Where Have We Been?

- Scheduling Algorithms
- Communication and Synchronization
- Software Engineering
  - Software Life Cycle
  - Specification Methods
    - Natural Language
    - Flowcharts
    - State Transition Diagrams
    - Structure Charts

Where Are We Headed?

- More Software Engineering
  - Specification Methods
    - Data Flow Diagrams/Control Flow Diagrams
    - Real-Time Data Flow Diagrams
    - Petri Nets
    - Statecharts
The View from Above

- **Informal Specification Methods**
  - Intuitive to Non-technical People
  - Difficult to prove anything
    - Natural Language
    - Flowcharts
    - Structure Charts

- **Semi-Formal Specification Methods**
  - Easy to learn, but requires training
  - Some rigor
    - Data Flow Diagrams/Control Flow Diagrams
    - Real-Time Data Flow Diagrams

- **Formal Specification Methods**
  - Difficult to Learn
  - Rigorous and Provable
    - State Transition Diagrams
    - Petri Nets
    - Statecharts
Dataflow Diagrams

• Analyze Data Flow Through System and Determine Major Functions
  • Emphasizes Flow of Data
  • Shows Concurrency Explicitly
  • Does Not Show Synchronization (implicit at best)

Process  Sink/Source  Data Store  Data Flow
(Data Transformation)
• Accel and Gyro Compensation are Concurrent
• Synchronization of Comp. Data is only Implied
• Hierarchies of DFDs Provide Process Details
• Describes What the System Does, not When
• Fixed some bugs in the [PL] diagram (magenta)
  • Use of Data Stores more closely match [JC] definition
  • Eliminated redundant stores
  • Acceleration input to “Torque Gyros” matches natural language spec.
• Added *Likely* Information to Better Justify Data Stores

  • This info is not explicitly in the spec but it makes the meaning clearer (left out to simplify the example)

  • Reinforces their existence as true Data Stores (not just temporary variables)
**Control Flow Diagrams**

- Analyze Control Flow (*Events and Actions*) and Determine Their Dynamics
  - Emphasizes Flow of Control
  - Control Transformations Represented by Attached State Transition Diagrams
  - Dynamic Behavior is Explicitly Indicated

[Diagram showing control transformation and events/actions]
Real-Time Data Flow Diagram

- Link CFD with DFD
- Introduce Synchronization Prompts
  - Enable/Disable Prompt - on until off'ed
  - Trigger Prompt - one shot
  - Activator Prompt - active for duration of state

```
L  prompt label (E,T,A)
```

(Note double-arrow meant continuous time in DFD)

- Prompts Couple CFD to DFD
Real-Time DFD Example

Real-Time DFD Example

(See next page)
STDs for Real-Time DFD Example

• Other Topologies Possible (represent the same thing as above)

Tick/T: comp. accel.  Tick/T: comp. accel; T: comp gyro; T: calc pos; T: torque gyros

Tick/T: comp. accel.

Tick/T: comp. accel.

Tick/T: comp. accel.
Real-Time Data Flow Diagrams

- The Combination of a Data-Flow Diagram, Control-Flow Diagram, and necessary State Transition Diagrams
  - All three are required
  - Each serve a unique purpose

The Real-Time Data Flow Diagram can specify data flow through a system, the timing, and synchronization.
Petri Nets

• Formal Method for Multitasking and Synchronization

“Initial Marking” Indicates How the Net Starts

• Transitions “Fire” if all Inputs have a Token
• Tokens are Consumed and Produced by Transitions
Petri Net Example [PL]

Table 1: Firing Table

<table>
<thead>
<tr>
<th>epoch</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m₀</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>m₁</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m₂</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All Transitions are Synchronized
- Transitions Triggered by Event such as Clock
Multiple Transitions Example [PL]

• Multiple Transitions are Enabled Simultaneously
• Evaluate All Transitions to See Which Fire
Petri Nets Can Be Non-Deterministic

- What Transitions are Enabled?
- Can Both T2 and T3 Fire?
- Beware the Errant Token
Associating Entities with Tokens and Transitions [PL]

- **Complex Multiplication:** \((a+bi)(a+bi) = a^2 - b^2 + 2abi\)
  - Tokens: intermediate results (variables)
  - Transitions: operations
  - Creation of tokens tied to external event
Escort Tokens and Inhibitors

• Basic production facility *(conveyor contention)*

![Diagram showing basic production facility with conveyor contention]

- Loading Dock
- Staging Area
- Work Station
- Finished Inventory
- Staging Area
- Loading Dock

unload truck
transport (conveyor in)
assemble
transport (conveyor out)
load truck

• Escort token to eliminate conveyor contention

![Diagram showing escort token to eliminate conveyor contention]

- Loading Dock
- Staging Area
- Work Station
- Finished Inventory
- Staging Area
- Loading Dock

unload truck
transport (conveyor in)
assemble
transport (conveyor out)
load truck
Escort Tokens and Inhibitors (cont.)

- Inhibitor to protect only conveyor (not entire station)

- To deal with finite time transitions, add “done” indicator that produces a token
Elements of Statecharts

- Finite State Automata
- Depth - states inside states (hierarchical)
- Orthogonality - concurrent processes
- Broadcast Communications - synchronization
Unified Modeling Language

- Aimed at Object-Oriented Systems Design
- Independent of Design Process
  - Modeling Language (semantics and notation)
  - Process of Application of that Language
- UML’s Characteristics
  - Semi-formal Language
  - Discrete Systems (vs. continuous)
    - based on finite state machines

semantics ==> “meaning”
syntax ==> representation of semantics
UML Components

• 3 Distinct Models
  • Requirements Model
    • Black box, hiding implementation
    • Actors and use cases (external to system)
  • Structural Model
    • Classes and how they collaborate
    • Associations among classes
    • Components (executables) and nodes (CPUs, I/O, etc)
  • Behavioral Model
    • Statecharts

• Refs:
  • Real-Time UML, Douglass, Addison Wesley