Elec 4309 Senior Design

Wendell H Chun

Oct. 24, 2017

College of Engineering
and Applied Science
UNIVERSITY OF COLORADO
DENVER | ANSCHUTZ MEDICAL CAMPUS

Information vs. Real-time Systems

Information Systems:

- **Analysed & designed on basis of *static data models*.**
- **Emphasis on *data storage, update and access*.**
- **Timing issues (although important) are *not critical*.**

Real-Time Systems:

- **Analysed & designed with an emphasis on *time related change in conditions and processes*.**
- **Data storage is usually *not encyclopedic* and therefore does not require rigorous modelling.**

Real-time (Embedded) Systems

DEFINITION:

- **Any information processing activity or system which must *respond to externally generated input stimuli within a finite & specified period of time* (Young - 1982)**

HARD REAL-TIME SYSTEMS:

- **Are critically time dependent.**
- **Missed deadlines may be disastrous.**

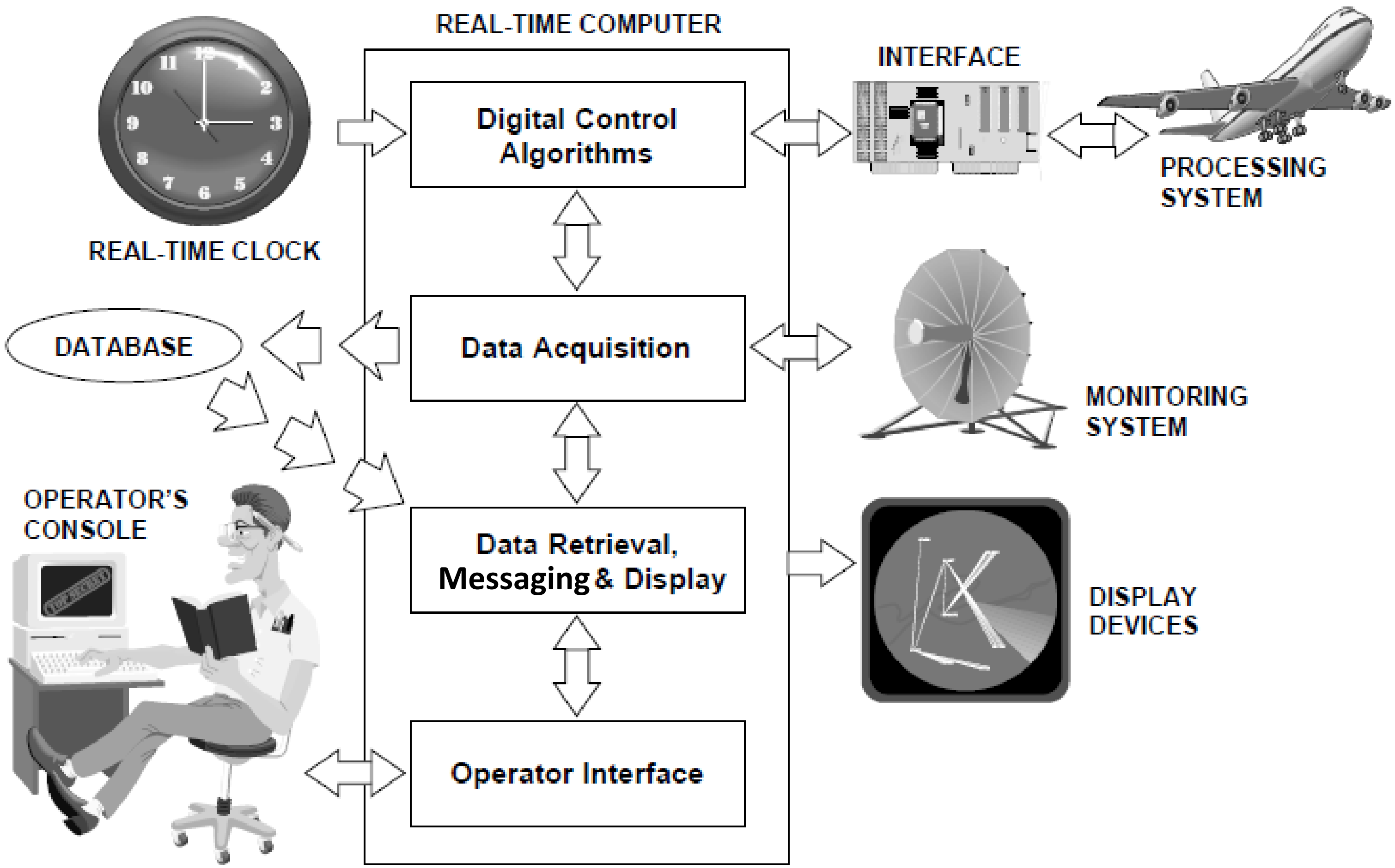
SOFT REAL-TIME SYSTEMS:

- **Are time dependent.**
- **Missed deadlines not usually disastrous.**

Aspect for Consideration

- **Size and Complexity**
  Management, Staffing, Development Techniques, Estimating, Scheduling, Maintenance, Metrics etc.
- **Algorithms**
  Manipulation of real numbers. Mathematical modelling of system components that are under control (eg. PID).
- **Reliability and Safety**
  Exceptional conditions of operation. Risk of failure predictions. Provision of facilities to guarantee reliability.
- **Concurrency**
  Recognition of separate system elements that must work or be controlled in parallel.
- **Time Dependency**
  Separation of those system inputs that cause the generation of certain system outputs within specific time constraints.
- **Interaction with other systems**

S/W Elements of Real-time Systems



The DeMarco Model

Basic Characteristics:-

- **Semi-formal framework in which to construct a set of Requirements Specifications describing what functional processing a System must do.**
- **A logical model that ignores implementation issues.**
    - » Logical models are ideal and thus do not represent the eventual system structure.
- **Assumes perfect technology:**
    - » (Data) input triggered with instantaneous response - *no timing*.
    - » No control issues - *sequentiality or concurrency is not determined.*
    - » No storage limitations.
- **Can be viewed as a large network of primitive (single-purpose) processes communicating via data flows, but is more conveniently represented as an abstracted hierarchy of functional processes.**

DeMarco Model Revisited

- **Assumes ideal technology:**

  Instantaneous responses.

  Steady state operation.

  Time independent.

- **De-emphasises control view:**

  Decisions described at PSpec level only.

  Ignores process sequencing/activation.

  Little support for describing states.

- **Emphasises data view:**

  All processing is on data.

Hatley/Pirbhai Model

- **Hatley/Pirbhai (Requirements) Model =**
  DeMarco (Data Processing) Model + Control Model

- **Control Model is used to specify:**
  - Requirements that exhibit finite state machine behaviour.
  - What behaviour the Data Process Model must exhibit when the system is under the influence of particular external or internal conditions or operating modes.
  - Separate operational modes.
  - High level decisions that affect operational modes.

The Hatley/Pirbhai Model

Requirements Specifications are constructed using:

- **Data Flow Diagrams & Control Flow Diagrams**

  Showing data processes, data inputs and outputs.

  Showing control processing, control inputs and outputs.

- **Requirements Dictionary**

  Containing definitions of data inputs, outputs, stores and intermediate data plus definitions of control inputs, outputs, stores.

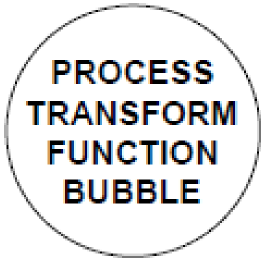
- **Structured Language**

  As for DeMarco Model.

- **Control Specifications**

  FSMs which map control inputs to control outputs &/or show *control* of data processes according to the control inputs.

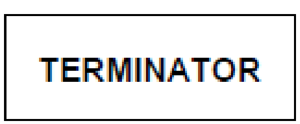
DFD/CFD Elements



Processes should be named with a short action clause summarising *what* is to be done to the *input* (data) in order to produce the *output* (data).

- Dataflows indicate the content and direction of flow of information (or materials) to and from processes, stores and terminators.
- Treat them as pipelines along which single or groups of data/material items of known content and nature can flow.
- Their names reflect their content - nouns or adjectives.
- They *do not* contain or represent dynamic behaviour - no verbal names.

- Stores represent dataflows that are frozen for an indeterminate time.
- The information/materials they represent can be accessed at any time and in any order.
- Nouns and/or adjectives should be used - sometimes plural.

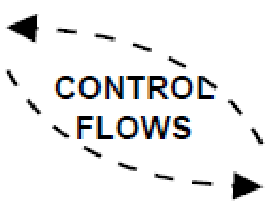


Terminators represent things that are external to the system, but which are important because they provide &/or receive system input and output.

DFD/CFD Elements

- Control Specifications (CSpecs) are used to indicate finite state machine behaviour in the form of:

  STATE TRANSITION DIAGRAMS
  STATE EVENT MATRICES
  DECISION TABLES
  PROCESS ACTIVATION TABLES



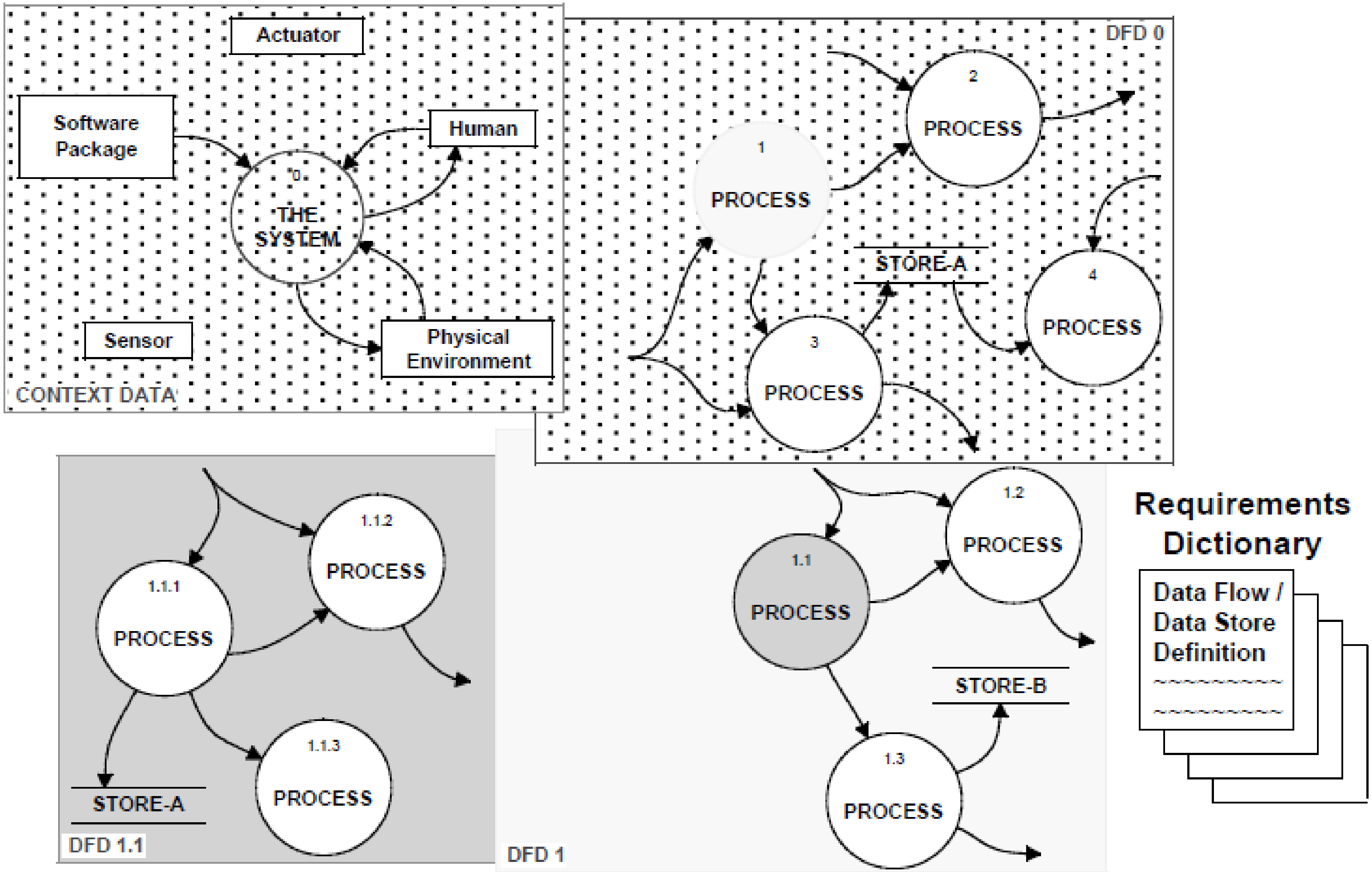
- Control flows indicate the composition and direction of flow of control information to and from CSpecs, control stores and terminators.
- Treat them as pipelines along which single or groups of control items of known composition flow.
- Their names reflect their content - nouns or adjectives.
- They *do not* contain or represent dynamic behaviour - no verbal names.
- Event flows (Ward/Mellor) have similar meaning to control flows but are also used as prompts to enable/disable processes and they do not contain grouped elements.

