DMTF Application Modeling and Extensions for Behavior

Karl Schopmeyer

k.schopmeyer@opengroup.org

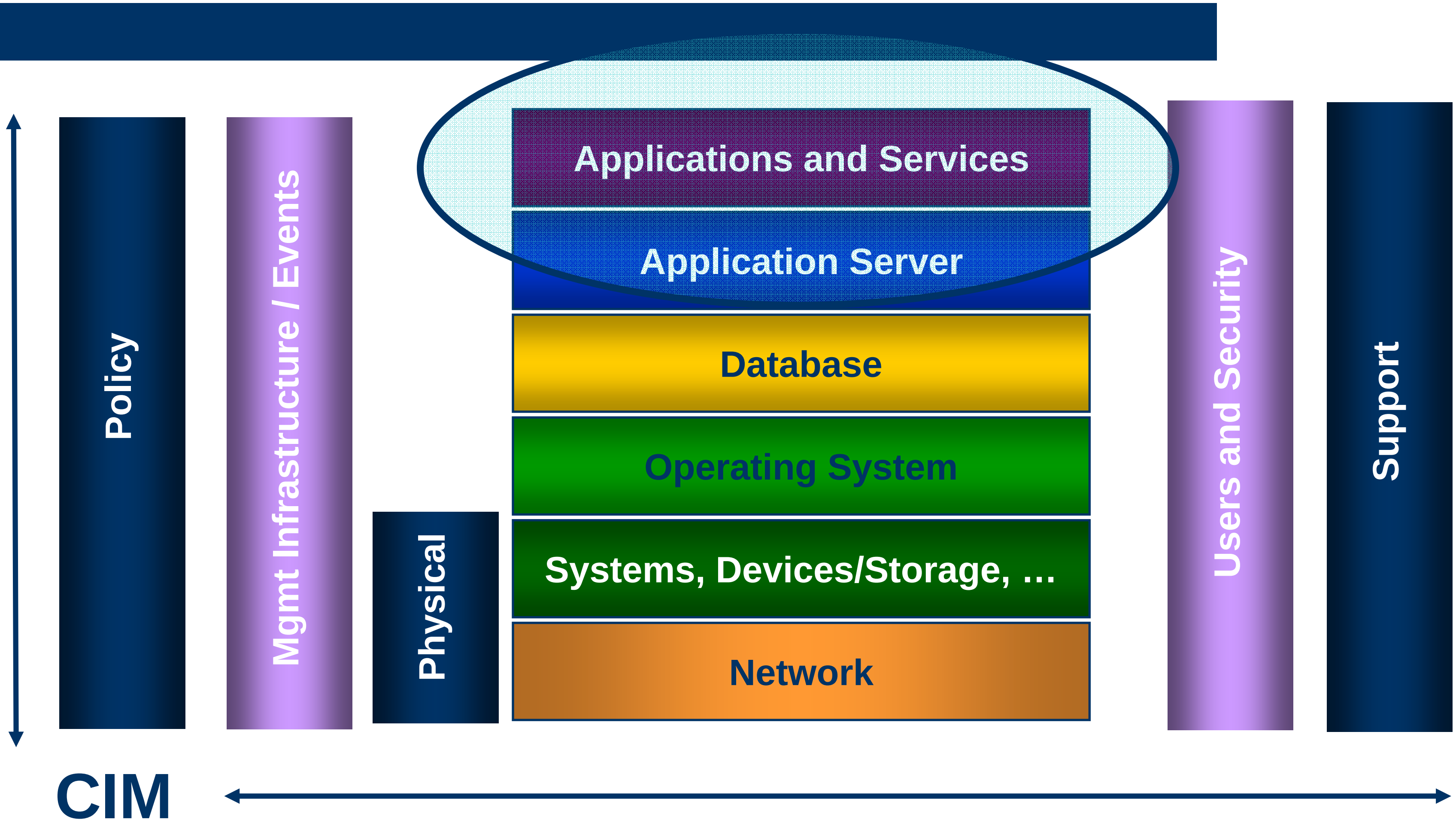
Presentation for GGF12 CIM GS sessions

V 1.1, 22 Sept 2004

Subjects

- Overview of DMTF / Open Group work in Applications Management
- New Work, Modeling Behavior and State Management in DMTF

