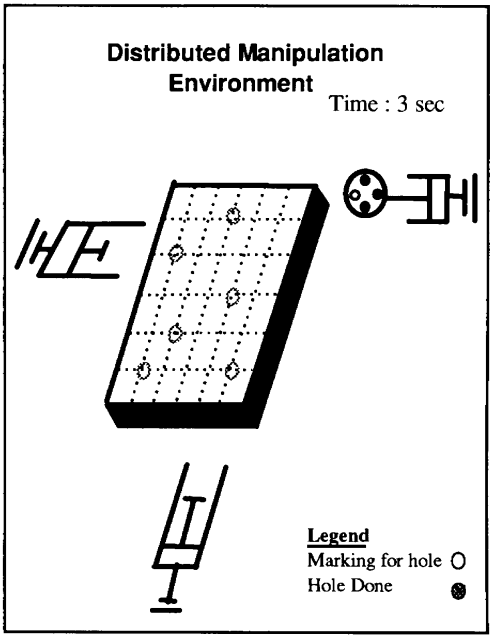


Diagram B2. Marking Done.

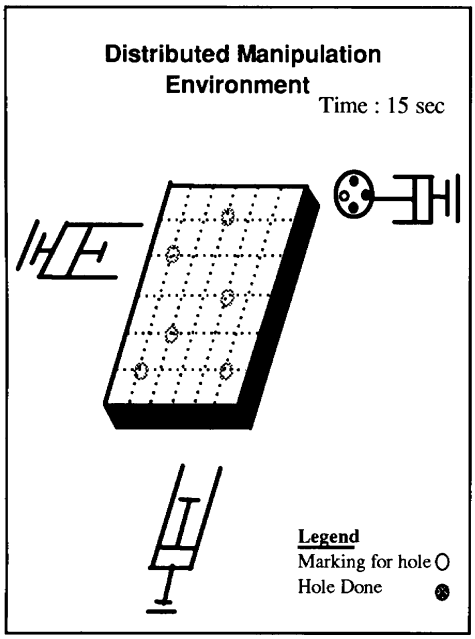


Diagram B3. Main. Modules Before 1st hole.

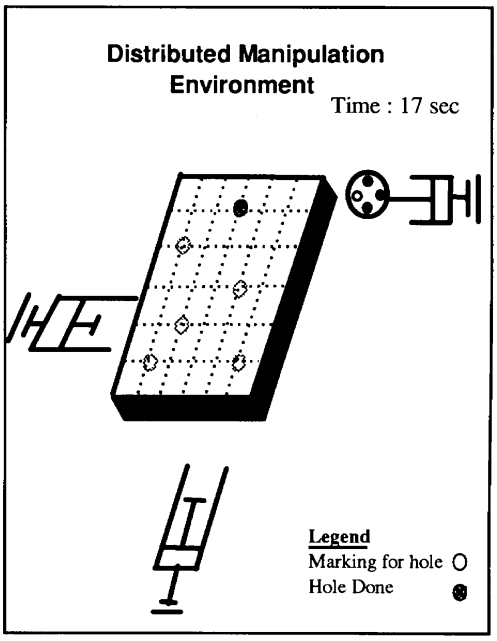


Diagram B4. First Hole Drilled.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

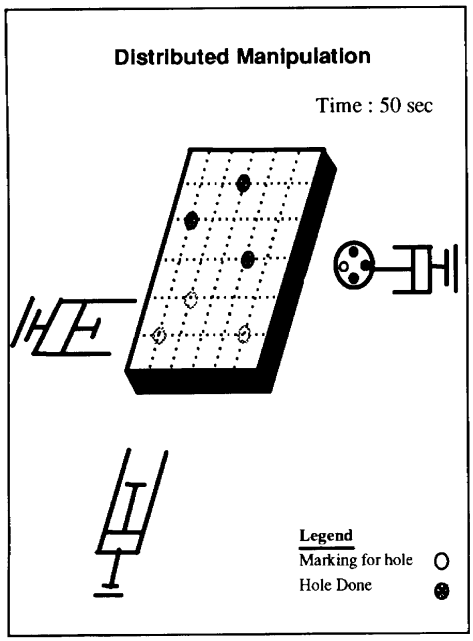


Diagram B5. Three Holes Drilled.

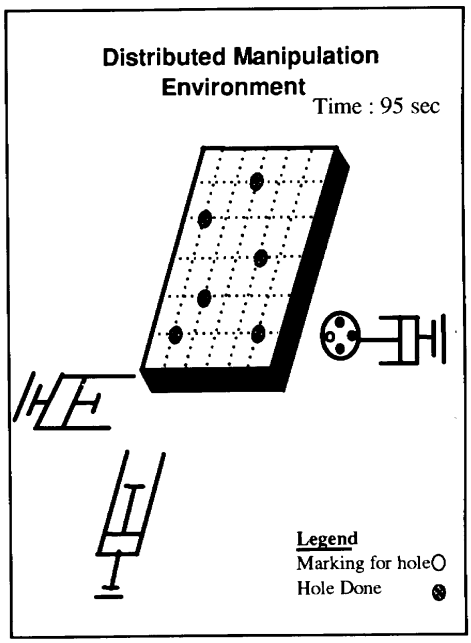


Diagram B6. Job Completed.

