# System Testing

Chapter 14

# Overview system testing

- Common experience
  - **In everyday live – not just programming**
  - **Use functional testing**
  - **Looking for correct behaviour, not looking for faults**
- Intuitively familiar
  - **Too informal**

# Overview system testing – 2

- Little test time due to delivery deadlines
  - **Too informal**

- Need a good understanding and theory
  - **Use threads**
  - **Atomic system functions**

# Possible thread definitions

- Difficult to define
  - **A scenario of normal usage**
  - **A system-level test case**
  - **A stimulus-response pair**

# Possible thread definitions – 2

- Behaviour that results from a sequence of system-level inputs

- An interleaved sequence of port input and output events

- A sequence of transitions in a state machine description of the system

# Possible thread definitions – 3

- An interleaved sequence of object messages and executions
- A sequence of
  - Machine instructions
  - Program statements
  - MM-paths
  - Atomic system functions

# Thread levels

- **Threads can occur at what levels?**

# Thread levels – 2

- Unit level
- Integration level
- System level

# Unit level thread

- **Describe a unit level thread?**

# Unit level thread – 2

- An execution-time path of program text statements / fragments

- A sequence of DD-paths

- Tests individual functions

# Integration level thread

- **Describe an integration level thread.**

# Integration level thread – 2

- An MM-path
- Tests interactions among units

# System level thread

- **Describe a system level thread.**

# System level thread – 2

- A sequence of atomic system functions
  - **Results in an interleaved sequence of port input and output events**
- Tests interactions among atomic system functions

# Atomic system function

- **Describe an atomic system function.**

# Definition – atomic system function

- Is an action that is observable at the system level in terms of
  - **Port input events**
  - **Port output events**

- Separated by points of event quiescence
  - **Analogous to message quiescence at the integration level**
  - **Natural end point**

# Definition – atomic system function – 2

- At system level no interest in finer resolution

- Seam between integration and system testing
  - **Largest item for integration testing**
  - **Smallest for system testing**

# Atomic system function begin & end

- Where would an atomic system function
  - Begin?
  - End?

# Atomic system function begin & end – 2

- Begin at a port input event
- Terminate with a port output event

# Atomic system function graph

- **Describe an atomic system function graph.**

# Atomic system function graph – 2

- A directed graph where
  - Nodes are ASFs
  - Edges represent sequential flow from ASF to ASF

# ASF graph sink & source nodes

- **Describe the sink and source nodes of an ASF graph.**

# ASF graph sink & source nodes – 2

- A source node is an entry point in the graph
  - In SATM the card entry function is a source

- A sink node is an exit node in the graph
  - In SATM the session termination function is a sink

# ASF graph thread

- **Describe a thread in an ASF graph.**

# ASF graph thread – 2

- A path from a source ASF to a sink ASF

# Thread graph

- **Describe a thread graph.**

# Thread graph – 2

- A directed graph where
  - Nodes are system threads
  - Edges represent sequential execution of threads