CIM’s Coverage



Application Management Modeling Overview

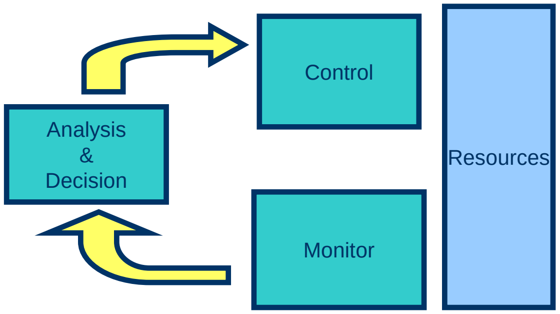
- Characteristics of an Application Management Model
  - Lifecycle management
    - Definition, Deployment, Installation, Configuration, Execution Control
  - Runtime management
    - Performance management, Service Level (QOS) management Problem Management, fault analysis, etc.
    - Inventory Management
    - …
- Contributing Management Information to the next Higher layer
  - Business Process Management
  - Service Level Management

Goals

- Management of wide range of applications
  - Distributed
  - Dynamic
  - Multicomponent
  - Large-scale
- Active management of applications as services
  - Not just Monitoring
  - Active, adaptive management

Model Components that Come Together for Application Management

- **Managing The Application**
  - Lifecycle (Deployment, Installation, Configuration, Execute)
  - Runtime (modeling the runtime structure, managing performance, Service Levels, fault determinations, …)
- **Measuring Application Traffic Flow**
  - **Metrics, Unit of Work (UOW), ARM API**
- **Automation**
  - From monitoring to management
  - From management to adaptive management (service optimization)
    - Policy
    - Service Levels, Quality of Service