APPENDIX C

Event-based control of the drilling system

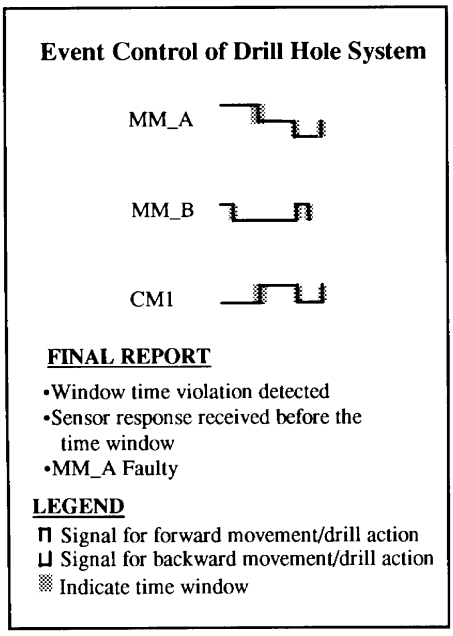


Diagram C1. Control Signals – MM_A Faulty.

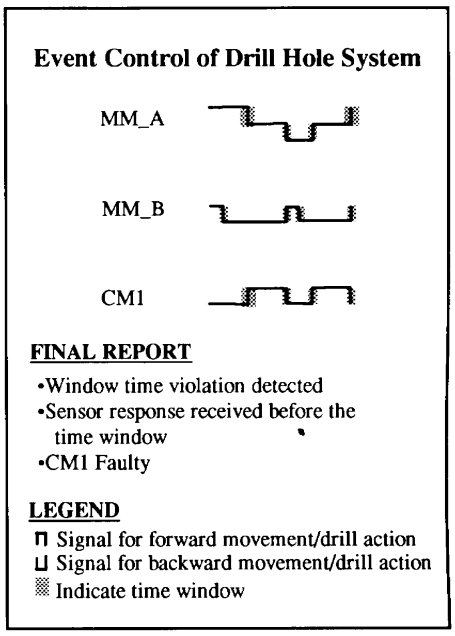


Diagram C2. Control Signals – CM1 Faulty.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press

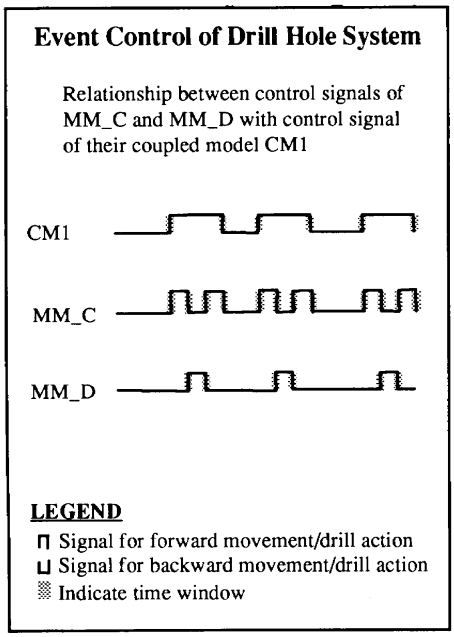


Diagram C3. Relationship Between Control Signals of MM_C, MM_D and CM1.

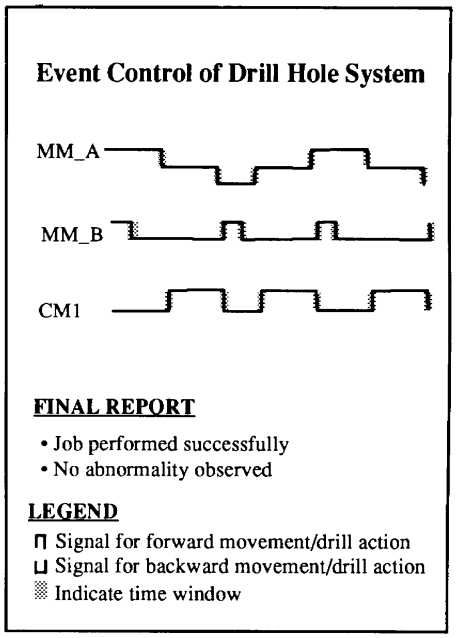


Diagram C4. Control Signals – Normal System Operation.

https://doi.org/10.1017/S0263574797000209 Published online by Cambridge University Press