# Basis for requirements specifications

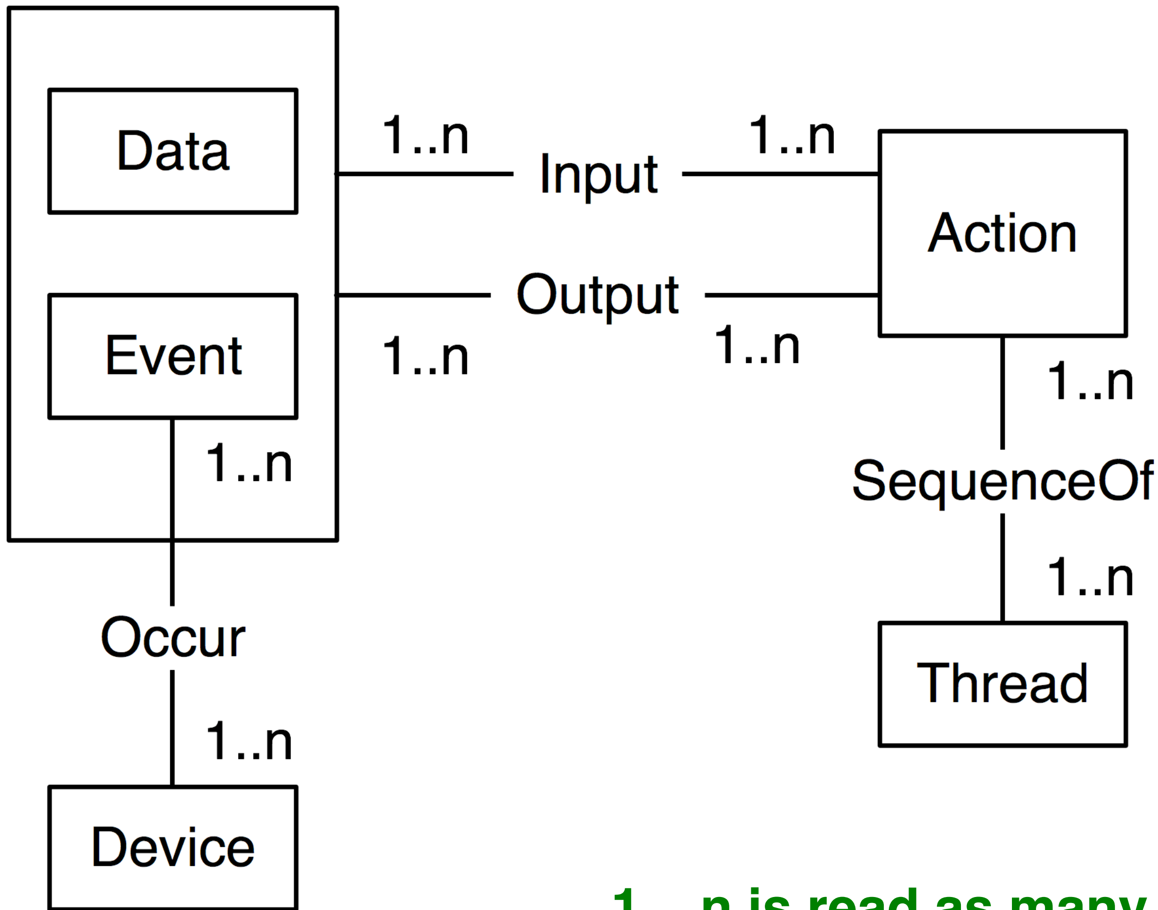
- All requirement specifications are composed of the following basis set of constructs
  - **Data**
  - **Events**
  - **Threads**
  - **Actions**
  - **Devices**

- All systems can be described in terms of the basis set of constructs

# Thread graph

- **Describe a thread graph.**

# Basis concepts E/R model




**1 .. n is read as many**

# Data

- In a system what is data.

# Data

- Focus on information used and created by the system

- Data is described using
  - **Variables, data structures, fields, records, data stores and files**

  - **Entity-relationship models describe highest level**

  - **Regular expressions used at more detailed level**
    - **Structure charts**
      - from Jackson System Development

# Data view

- **For what is a data view**
  - **Good?**
  - **Bad?**

# Data view – 2

- Good for transaction view of systems
- Poor for user interface

# Data and thread relationships

- Threads can sometimes be identified from the data model
  - **1-1, N-1, 1-N and N-N relationships have thread implications**
    - **Need additional data to identify which of many entities is being used**
      - e.g. account numbers

- Read-only data is an indicator of source atomic system functions

# Actions

- **What is the relationship between a system and actions?**

# Actions – 2

- Action-centered modeling is a common form for requirements specification
- Actions have input and output
  - **Either data events**
  - **Or port events**
- Synonyms
  - **Transform, data transform, control transform, process, activity, task, method and service**

# Actions – 3

- Used in functional testing

- They can be refined (decomposed)
  - **Basis of structural testing**

# Devices

- **What is the relationship between systems and devices?**

## Devices – 2

- A port is a point at which an I/O device is attached to a system

- Physical actions occur on devices and enter / leave system through ports
  - **Physical to logical translation on input**
  - **Logical to physical translation on output**

- Port input and output is handled by devices

# Devices – 3

- System testing can be moved to the logical level
  - **Ports**
    - **No need for devices**
- Thinking about ports helps testers define the input space and output space for functional testing

# Events

- **What is the relationship between systems and events?**

# Events – 2

- A system-level input / output that occurs on a port device
- Data-like characteristic
  - **Input / output actions**
  - **Discrete**