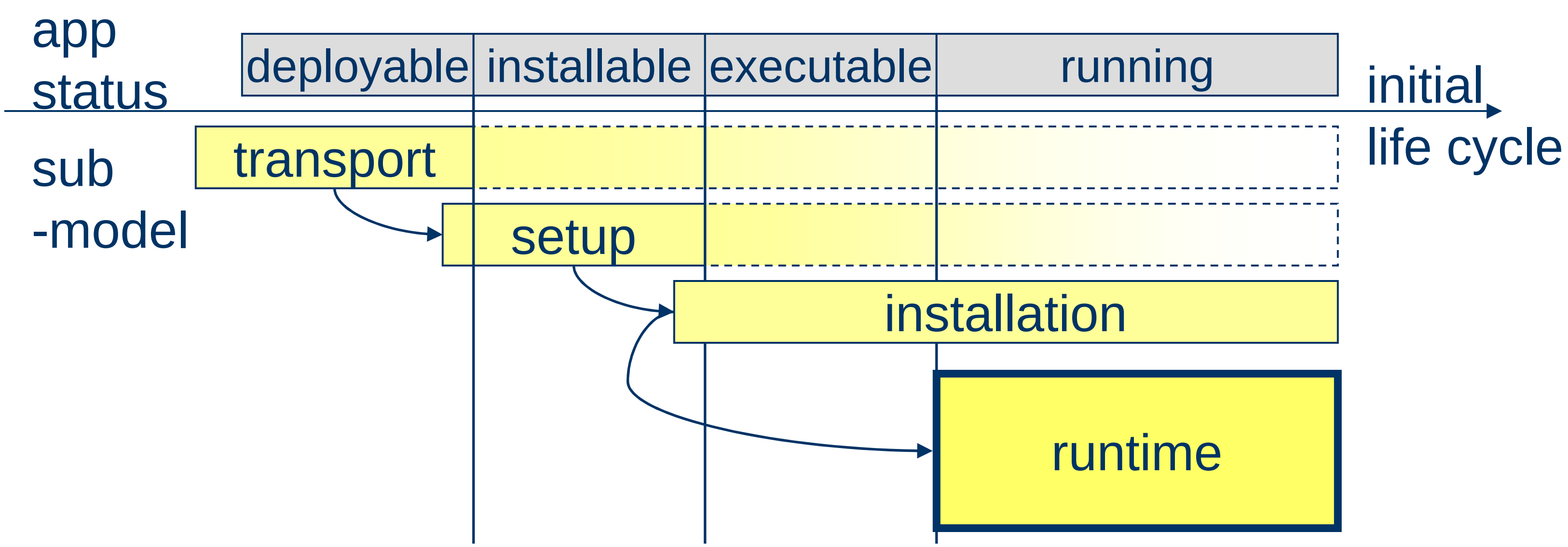


The Applicable DMTF Groups

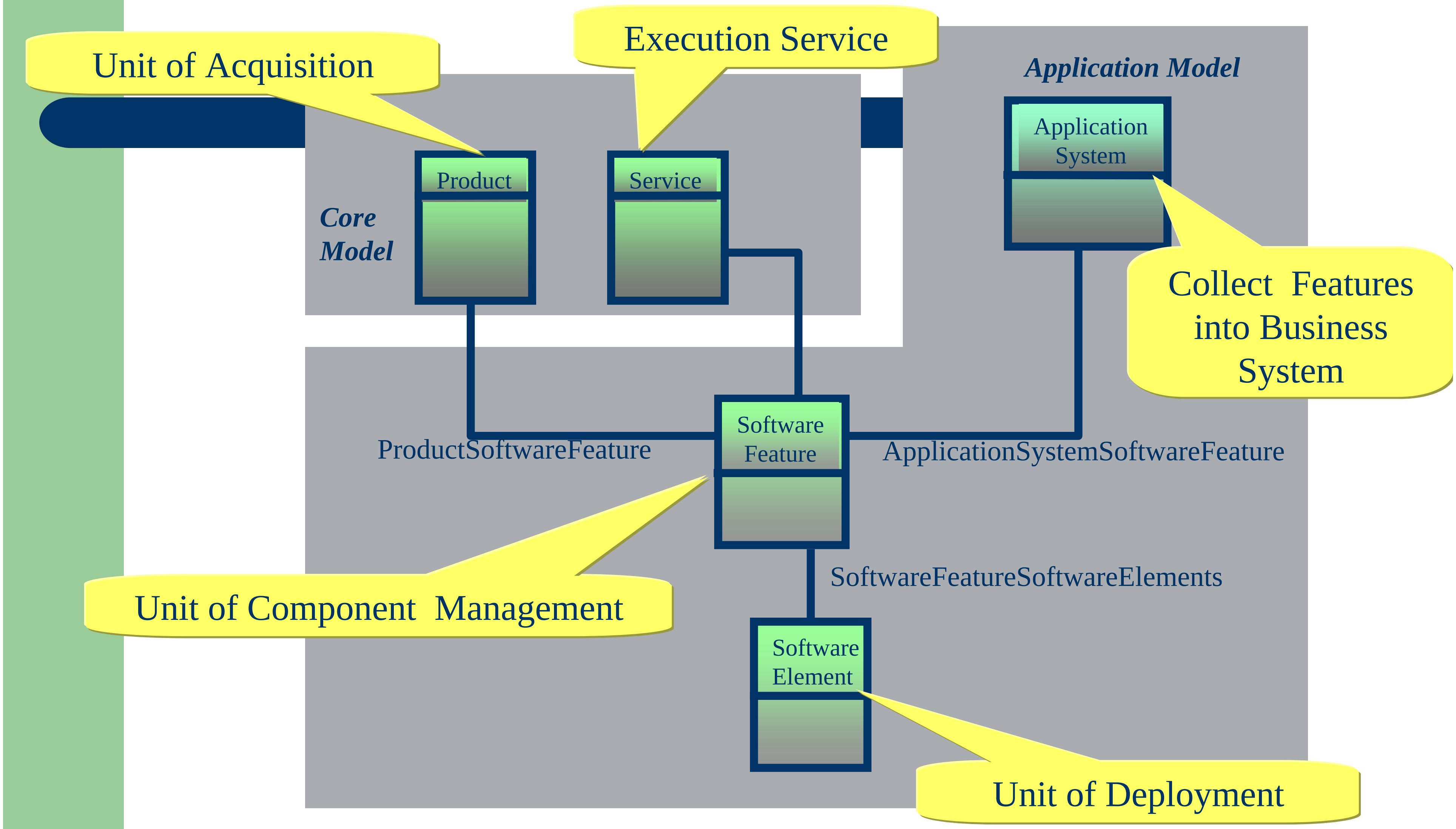
- Application Work Group
  - Runtime Model
  - Lifecycle Model
  - J2EE JSR 77 Model
  - Metrics
  - Unit of Work
- Database Work Group
  - Database model
- Policy and SLA Work Group
  - Policies