Control Flow Diagrams

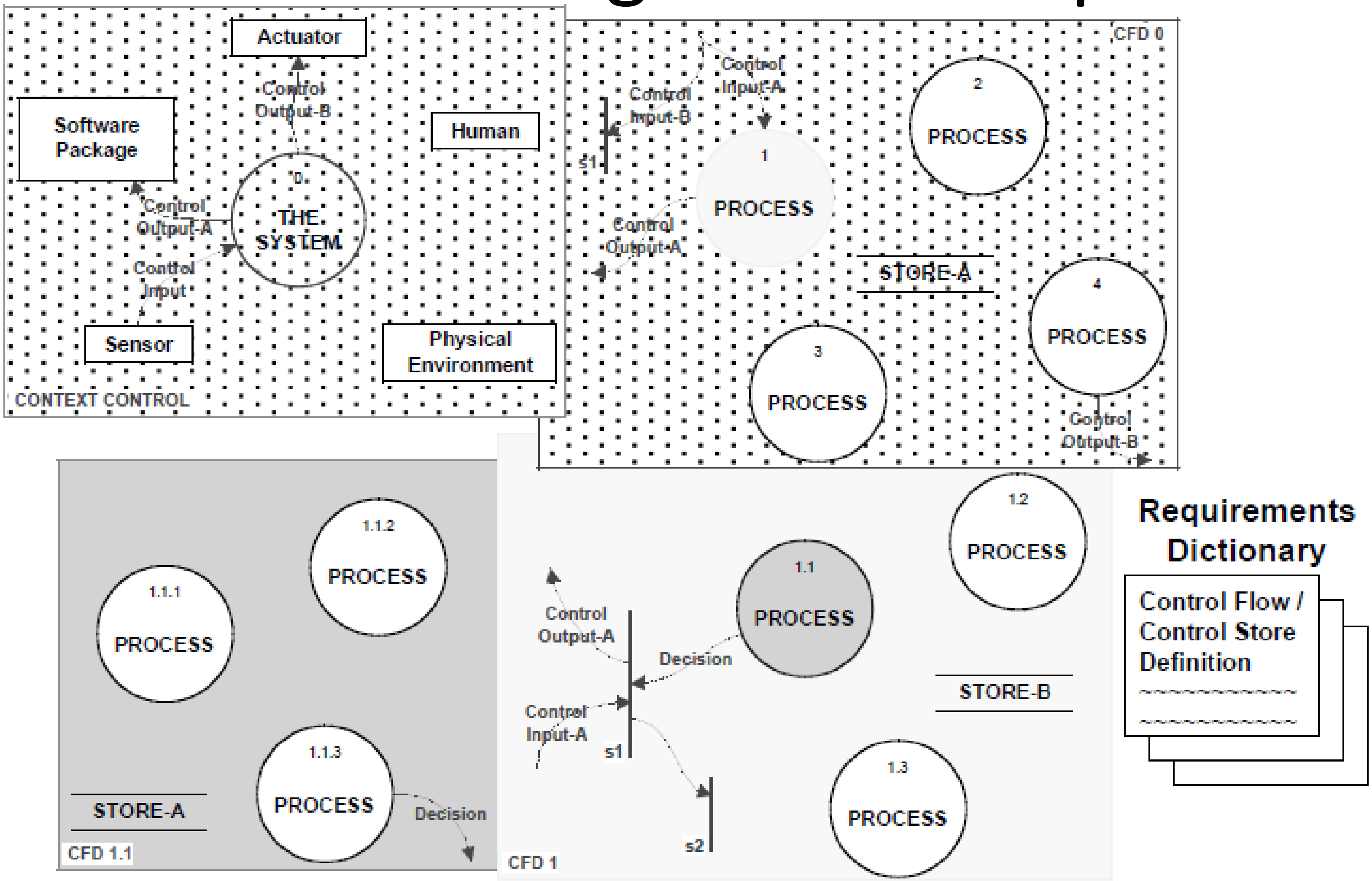
Basic characteristics of CFDs:

- **They are paired with Data Flow Diagrams**
  - Have the same:
    Numbering, Levelling, Balancing, Parent/Child relationships.
- **Show processes (same as DFDs) but not Data Flows.**
  - Only concerned with elements that are affected by control.
  - Data processors are affected by high level control decisions but data flows are not.
- **Show Control Flows and Stores**
- **Show Control Specifications (if any)**

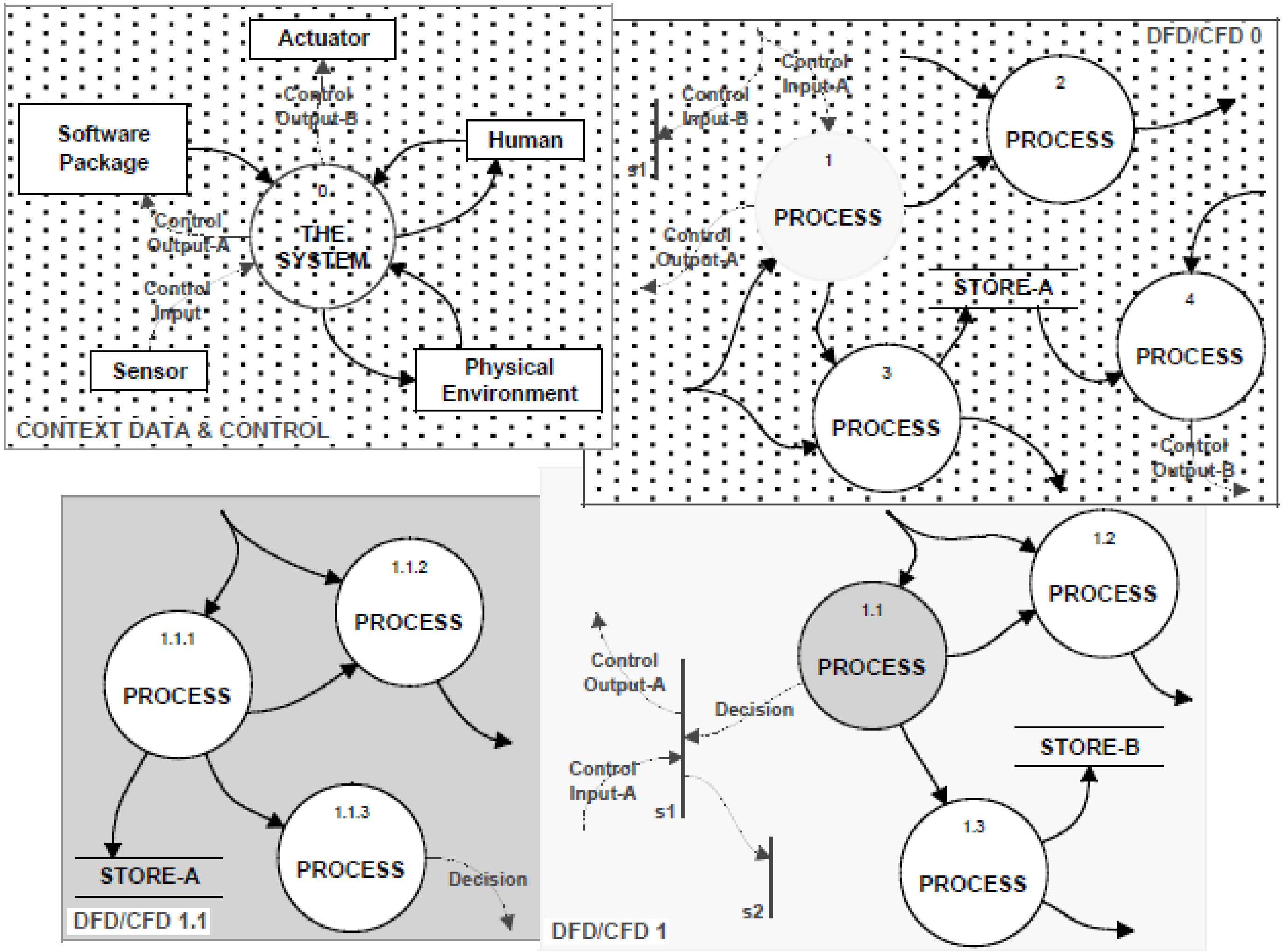
Dataflow Diagram Decomposition



Control Flow Diagram Decomposition



Combined DFD/CFD Decomposition



Control Flows

- Represent pipelines that transport control information of known composition.
- Named according to content - use nouns or adjectives.
- Compositional elements can be primitive or collective.
- Primitive elements always consist of discrete values.

Control Flows

POSSIBLE SOURCES:

- » **External environment**
- » **Control Specifications (CSpecs)**
- » **Process Specifications - resulting from decisions made within PSpec about input data conditions.**

POSSIBLE DESTINATIONS:

- » **External environment**
- » **Control Specifications**
- » ***Not PSpecs*** **- they only transform data inputs.**

Requirements Dictionary

- Contains definitions of both data flows & stores and control flows & stores.
- Notation used for all definitions is that of DeMarco.
- Primitive data flow definitions can be continuous or discrete in nature.
- Primitive control flow definitions must be discrete in nature.

Requirements Dictionary

<table>
<tr><th>Symbol</th><th>Meaning</th></tr>
<tr><td>=</td><td>composed of</td></tr>
<tr><td>This + That</td><td>This together with That</td></tr>
<tr><td>n{ That }m</td><td>n to m iterations of That</td></tr>
<tr><td>[This | That | Another | ...]</td><td>select one of This or That or Another or ...</td></tr>
<tr><td>(This)</td><td>This is optional</td></tr>
<tr><td>“ That “</td><td>literally the word That</td></tr>
<tr><td>* Note about this & that *</td><td>comment field and/or primitive element</td></tr>
<tr><td>This + That + Another + ...</td><td>key attribute of data entity (@ in TeamWork)</td></tr>
<tr><td><That>This</td><td>That version of This (TeamWork only)</td></tr>
</table>

Control Flow Definition Examples

Door_Position = ["Open" | "Closed"]
* The only door positions of interest *

Rate: Event-driven by operator.

Fan_Setting = ["Off" | "Low" | "Medium" | "High"]
* Various fan rotation speed settings *

Rate: As required.

Received = ["True" | "False"]
* Indicator for acknowledgment of receipt of certain signals *

Versions: <Start_Date>, <Start_Time>, <Duration_Time>

Control Flow Definition Examples

Primitive definition:

Windows_Command =
[“Main” | “Accessories” | “Windows_Applications”]

Decomposable definition:

Windows_Command =
[Main | Accessories | Windows_Applications]
Main =
[“File_Manager” | “Control_Panel” | “Print_Manager” | “Clipboard” | “DOS_Prompt” | “Windows_Setup”]
Accessories =
[“Notepad” | “Write” | “Clock” | “Calendar” | “Terminal”]

Control Flow Definition Examples

Grouped definition:

File_Management_Activity =
Main + File_Manager + File_Selection + (File_Operation) + ...

- **Primitive control flow definitions are common with most systems.**
- **Decomposable definitions are less common.**
- **Grouped definitions are relatively uncommon.**

Control Specifications

Absolutes of Real-Time Systems:

- **Behave predictably.**
- **Maintain history of events or system conditions.**
- **Change behaviour (predictably) according to event history (past and present).**

- **These properties imply that real-time systems must deal with a finite number of events and produce a finite number of outcomes in order to behave predictably.**

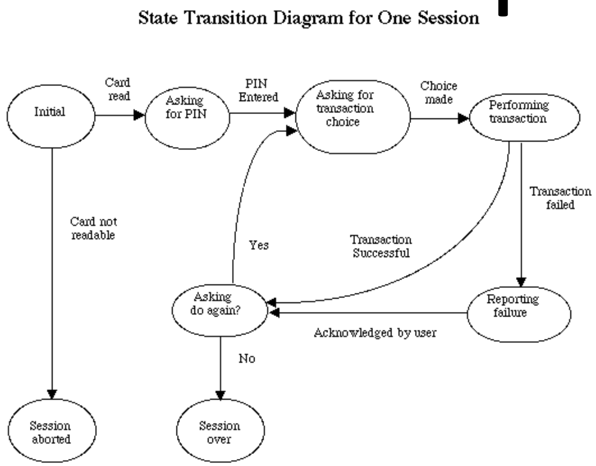
Control Specifications

Represent various finite state machines as follows:

• State Transition Diagrams	STD
• State Event Matrices	SEM
• Decision Tables	DT
• Process Activation Tables	PAT

- **Finite State Machines (FSMs) can be used only when a finite number of inputs having a finite set of values can lead to a finite number of outputs (or set of actions) having a finite set of values.**

Control Specification Example



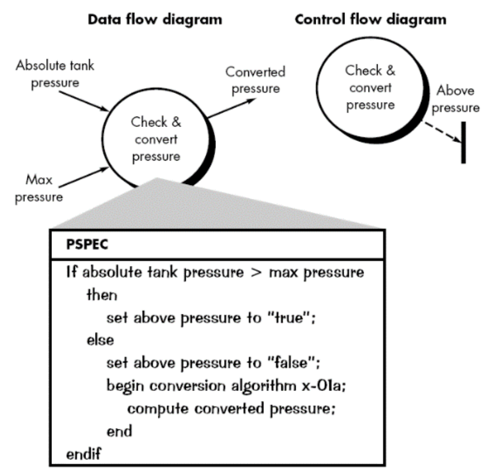
State Transition Diagram (STD)

Conditions	1	2	3	4	5	6
A == 0	Y	Y	N	N	N	N
A > 5			Y	Y	N	N
A == B	Y	N	Y	N	Y	N
Actions	**1**	**2**	**3**	**4**	**5**	**6**
Print "A is 0"	X	X				
Print "A > 5"			X	X		
Print "A == B"	X		X		X	

Decision Tables (DT)