# Events – 3

- Action-like characteristic
  - **The physical – logical translation done at ports**

- From the tester's viewpoint think of it as a physical event
  - **Logical event is a part of integration testing**

# On continuous events

- No such thing
  - **Textbook is incorrect**

- Events have the following properties
  - **Occur instantaneously – No duration**
    - **A person can start eating and stop eating**
    - **No corresponding event eating**
  - **Take place in the real world, external to the system**
  - **Are atomic, indivisible, no substructure**
  - **Events can be common among entities**

# On continuous events – 2

- To handle duration
  - **Need start and end events**
  - **Time-grain markers to measure the duration**

- Events are detected at the system boundary by the arrival of a message

- For testing, events are also the output of a message
  - **The entry of the message to the real world is the event**

# On the temperature event

- Temperature is not an a continuous event
  - **To be continuous a continuous message would have to arrive at the system boundary**
    - **A continuous message is not a meaningful concept**
    - **Messages are discrete**

# On the temperature event – 2

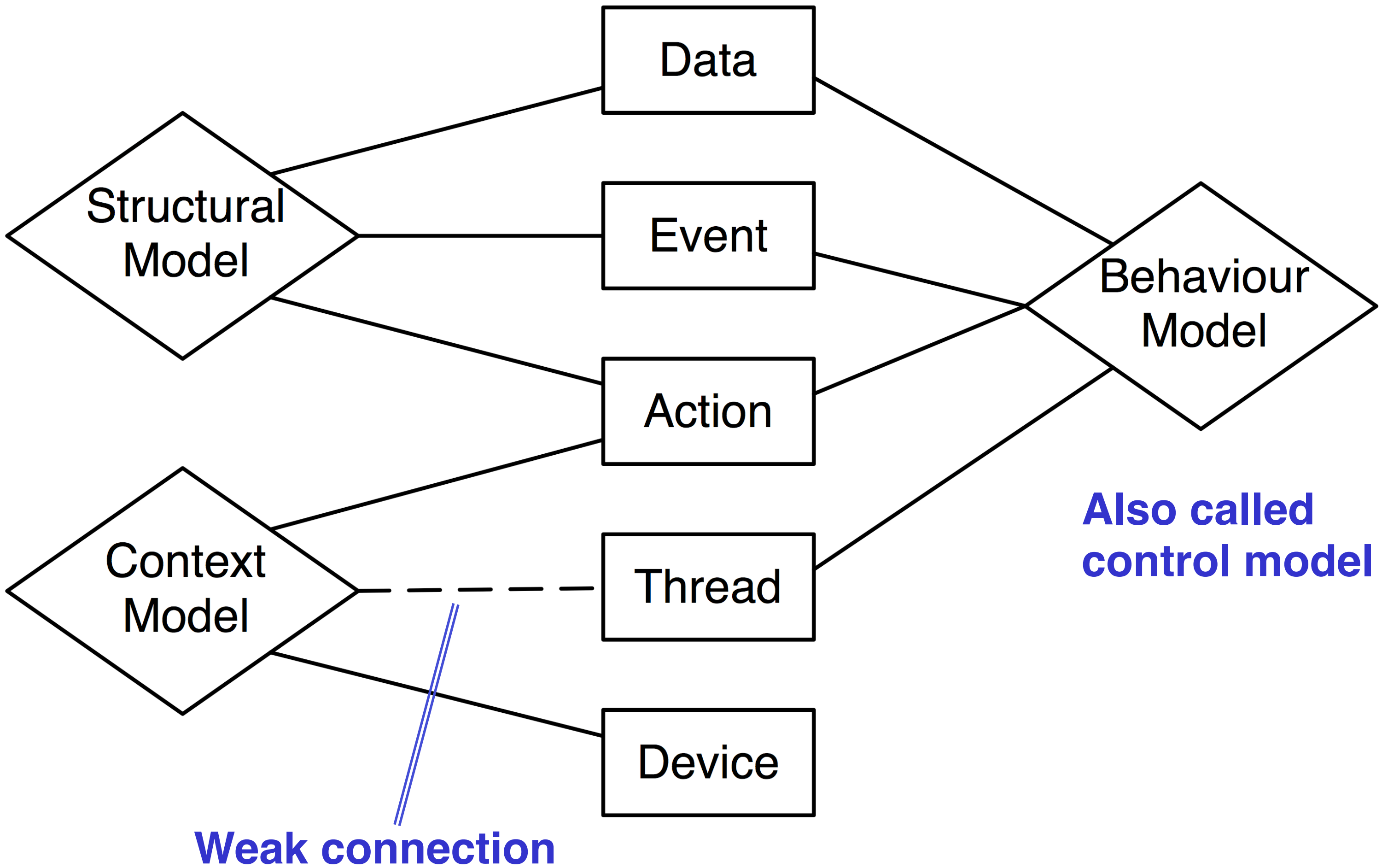
- In practice, thermometers do not send messages to a system, instead a system reads a thermometer
  - **Reading is at the discretion of the receiver not the sender**
    - **Called a statevector connection**
  - **The other option is message sending which is at the option of the sender, receiver can only read after the message is sent**
    - **Called a data stream connection**