Application Management

- Lifecycle Model
- Runtime Model

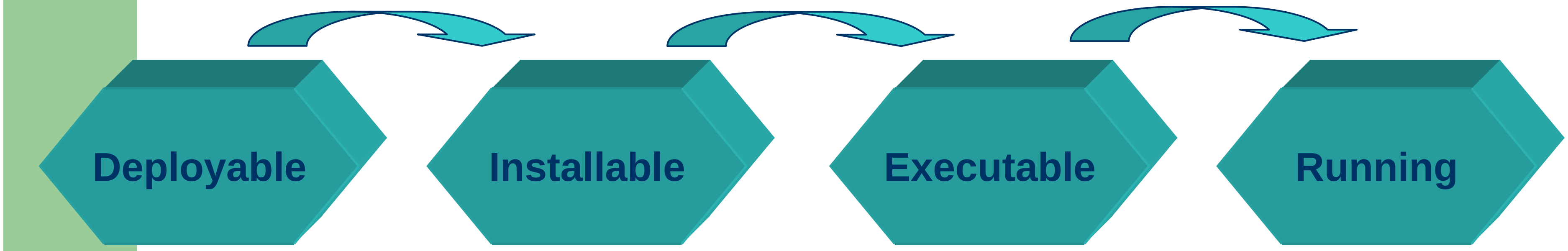


Lifecycle Model Overview

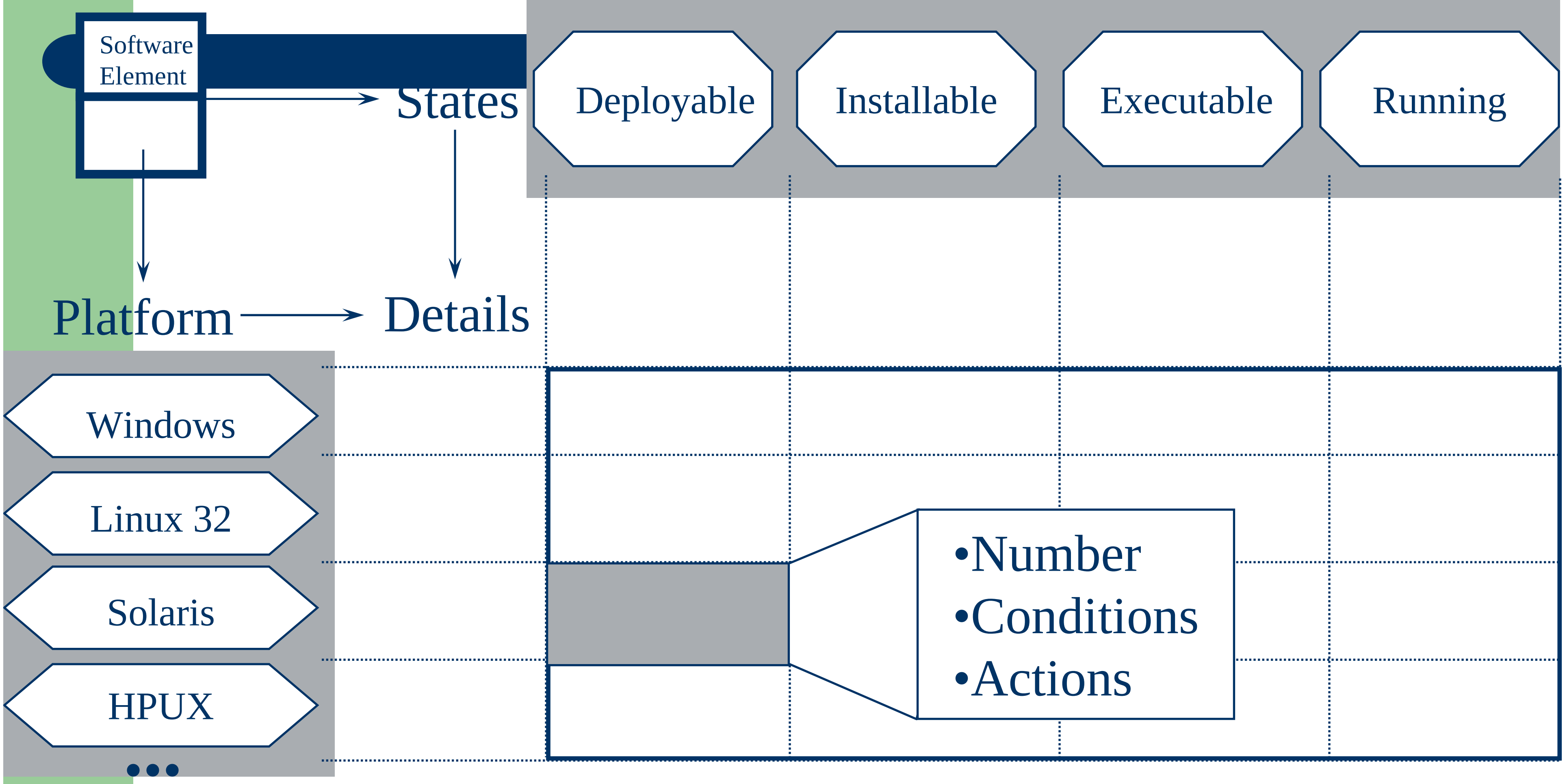


Application Life Cycle