State / Event	Off	Orienting	Turning	Traveling	Avoid: Backing	Avoid: Turning	Avoid: Forward
Button 1 pressed	Orienting	Off	Off	Off	Off	Off	Off
Completed Calculating next heading		Turning					
Pointed towards target heading			Traveling				
Detected obstacle				Avoid: Backing		Avoid: Backing	Avoid: Backing
Traveled to either waypoint or target distance				Orienting			
Last waypoint reached		Off					
Moved backwards preset distance					Avoid: Turning		
Turned preset amount						Avoid: Forward	
Moved forward preset distance							Orienting

State Event Matrices (SEM)



Process Activation Tables (PAT)

College of Engineering and Applied Science
UNIVERSITY OF COLORADO
DENVER | ANSCHUTZ MEDICAL CAMPUS

Control Specifications

CSpecs are best used for systems that exhibit:

- **significant numbers of states or modes of operation.**
- **significant complexity in terms of elaborating decisions that affect or determine system configuration or modes of operation.**
- **significant numbers of inputs (events) that have no effective content and merely serve as triggers/signals for actions that affect operational modes (or states) of parts of the system.**
- **high level management control of large portions of the system.**

Control Specifications

The FSMs within CSpecs are best used to:

- **elaborate system states in terms of externally observable behaviour - with STDs.**
- **specify states in terms of combinations of active and inactive processes - with STDs and PATs.**
- **specify sequences of high level actions that change operational modes - with STDs and PATs**
- **specify results of particular combinations of control input - with DTs and PATs**
- **map combinations of input control signals into an output signal - with DTs.**

Types of FSM Used within CSPECs

Two Categories:

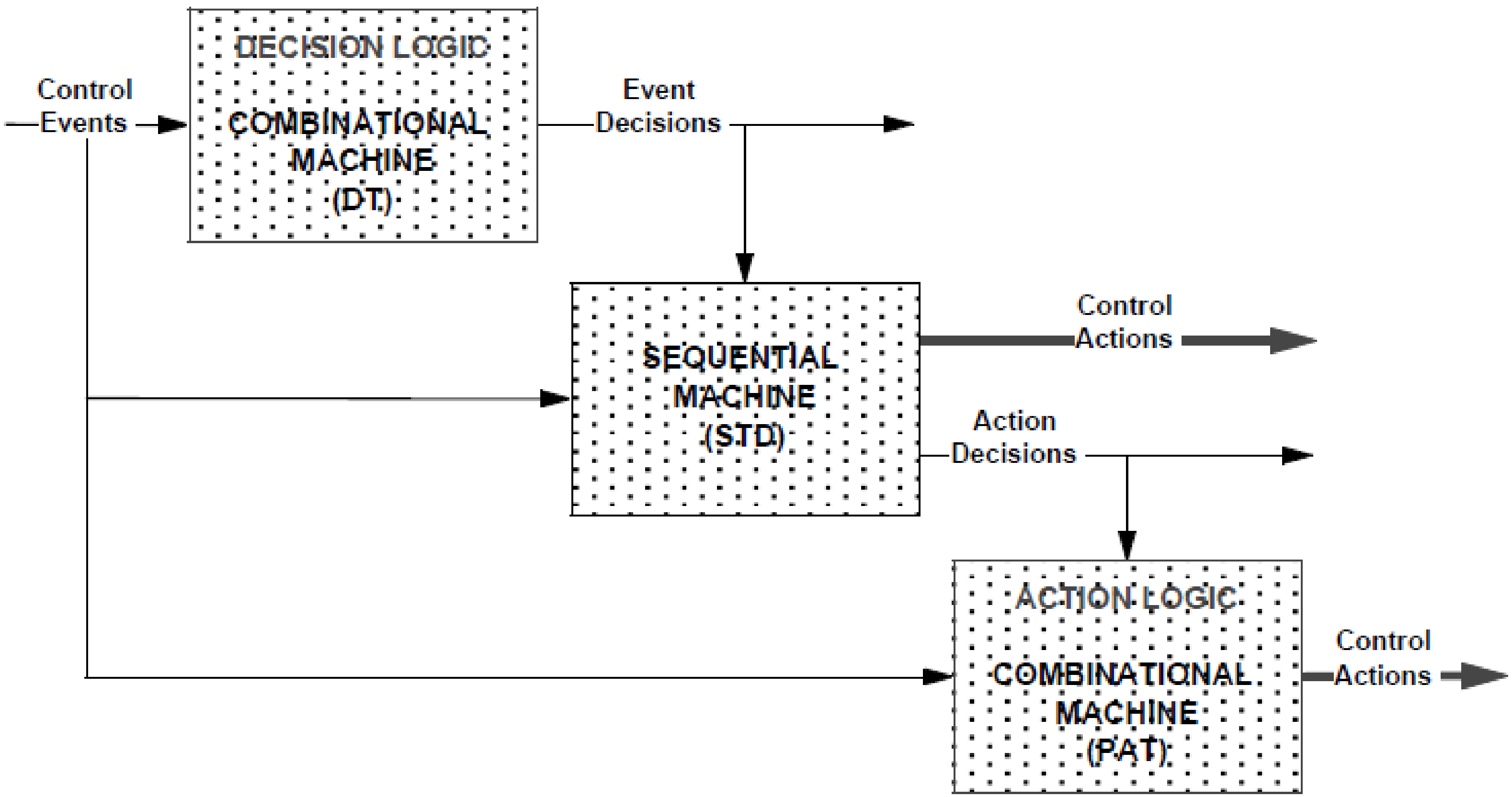
COMBINATIONAL

- **Outputs are uniquely determined by current inputs.**
- **Keeps no information of past events (no memory).**
- **Include Decision Tables & Process Activation Tables.**

SEQUENTIAL

- **Outputs determined using both current and previous inputs.**
- **Maintains information on specific system conditions (has memory).**
- **Include State Transition Diagrams & SEMs.**

Combinations of FSM Types in CSpecs



Hatley/Pirbhai & Sequential FSMs

Basic Characteristics:-

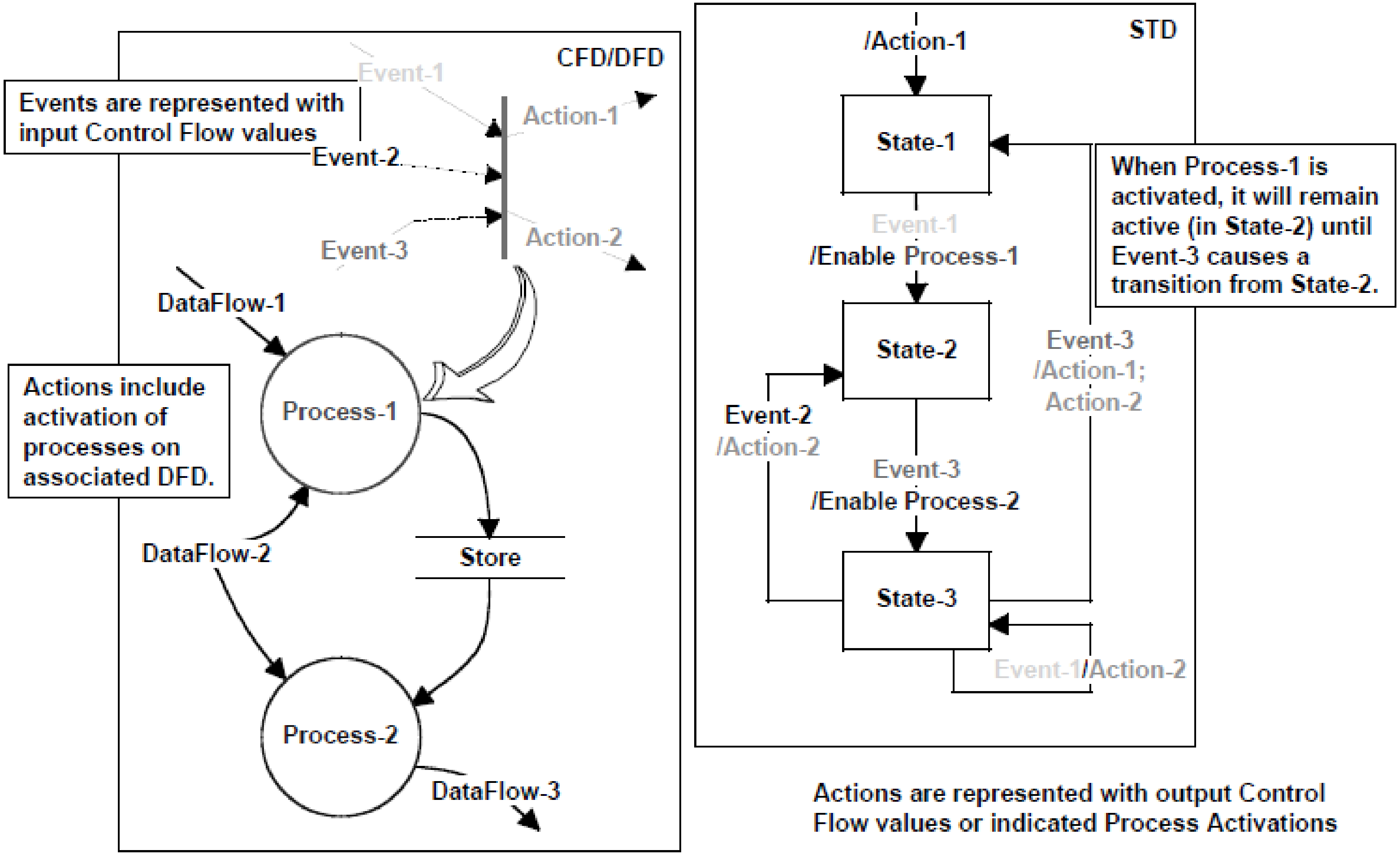
1 Use a hybrid form of Mealy and Moore models in an attempt to take advantage of both models.
   - Basically use Mealy model.
   - Additionally use Moore when it is (more) convenient.

2 Define states as being the mode or condition of system.

3 Output of SFSM includes activation of processes.

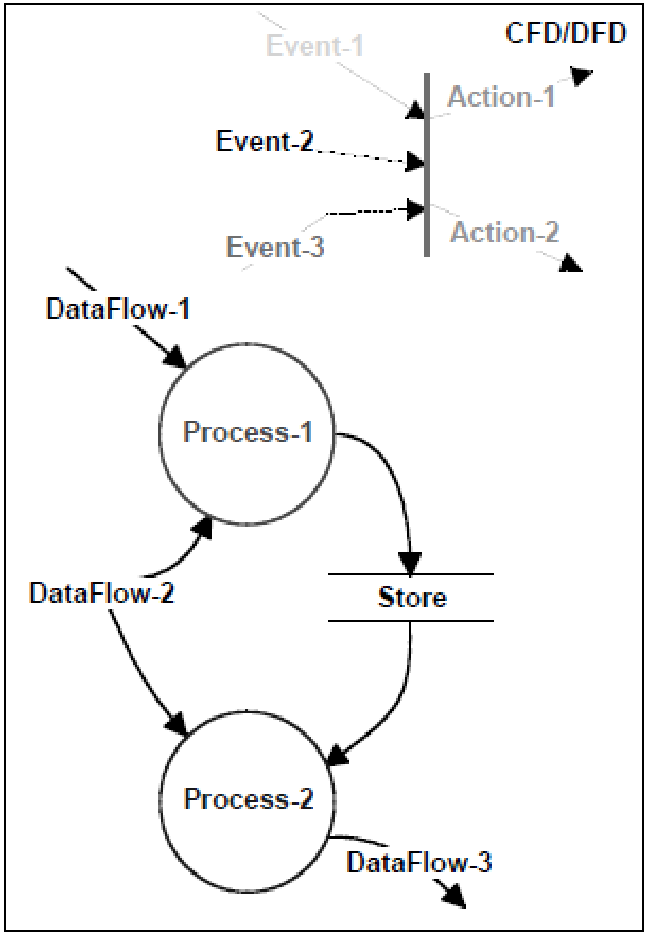
4 Actions associated with a transition continue in effect until the next transition.

Hatley/Pirbhai SFSM Conventions - STD



Actions are represented with output Control Flow values or indicated Process Activations

Hatley/Pirbhai SFSM Conventions - SEM



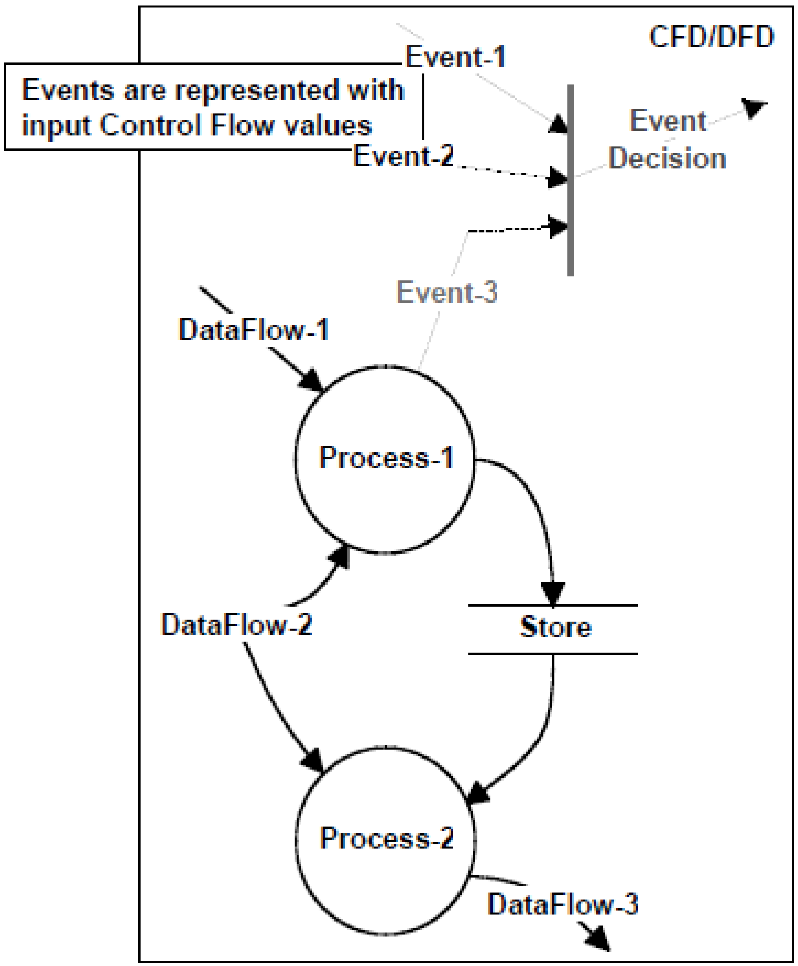
SEM

For Mealy form of SFSM cells of the matrix that correspond to a transition will contain the resulting action together with the name of the destination state.