# Threads

- Almost never occur in requirements specifications
  - **Testers have to search for them in the interactions among data, actions and events**
  - **Can occur in rapid prototyping with a scenario recorder**

- Behaviour models of systems make it easy to find threads
  - **Problem is they are models – not the system**

# Modeling with basis concepts



# Behaviour model

- Need appropriate model
  - **Not too weak to express important behaviours**
  - **Not too strong to obscure interesting behaviours**

- Decision tables
  - **Computational systems**

# Behaviour model – 2

- Finite state machines
  - **Menu driven systems**
- Petri nets
  - **Concurrent systems**
  - **Good for analyzing thread interactions**

# Finding threads in finite state machines

- Construct a machine such that
  - **Transitions are caused by port input events**

  - **Actions on transitions are port output events**
    - **Definition of the machine may be hierarchical, where lower levels are sub-machines – may be used in multiple contexts**

# Finding threads in finite state machines – 2

- Test cases follow a path of transitions
  - **Take note of the port input and output events along the path**
  - **Problem is path explosion**
    - **Have to choose which paths to test**

# Structural strategy for thread testing

- Bottom-up
  - **The only one**

# Structural coverage metrics

- Given a finite state machine with input and output ports, what structural coverage metrics could we use?

## Structural coverage metrics – 2

- Use same coverage metrics as for paths in unit testing
  - **Finite state machine is a graph**
- Node coverage is analogous to statement coverage
  - **The bare minimum**
- Edge (transition) coverage is the better minimum standard
  - **If transitions are in terms of port events, then edge coverage implies port coverage**

# Functional strategies for thread testing

- **What are they?**
  - **Look at slides ST-28 and ST-30 for a hint**

# Functional strategies for thread testing – 2

- Event-based
  - **Recall that events are port input and output**
- Port-based
- Data-based

# Port input thread coverage metrics

- Five port input thread coverage metrics are useful
  - **PI1: Each port input event occurs**
    - **Inadequate bare minimum**
  - **PI2: Common sequences of port input events occur**
    - **Most common**
    - **Corresponds to intuitive view of testing**
    - **Problem:**
      - What is a common / uncommon sequence?

# Port input thread coverage metrics – 2

- **PI3:** Each port input event occurs in every relevant data context
  - Physical input where logical meaning is determined by the context in which they occur
  - Example is a button that has different actions depending upon where in a sequence of buttons it is pressed

# Port input thread coverage metrics – 3

- **PI4:** **For a given context, all inappropriate input events occur**
  - **Start with a context and try different events**
  - **Often used on an informal basis to try to break the system**
  - **Partially a specification problem**
    - Difference between prescribed and proscribed behaviour
    - Proscribed behaviour is difficult to enumerate

# Port input thread coverage metrics – 4

- **PI5:** For a given context, all possible input events occur
  - Start with a context and try all different events

# Event-based thread testing

- PI4 & PI5 are effective
  - **How does one know what the expected output is?**
  - **Good feedback for requirements specification**
  - **Good for rapid prototyping**

# Output port coverage metrics

- Two output port coverage metrics
  - **PO1: Each port output event occurs**
    - **An acceptable minimum**
    - **Effective when there are many error conditions with different messages**
  - **PO2: Each port output event occurs for each cause**
    - **Most difficult faults are those where an output occurs for an unsuspected cause**
    - **Example: Message that daily withdrawal limit reached when cash in ATM is low**

# Port-based thread testing

- For each port
  - **Try threads that exercise ports with respect to the events in which they can engage**
  - **Useful when port devices come from outside suppliers**
  - **The many-to-many relationship between ports and events should be exercised in each direction**
    - **See E/R diagram**
- Complements event-based testing

# Event driven systems

- Event and port based testing is good for event driven systems
- Reactive systems – react to input events, often with output events
  - **Are long running**
  - **Maintain a relationship with the environment**
  - **E/R model is simple and not particularly useful**

Note: payroll example when properly designed is a long running process.  It is a sequence of payroll runs, where each run is in the context of previous runs.