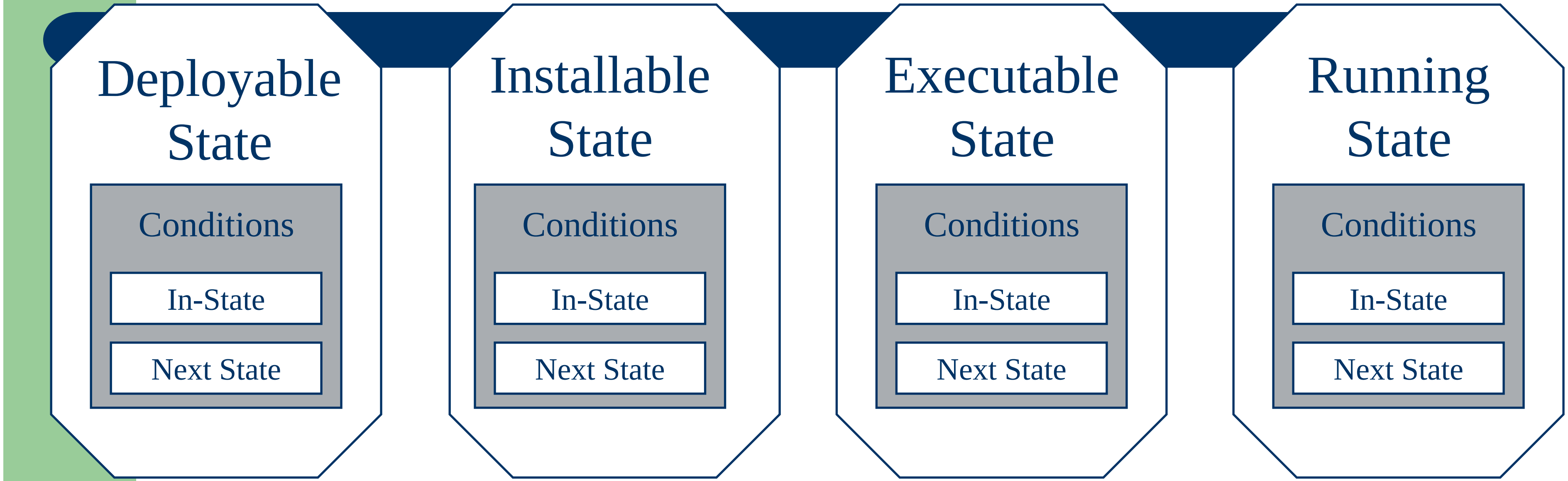
- Critical states in process of transition from development to operational
- Applies to lowest-level component
  - Software Elements
- States