EVENT / STATE	Event-1	Event-2	Event-3
State-1	Enable P-1 State-2		
State-2		Action Next State	Enable P-2 State-3
State-3	Action-2	Action-2 State-2	A-1 & A-2 State-1

The State Event Matrix is an alternative form for an STD.

Hatley/Pirbhai CFSM Conventions



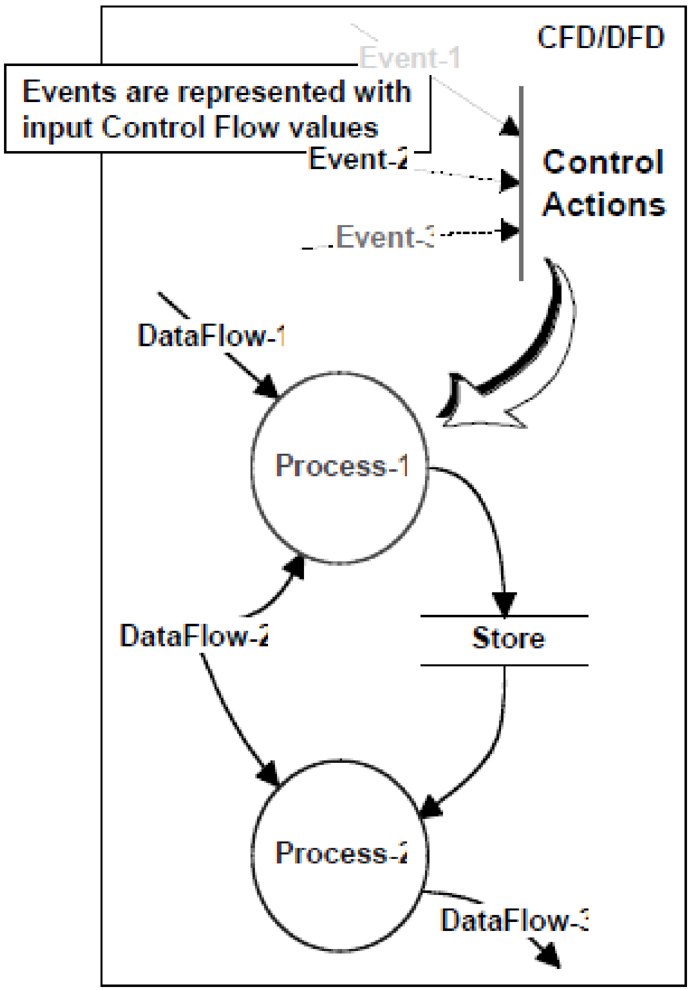
DT

The value of Event Decision is determined by the combination of the input values of Event-1, Event-2 and Event-3.

Event-1	Event-2	Event-3	Event Decision
Discrete Value-1	Discrete Value-2	Discrete Value-3	Decision 1
Discrete Value-2	Discrete value-3	Discrete Value-1	Decision 2
Discrete Value-1	Discrete Value-1	Discrete Value-2	Decision 2

Event decisions are represented with output Control Flow values.

Hatley/Pirbhai CFSM Conventions



PAT

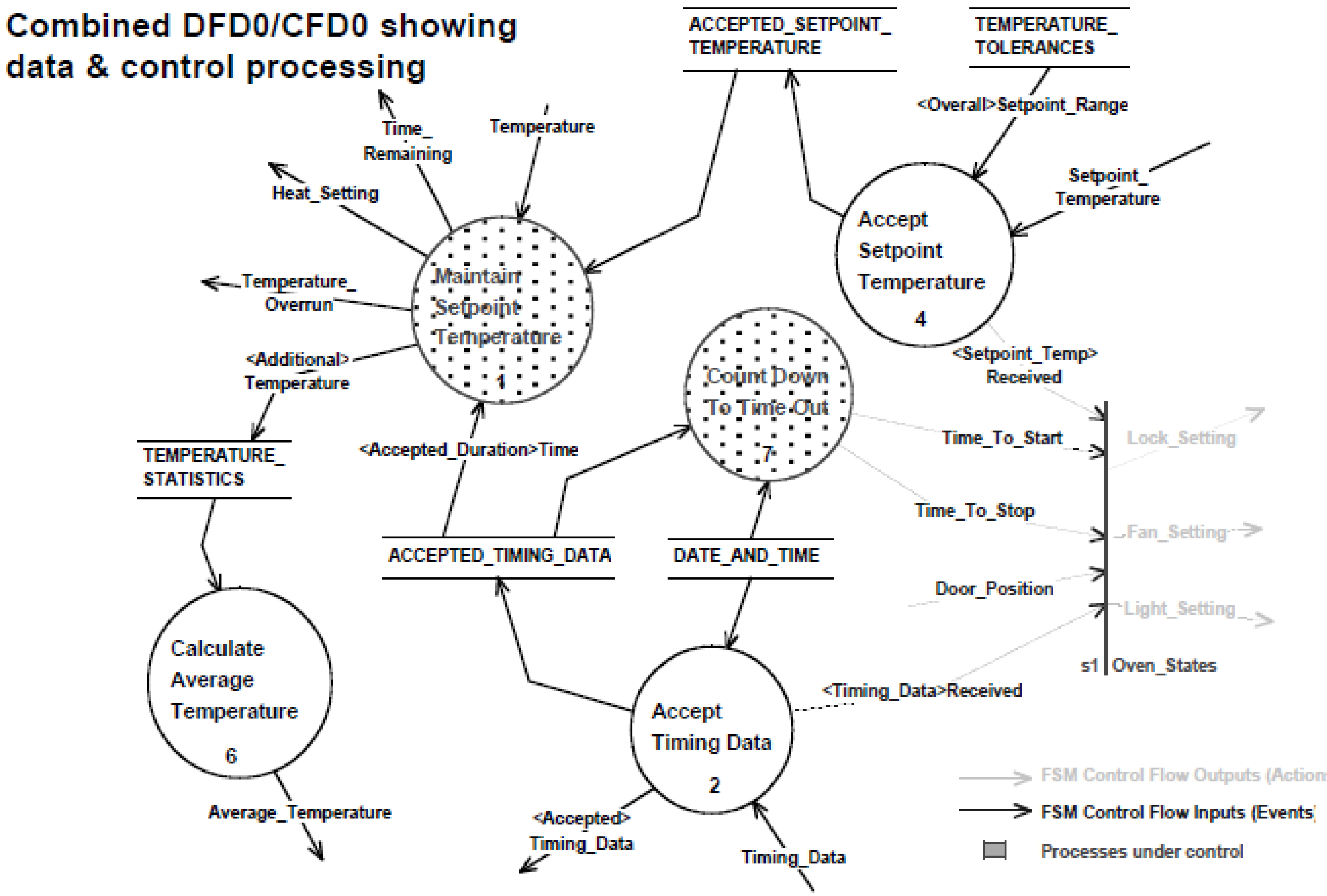
The integer values indicate (de)activation of Process-1 and/or Process-2 and are determined by the combination of the input values of Event-1, Event-2 and Event-3.

Event-1	Event-2	Event-3	Process 1	Process 2
Discrete Value-1	Discrete Value-2	Discrete Value-3	1	0
Discrete Value-2	Discrete value-3	Discrete Value-1	1	2
Discrete Value-1	Discrete Value-1	Discrete Value-2	0	1

Control Actions are represented with indications of process activation in a table.

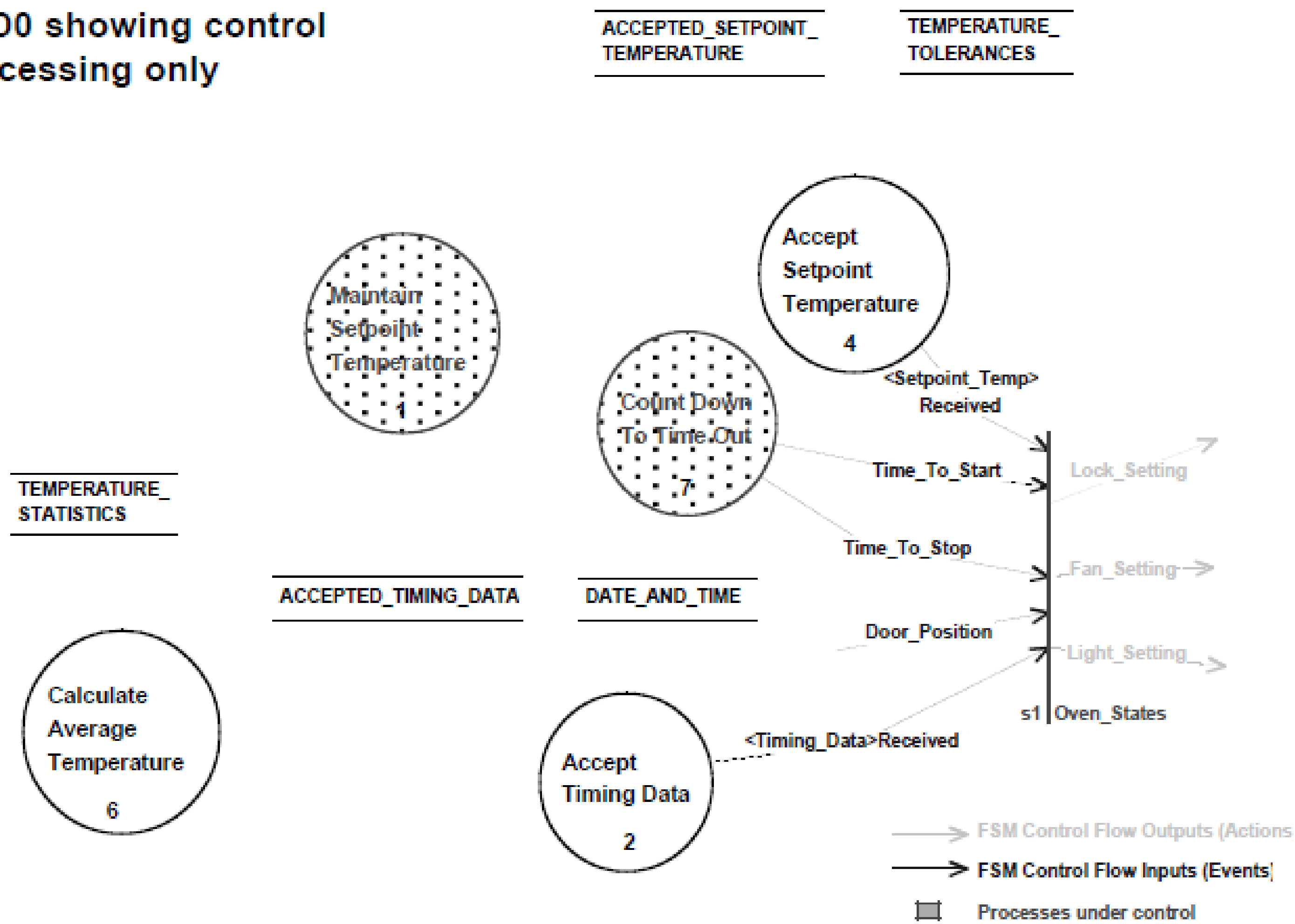
Only those processes being controlled are shown in the PAT