# Data-based thread testing

- Good for systems where data is of primary importance
  - **Static**
  - **Transformational**
    - **Support transactions on a database**
  - **E/R model is dominant**

# Data-based thread testing – 2

- Data-based coverage metrics – based on E/R model
  - **DM1: Exercise the cardinality of every relationship**
    - **1-1, 1-N, N-1, N-N**
  - **DM2: Exercise the participation of every relationship**
    - **Does every specified entity participate**
    - **Can have numerical limits**

# Data-based thread testing – 3

- **DM3: Exercise the functional dependencies among relationships**
  - **Functional dependencies are explicit logical connections**
    - Cannot repair a machine that one does not have

# Thread explosion – Pseudo-structural testing

- Use the graph-based metrics as a cross-check on the functional coverage metrics
  - **Analogous to using DD-paths to identify gaps and redundancies of functional testing at the unit level**

- Pseudo occurs because graph is on the control model, which is not the system itself

# Thread explosion – Pseudo-structural testing – 2

- Weak method if model is poor
  - **used the incorrect model for type of system**
    - **Can be transformational, interactive, concurrent**
  - **Did not design a good model**

# Thread explosion – Pseudo-structural testing – 3

- Decision tables and finite state machines good for atomic system function testing

- Thread-based testing is best done with Petri nets
  - **Devise tests to cover**
    - **Every place**
    - **Every transition**
    - **Every sequence of transitions**

## Thread based testing problem

- **What is the big problem of using thread based system testing?**

# Thread based testing problem – 2

- What is the big problem of using thread based system testing?
  - Thread explosion

- How do we deal with thread explosion?

# Thread based testing problem – 3

- What is the big problem of using thread based system testing?
  - Thread explosion

- How do we deal with thread explosion?
  - Operational profiles

# Operational profiles

- **What is an operational profile?**

# Operational profiles – 2

- Make use of Zipf's law
  - **80% of activities occur in 20% of the activity space**
- Make use of the idea that you want to reveal faults
  - **Testing is to find cases that when a failure occurs the location of a fault is revealed**
- Make use of the fact
  - **Distribution of faults is indirectly related to the reliability of a system**

# Reliability question

- **What is system reliability?**

# Operational profiles – 3

- Make use of system reliability
  - **System reliability is the probability that no failure occurs within a given time-period**
  - **Faults are on low use threads**
    - **The system is reliable**
  - **Faults are on high use threads**
    - **The system is unreliable**

# Operational profiles – 4

- When test time is limited maximize probability of finding faults by finding failures in the most frequently used threads

# Operational profiles – 5

- Use a decision tree
  - **Works well with hierarchy of finite state machines**
  - **Estimate the probability of each outgoing transition (sum to 1)**
    - **Can get statistics from customer monitoring / feedback**
  - **Probabilities in sub-states split the probability of the parent state**
  - **The probability of a thread is the product of the transitions comprising the thread**
  - **Test from high to low probability**

# Progressive and regressive testing

- **What are they?**

# Progressive & regressive testing – 2

- Use of builds makes a need for regression testing
  - **20% of changes to a system create new faults**
  - **Regression testing takes a significant amount of time**
  - **Reduce by looking at difference between progression and regression testing**

# Progressive & regressive testing – 3

- Most common regression testing is to run all the tests
- Progressive testing needs to be diagnostic to isolate faults more easily
  - **Use short threads**
- Regressive testing not as concerned with fault isolation
  - **Use long threads**

# Progressive & regressive testing – 4

- Together have good coverage
  - **State & transition coverage sparse for progressive tests, dense for regressive tests**

- Different from operational profiles
  - **Good regressive tests have low operational probability**
  - **Good progressive tests have high operational probability**