Refining Software Element



Software Element Conditions



Deployable State

Conditions

In-State

Next State

Installable State

Conditions

In-State

Next State

Executable State

Conditions

In-State

Next State

Running State

Conditions

In-State

Next State

Conditions are situations that are expect to exist or not exist in an environment

In-State Conditions are characteristics of an environment that contains an element

Next-State Conditions are characteristics that need to be true in the target environment for the next state of a software element.

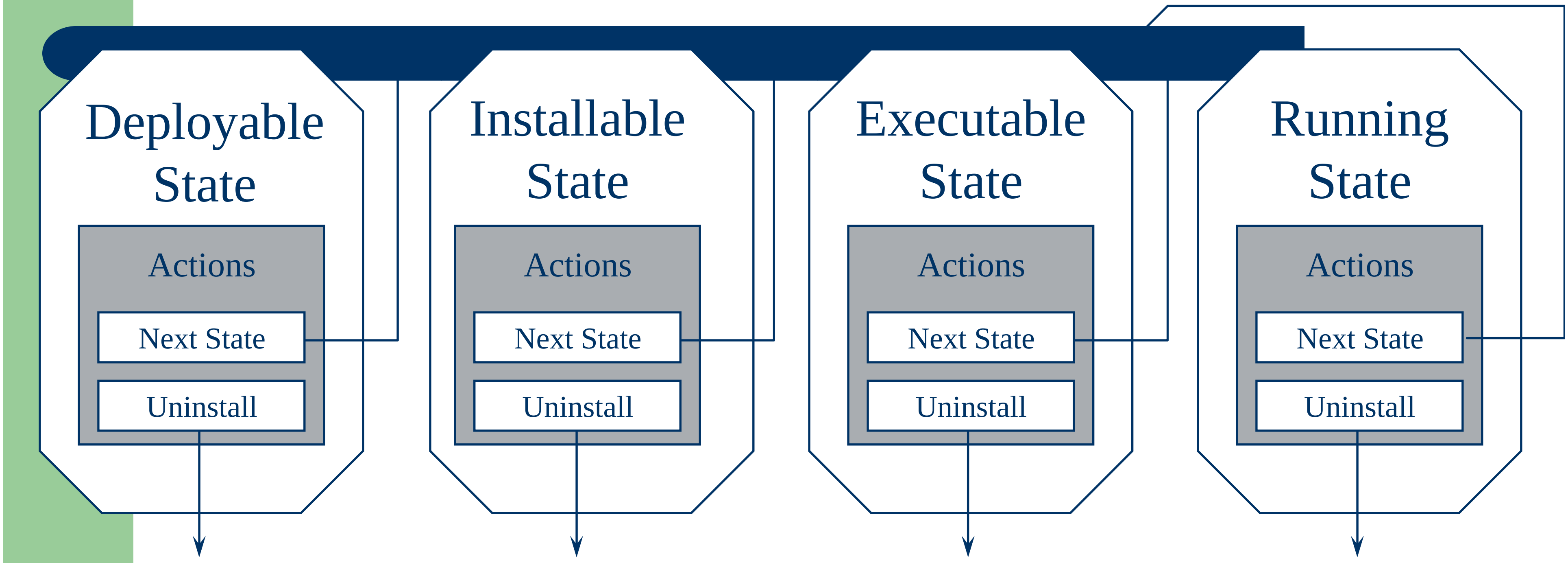
Software Element Conditions

Condition	In-State Interpretation	Next-State Interpretation
Memory Requirements	Minimum Amount of memory required to transition into the ***current*** state.	Minimum amount of memory required to transition into the ***next*** state
Disk Space	Minimum amount of disk space required to transition into the ***current*** state.	Minimum amount of disk space required to transition into the ***next*** state.
Swap Space	Minimum amount of swap space required to transition into the ***current*** state.	Minimum amount of swap space required to transition into the ***next*** state.

Software Element Conditions

Condition	In-State Interpretation	Next-State Interpretation
Architecture	The architecture required by a software element in the current state.	The architecture required by the software element to transition into the ***next*** state.
Files	A file that is expect to exist or not exist when a software element is in the ***current*** state.	A file this is expect to exist or not exist before a software element transitions into the ***next*** state.