Control Specification as STDs

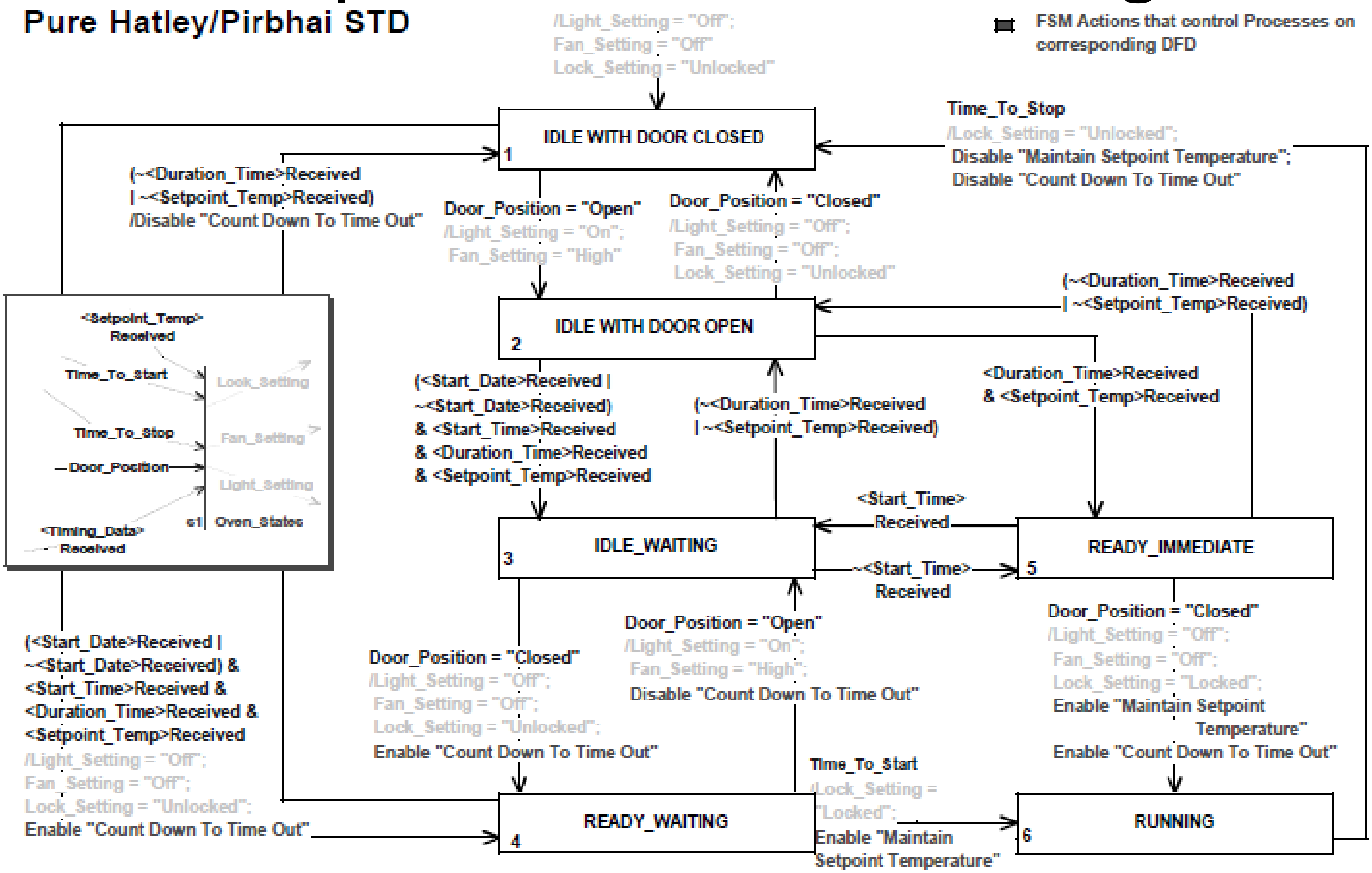


Control Specification as STDs

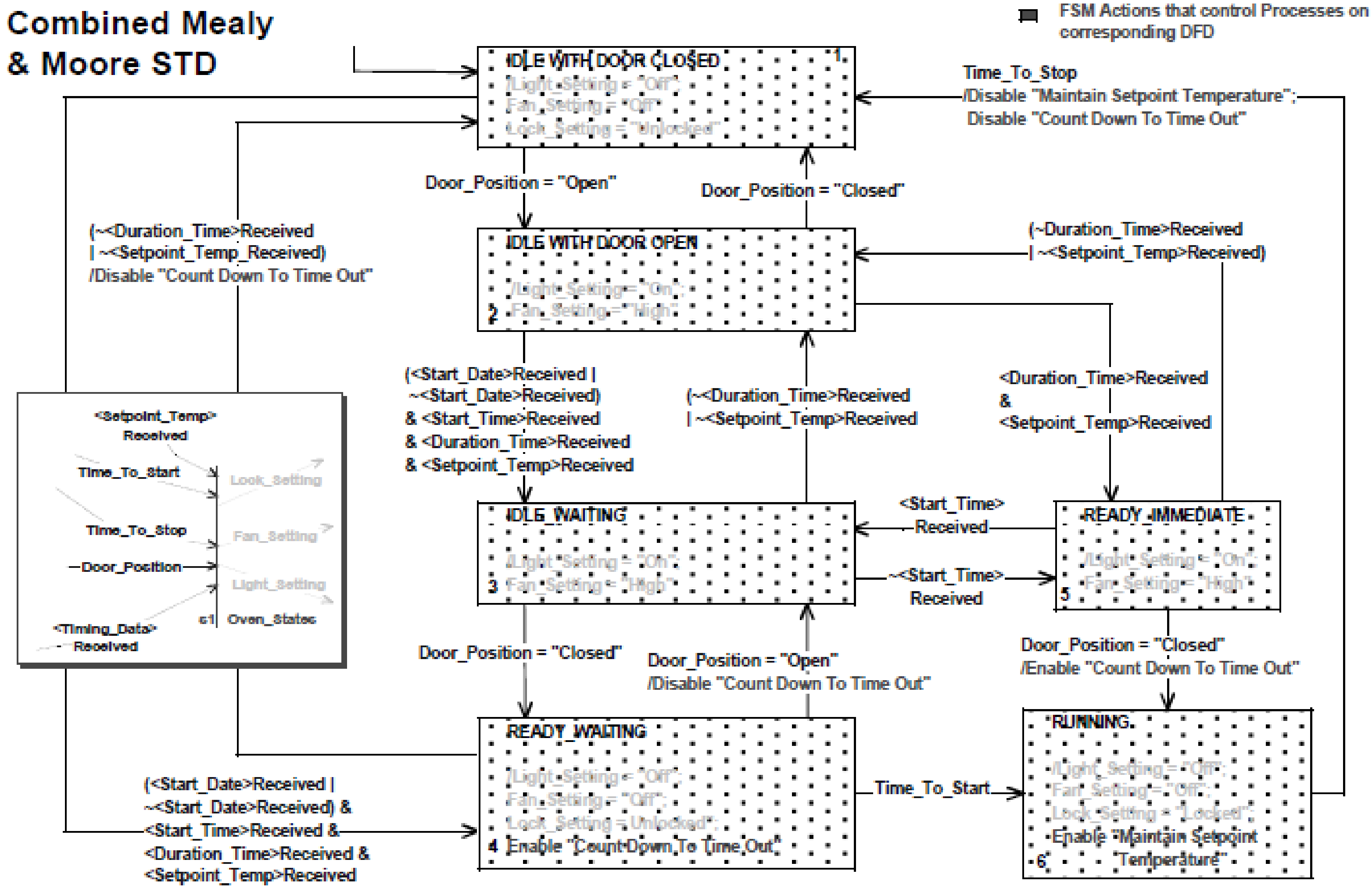
CFD0 showing control processing only



Control Specification Containing STDs



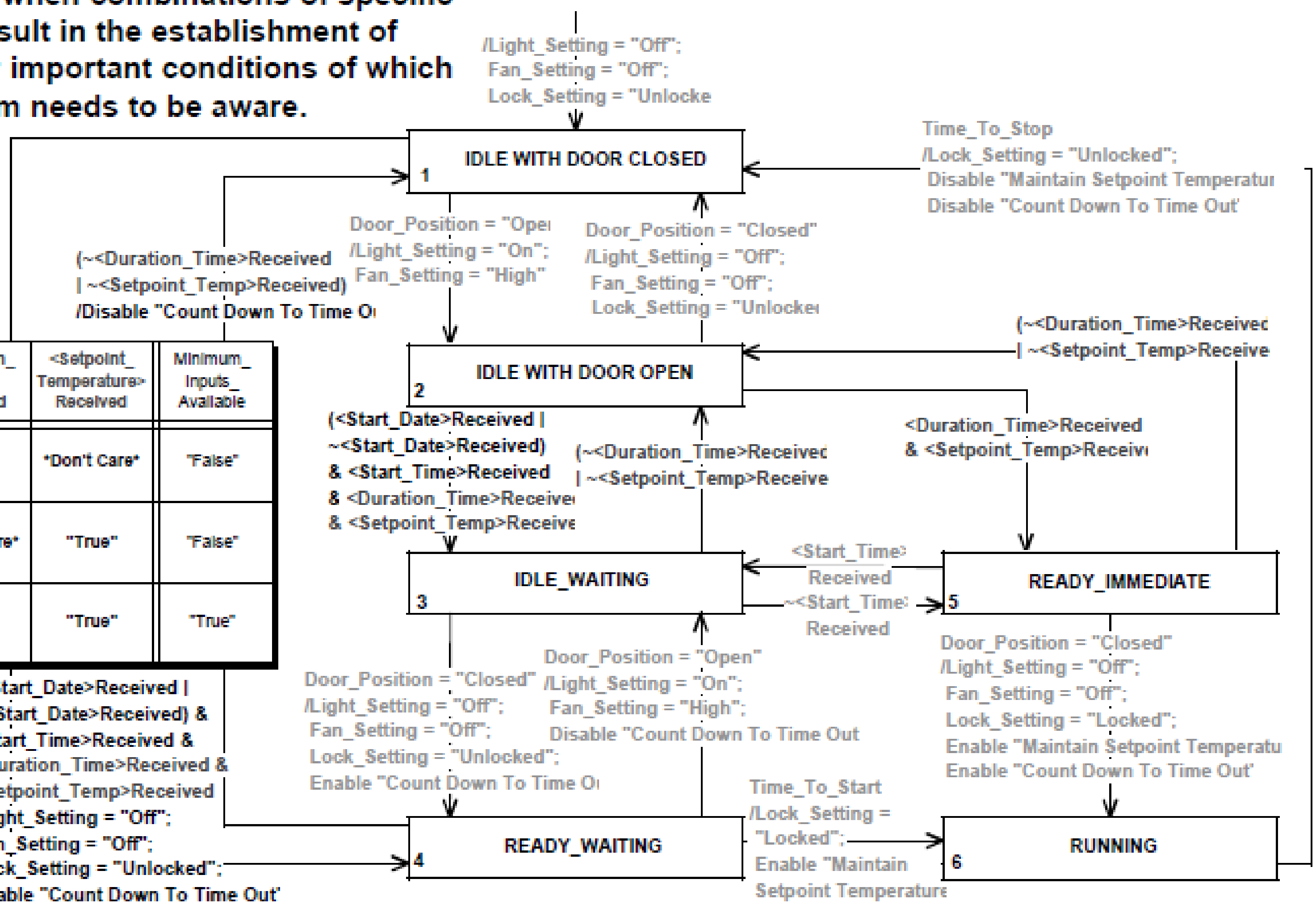
Control Specification Containing STDs



Control Specs Containing DTs and STDs

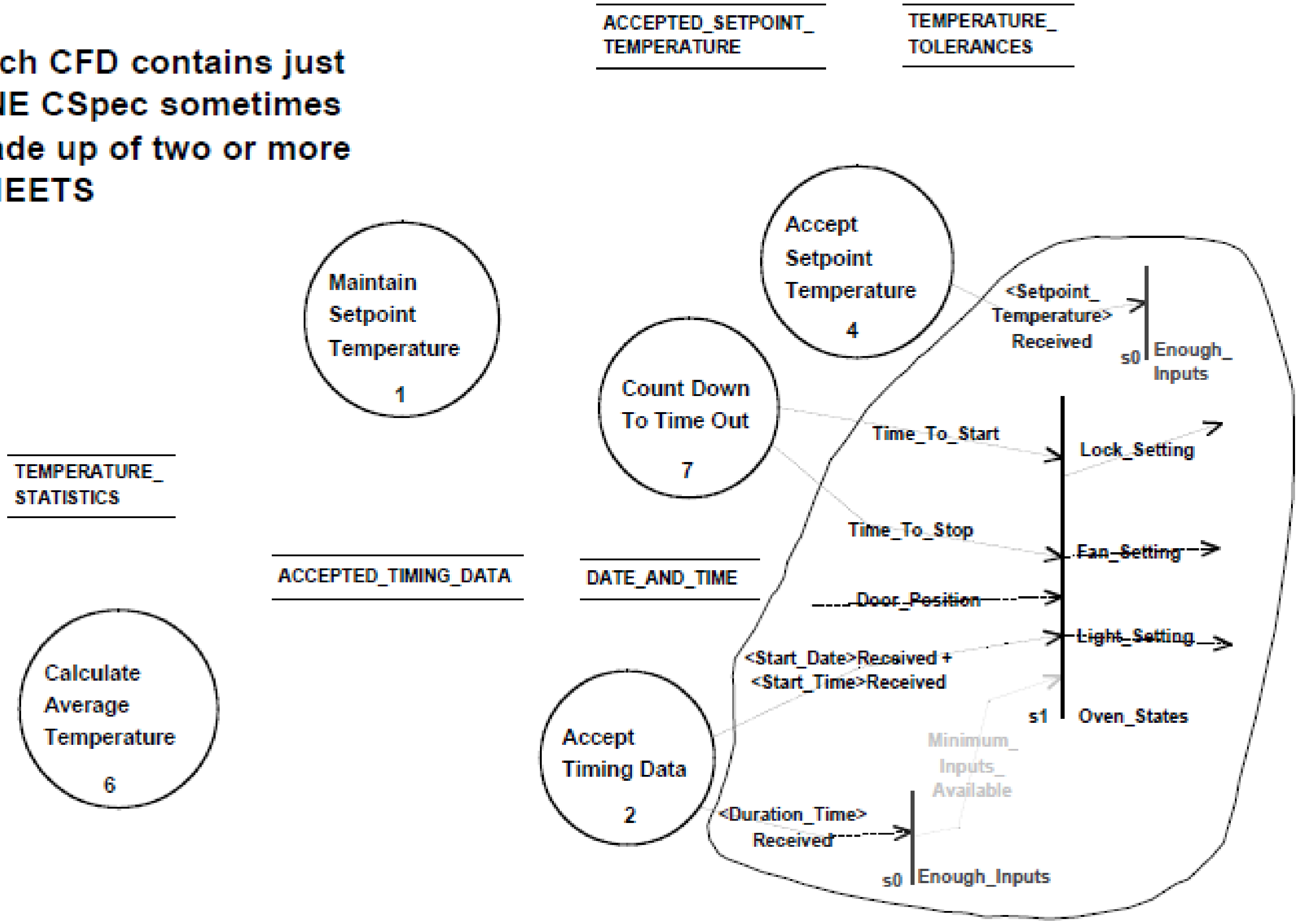
Use a DT when combinations of specific events result in the establishment of particular important conditions of which the system needs to be aware.

<Duration_ Time> Received	<Setpoint_ Temperature> Received	Minimum_ Inputs_ Available
"True"	"Don't Care"	"False"
"Don't Care"	"True"	"False"
"True"	"True"	"True"



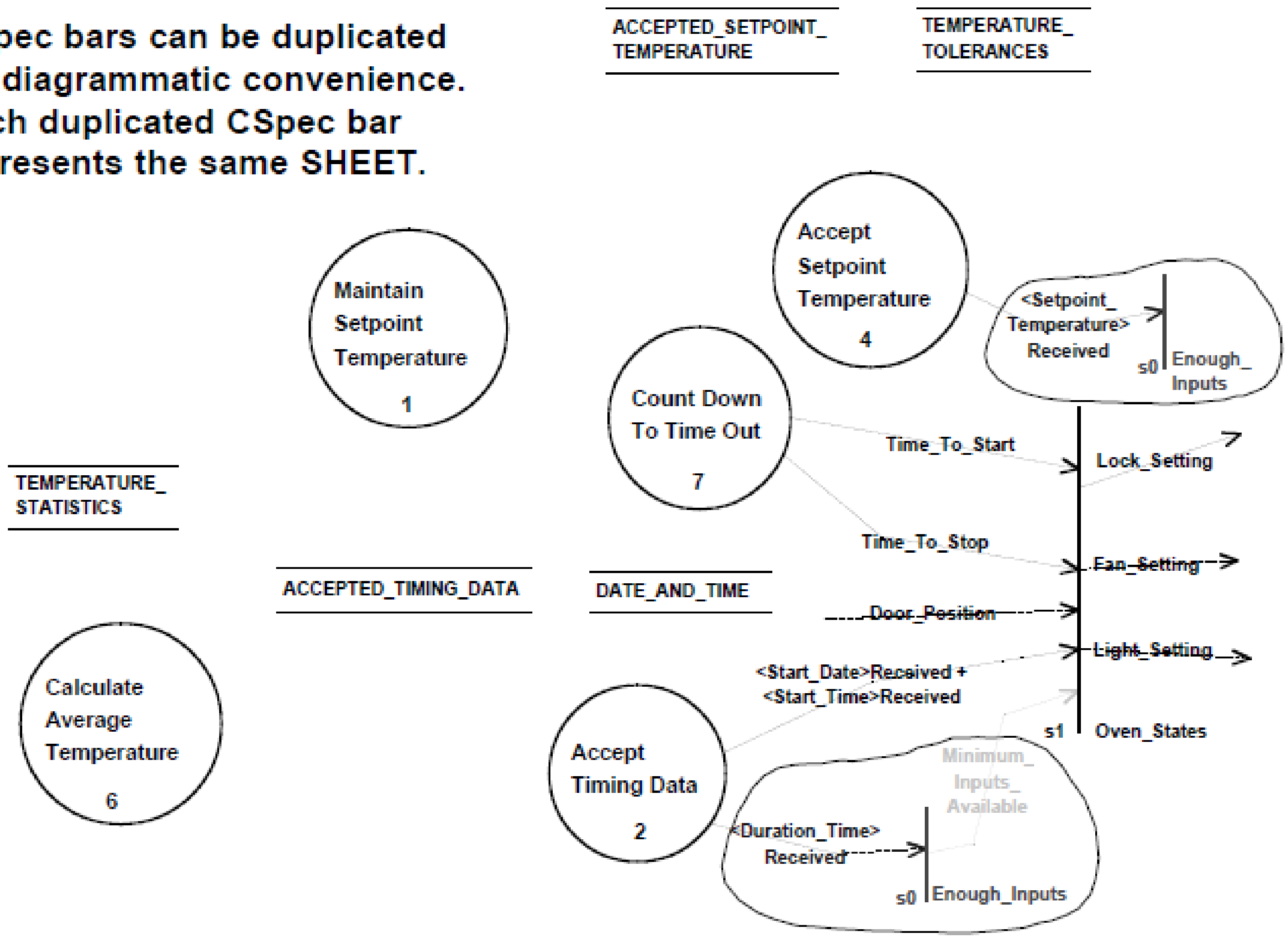
Control Specs Containing DTs and STDs

Each CFD contains just ONE CSpec sometimes made up of two or more SHEETS

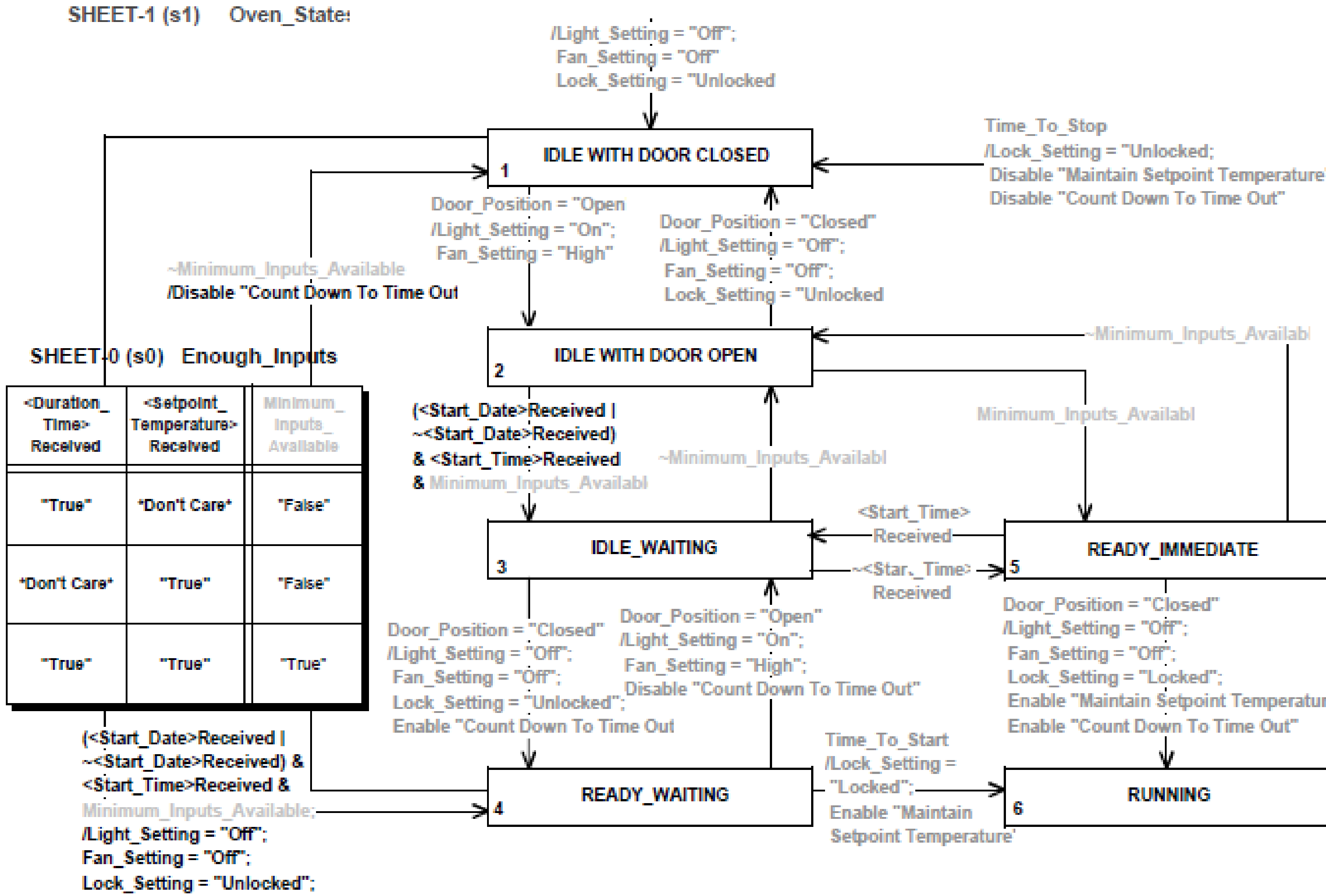


Control Specs Containing DTs and STDs

CSpec bars can be duplicated for diagrammatic convenience. Each duplicated CSpec bar represents the same SHEET.



Control Specs Containing DTs and STDs



<Duration_Time> Received	<Setpoint_Temperature> Received	Minimum_Inputs_Available
"True"	"Don't Care"	"False"
"Don't Care"	"True"	"False"
"True"	"True"	"True"

Control Specs Containing PATs and DTs

CONTROL EVENTS							CONTROL ACTIONS		EVENT DECISIONS		
Door_ Position	<Start_Date> Received	<Start_Time> Received	<Duration_ Time> Received	<Setpoint_ Temperature> Received	Time_To_ Start	Time_To_ Stop	PROCESS 1	PROCESS 7	Light_ Setting	Fan_ Setting	Lock_ Setting
"Open"	"False"	"False"	"False"	"False"	"False"	"False"	0	0	"On"	"High"	"Unlocked"
"Closed"	"False"	"False"	"False"	"False"	"False"	"False"	0	0	"Off"	"Off"	"Unlocked"
"Open"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	0	"On"	"High"	"Unlocked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	1	"Off"	"Off"	"Unlocked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"True"	"False"	1	1	"Off"	"Off"	"Locked"
"Open"	"False"	"False"	"True"	"True"	"False"	"False"	0	0	"On"	"High"	"Unlocked"
"Closed"	"False"	"False"	"True"	"True"	"False"	"False"	1	1	"Off"	"Off"	"Locked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"True"	0	0	"Off"	"Off"	"Unlocked"

Control Specs Containing PATs and DTs

The DT part of a combined PAT and DT.

Door_ Position	<Start_Date> Received	<Start_Time> Received	<Duration_ Time> Received	<Setpoint_ Temperature> Received	Time_To_ Start	Time_To_ Stop	Light_ Setting	Fan_ Setting	Lock_ Setting
"Open"	"False"	"False"	"False"	"False"	"False"	"False"	"On"	"High"	"Unlocked"
"Closed"	"False"	"False"	"False"	"False"	"False"	"False"	"Off"	"Off"	"Unlocked"
"Open"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	"On"	"High"	"Unlocked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	"Off"	"Off"	"Unlocked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"True"	"False"	"Off"	"Off"	"Locked"
"Open"	"False"	"False"	"True"	"True"	"False"	"False"	"On"	"High"	"Unlocked"
"Closed"	"False"	"False"	"True"	"True"	"False"	"False"	"Off"	"Off"	"Locked"
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"True"	"Off"	"Off"	"Unlocked"

DTs should be used whenever complicated decisions about control input combinations need to be made in order to produce possible combinations of control outputs.

Control Specs Containing PATs and DTs

The PAT part of a combined PAT and DT.

Door_ Position	<Start_Date> Received	<Start_Time> Received	<Duration_ Time> Received	<Setpoint_ Temperature> Received	Time_To_ Start	Time_To_ Stop	PROCESS 1	PROCESS 7
"Open"	"False"	"False"	"False"	"False"	"False"	"False"	0	0
"Closed"	"False"	"False"	"False"	"False"	"False"	"False"	0	0
"Open"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	0
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	1
"Closed"	*Don't Care*	"True"	"True"	"True"	"True"	"False"	1	1
"Open"	"False"	"False"	"True"	"True"	"False"	"False"	0	0
"Closed"	"False"	"False"	"True"	"True"	"False"	"False"	1	1
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"True"	0	0

PATs can be used whenever complicated decisions about control input combinations need to be made in order to produce possible combinations of control actions.

Control Specs Containing PATs and DTs

This PAT contains the transition information of the previous STD.
Each row of this PAT could be viewed as representing a state - perhaps!

Door_ Position	<Start_Date> Received	<Start_Time> Received	<Duration_ Time> Received	<Setpoint_ Temperature> Received	Time_To_ Start	Time_To_ Stop	PROCESS 1	PROCESS 7
"Open"	"False"	"False"	"False"	"False"	"False"	"False"	0	0
"Closed"	"False"	"False"	"False"	"False"	"False"	"False"	0	0
"Open"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	0
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"False"	0	1
"Closed"	*Don't Care*	"True"	"True"	"True"	"True"	"False"	1	1
"Open"	"False"	"False"	"True"	"True"	"False"	"False"	0	0
"Closed"	"False"	"False"	"True"	"True"	"False"	"False"	1	1
"Closed"	*Don't Care*	"True"	"True"	"True"	"False"	"True"	0	0

PATs are best used when it is necessary to view all of the possible process activations for a given CFD.

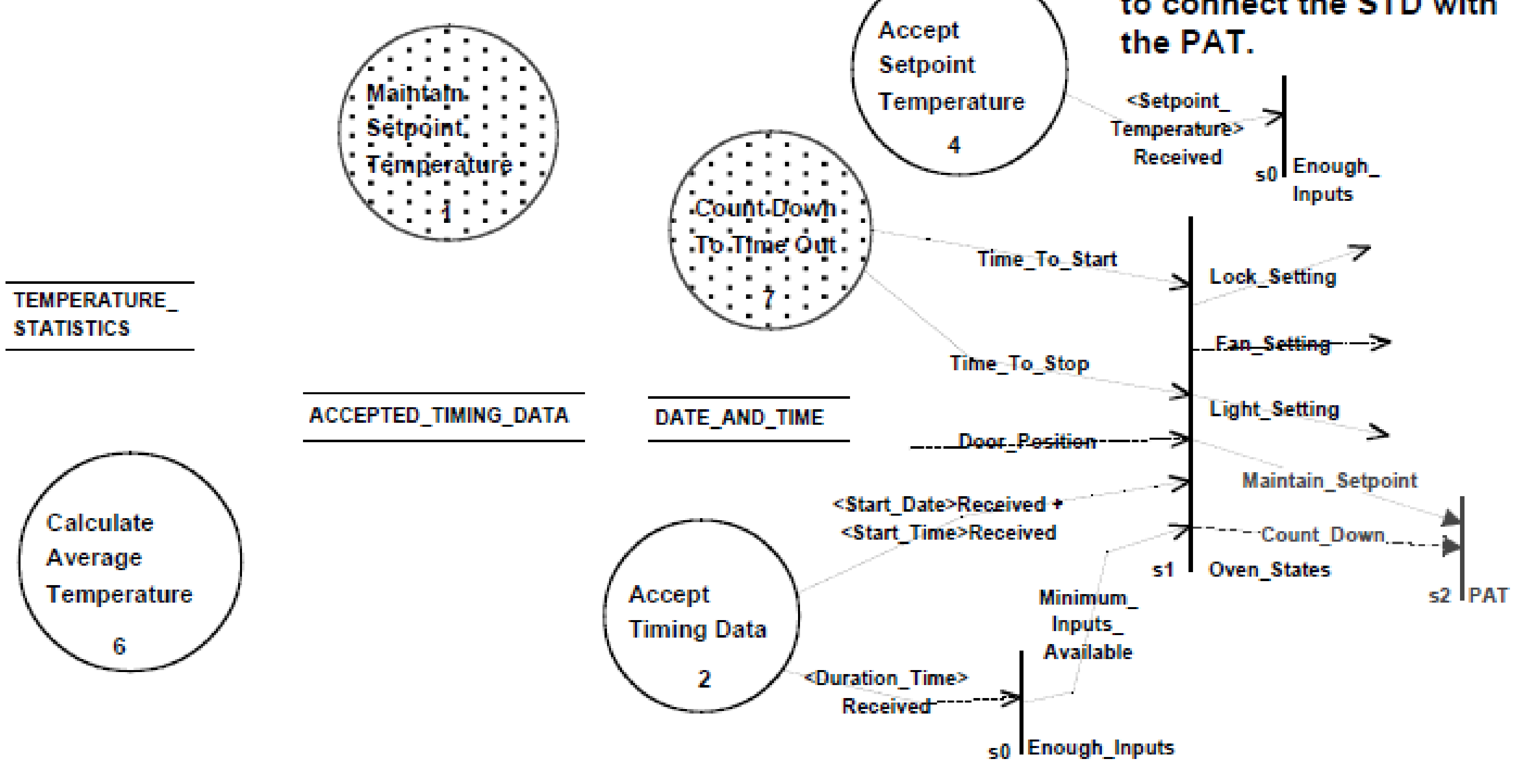
PATs should not be used in lieu of STDs because they tend to hide important state information.

Using a COMBINATIONAL FSM to represent state information creates ambiguity.
Combinational FSMs should be used only in situations when past history is unimportant.

Control Spec Containing DT, STD & PAT

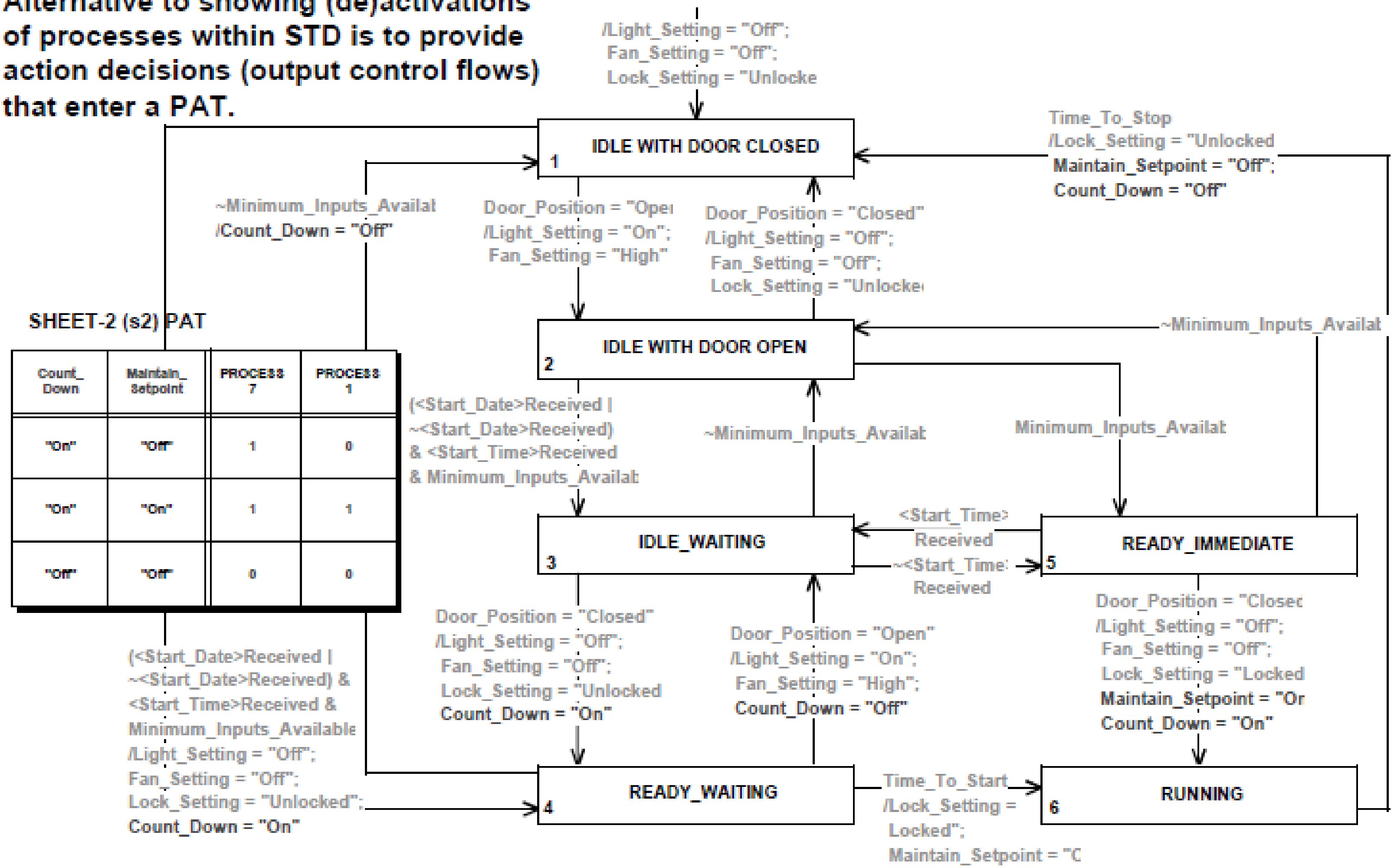
Process (de)activations resulting from control actions within an STD can be made more obvious on the corresponding CFD by inclusion of a PAT.

The disadvantage is that some superfluous control flows have to be created to connect the STD with the PAT.



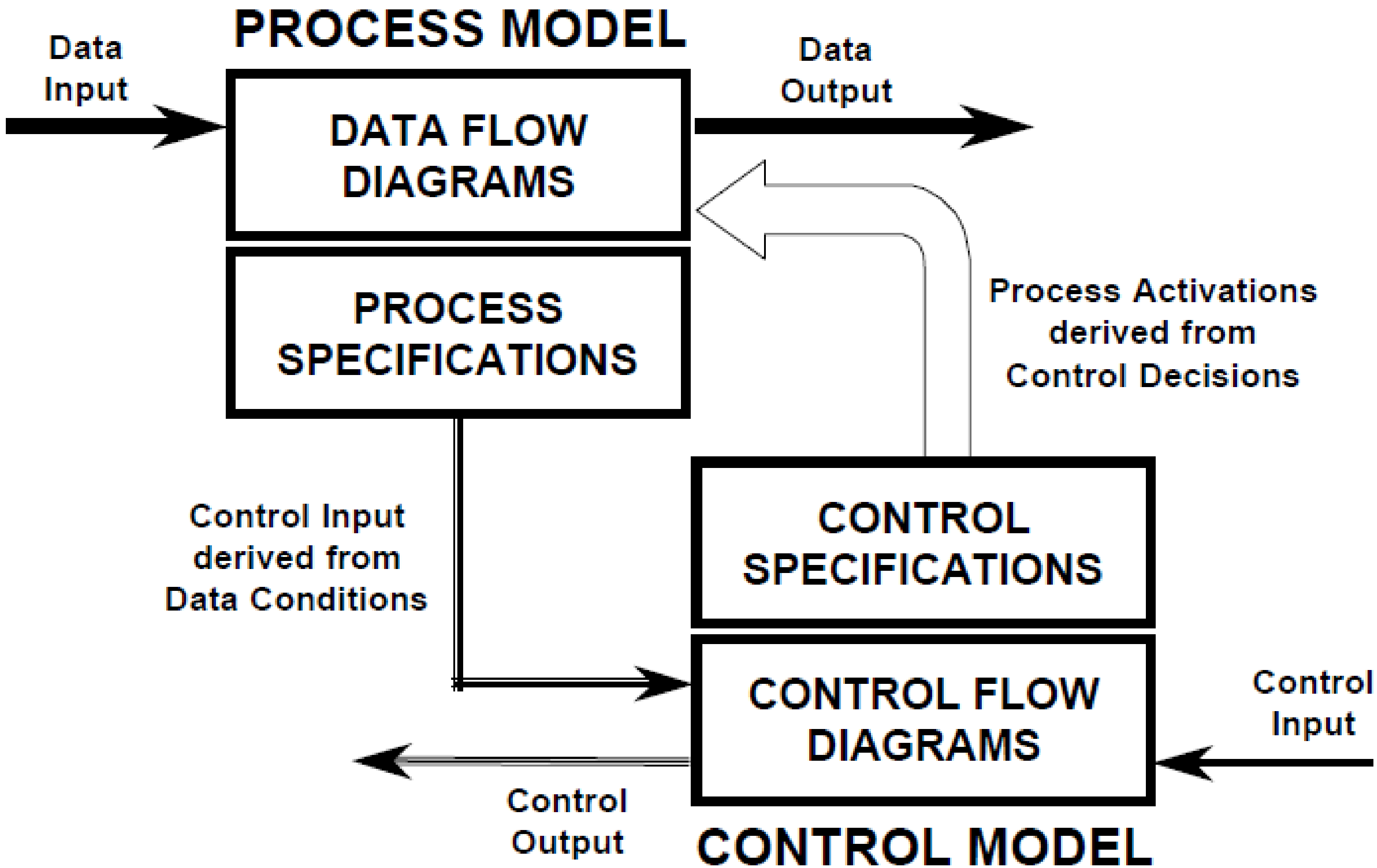
Control Spec Containing DT, STD & PAT

Alternative to showing (de)activations of processes within STD is to provide action decisions (output control flows) that enter a PAT.



Count_ Down	Maintain_ Setpoint	PROCESS 7	PROCESS 1
"On"	"Off"	1	0
"On"	"On"	1	1
"Off"	"Off"	0	0

Requirements Model Summary



Timing Specifications

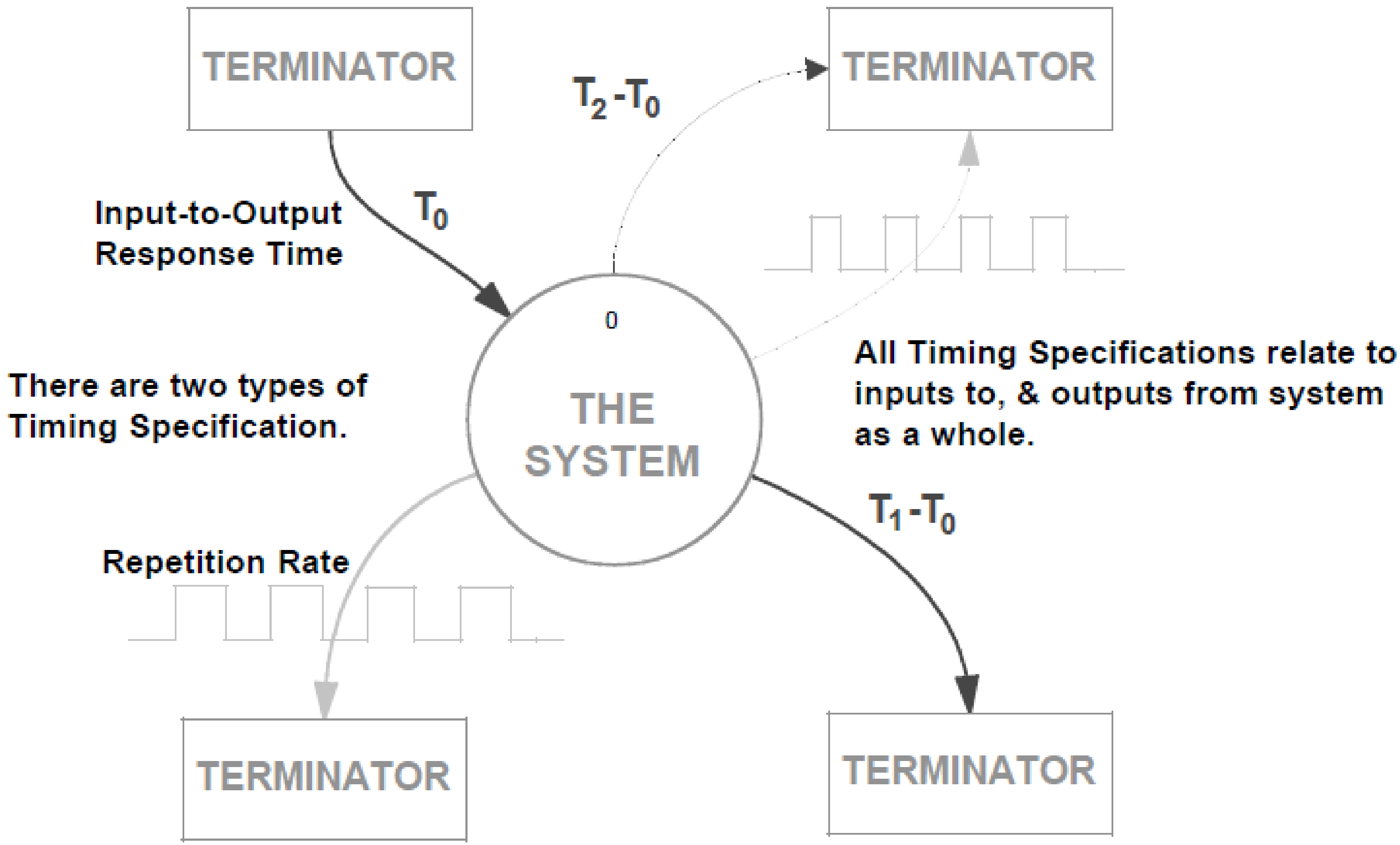
System and Software Requirements Specifications:

- are the result of analysis.
- are fully documented statements of what a system or software product must achieve.
- should not contain descriptions of system structure or how to achieve particular requirements.
- become the starting point for design.

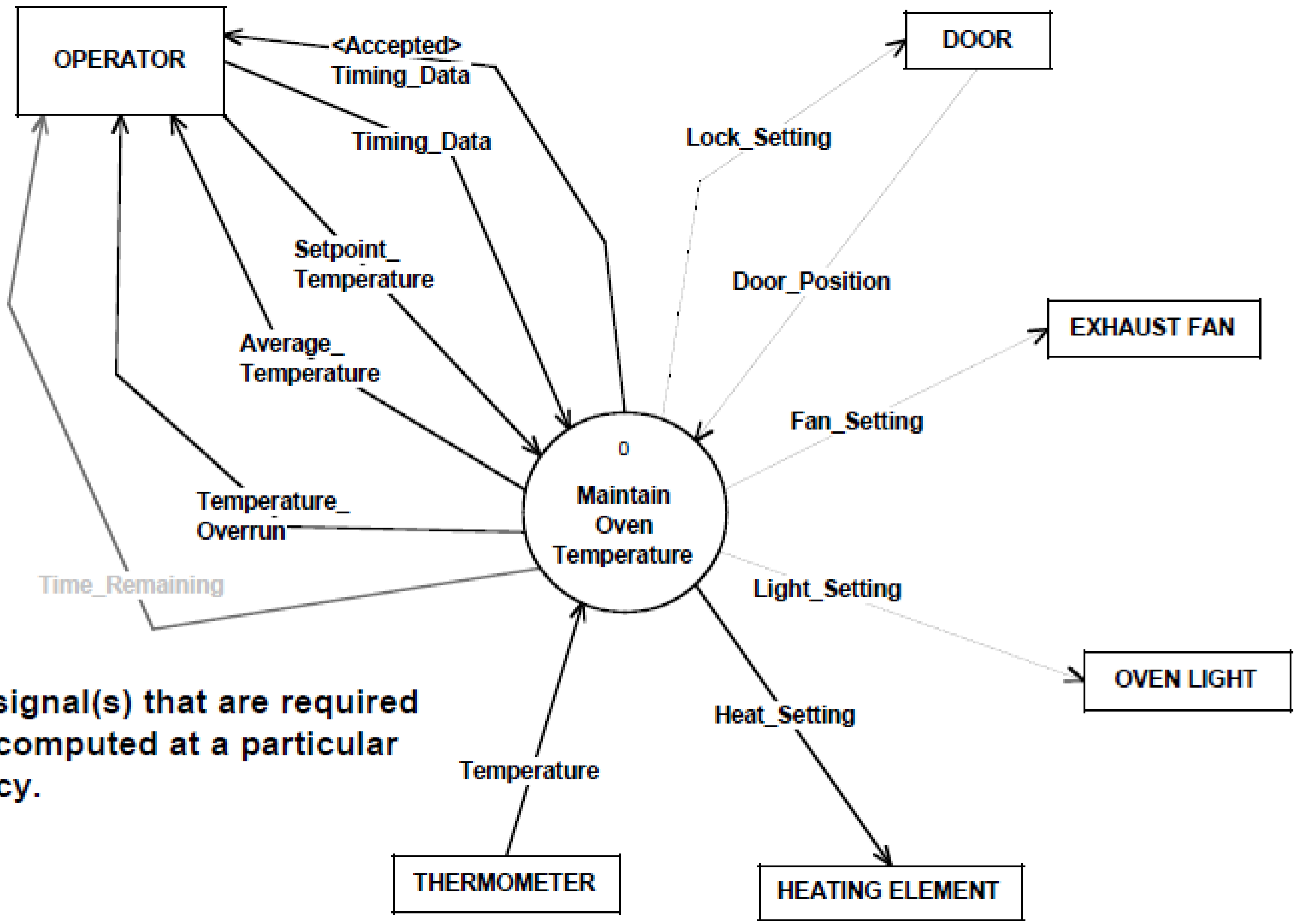
Timing Specifications therefore:

- are determined for overall (external) system behaviour.
- are not provided for particular (internal) processes since no design structure has yet been determined.

Timing Specifications



Timing Specifications – Repetition Rate



Output signal(s) that are required to be recomputed at a particular frequency.

Timing Specifications – Repetition Rate

REPETITION RATE:

- The required recomputation rate of signal outputs to the external environment.
- Is included with the description of the signal in the Requirements Dictionary.

- Example (from previous slide)

Time_Remaining = *The amount of time that remains before a run ends*

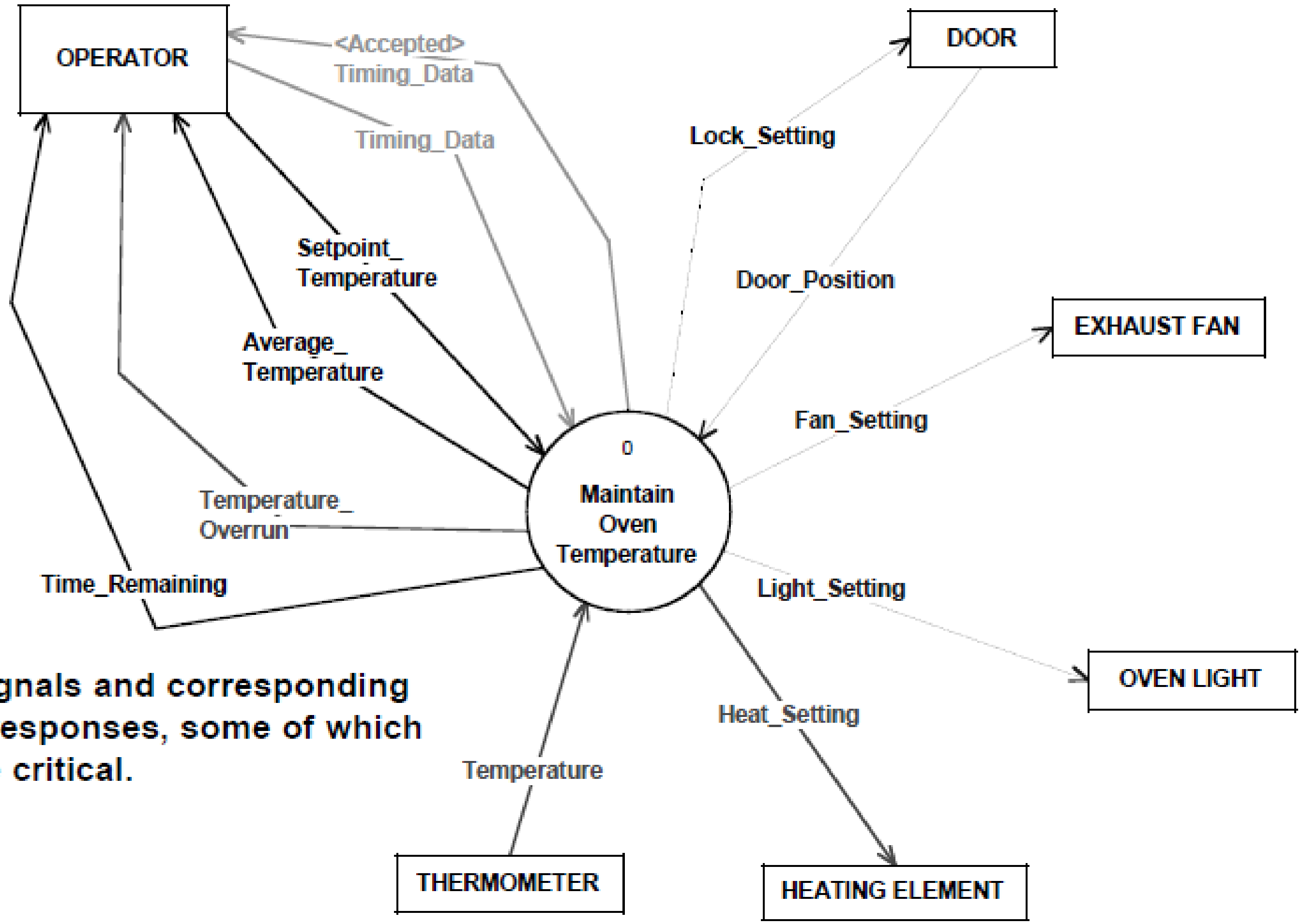
Units:	Hours:Minutes:Seconds
Range:	0:0:0 to 12:0:0
Accuracy:	1 second
Rate:	Once per minute

Timing Specification – I>0 Response Time

INPUT-TO-OUTPUT RESPONSE TIME:

- A specification of allowable timing ranges (within which the system must respond) for each input event and the resulting output response(s).
- Can be included with each of the dictionary components that contribute to the various sets of event-responses.
- Is better to tabularise.

Timing Specification – I>0 Response Time



Input signals and corresponding output responses, some of which are time critical.

Timing Specification – I>0 Response Time

Table of Specified Input-to-Output Response Times

EXTERNAL INPUT SIGNAL	EVENT	EXTERNAL OUTPUT SIGNAL	EVENT	RESPONSE TIME
Temperature	Temperature value is read	Heat_Setting	Heat_Setting value is issued	< 1 second
	Temperature outside tolerance	Temperature_ Overrun	Temperature_ Overrun value is issued	< 0.5 second
Door_Position	Door is closed	Fan_Setting, Light_Setting	Fan and Light are turned off.	< 50 milli secs
Timing_Data	Timing data is entered	<Accepted> Timing_Data	Timing_Data value is accepted	Not critical.
....	....	....	....	....

Guidelines for Using Hatley/Pirbhai

There are no strict step-by-step procedures.

However, the following guidelines are useful:

- **Construct an Event/Action list to more easily:**
  - **Identify processes that transform input data (or material) flows into output data flows.**
  - **Isolate those inputs that directly control the internal operation of the system.**
  - **Isolate those data inputs that are used to make decisions about controlling parts of the system.**
  - **Identify various operating modes together with the inputs that cause operational mode changes.**

Guidelines for Using Hatley/Pirbhai

- **Use Event/Action list to separate data signals from control signals.**
  - **Data signals usually have some content in the form of a range of non-discrete values.**
  - **Control signals will always have discrete values and tend to be used to trigger some action.**

    Activate, enable, engage, execute, stop/start, trigger, toggle.
  - **Some signals may have discrete values but not be part of the control model - they are then data signals.**

Guidelines for Using Hatley/Pirbhai

- **Use Event/Action list to establish processes first.**
  - **Because it is preferable to identify what is to be controlled before isolating control-type activities.**
    - Establishing what is to be controlled before thinking too much about control issues ensures less confusion and faster progress toward a good set of requirements specifications.
  - **Because the control model is dependent upon the process model in most systems.**
  - **By applying DeMarco approach to construct the essential functional abstractions and decompositions.**
    - This can be done by transforming the event/action list into a set of event/action diagrams before composing a set of DFDs.

Guidelines for Using Hatley/Pirbhai

- **Keep control issues at a high-level.**
  - **Because it is important to determine system operating modes (states).**
  - **Because low-level control issues tend to be more strongly related to implementation issues.**
    - Overuse of Hatley/Pirbhai Model at lower levels of functional decomposition can obscure specifications with implementation decisions.
  - **Because it will aid architectural decisions during the design phase.**

Hatley/Pirbhai & System Development