Directories	A directory that is expect to exist or not exist when a software element is in the ***current*** state.	A directory this is expect to exist or not exist before a software element transitions into the ***next*** state.
OS Version	The version or ranges of versions a software element requires in its ***current*** state.	The version or ranges of versions a software element elements requires before it transitions into the ***next*** state.
Software Elements	A software element that is expect to exist or not exist when a software element is in the ***current*** state.	A software element that is expect to exist or not exist before a software element transitions into the ***next*** state.

Software Element Actions

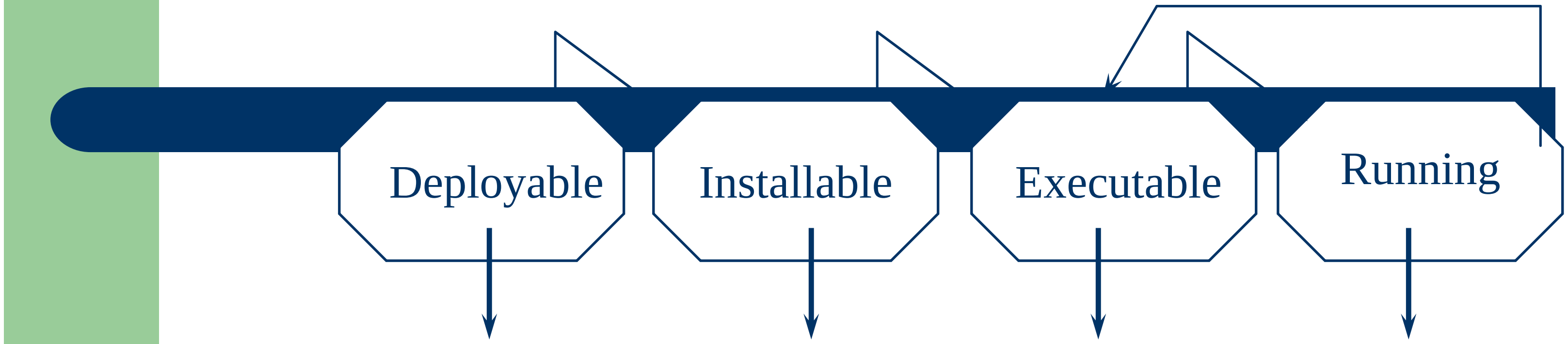


Actions are a sequence of operations

Next State Actions create a software element in a particular state.

Uninstall Actions properly remove a software element

Software Element Actions



Actions	Description
Directory	An action to create or remove a directory.
File	An action to create or remove a file.
Re-boot	An action the signals the need to reboot the computer system.
Execute Program	An action that execute a program. This can be the install script or program (e.g., setup.exe) when a software element in the installable state transitions to the executable state.

Application Management and SLAs

- The Business issue is providing services, not just the applications.
- SLAs are the contractual/agreement model for service level
- SLOs(service Level Objectives) are the service goals required to satisfy SLAs.

- **Both the systems and the applications are part of the service level determination**

Application Management and SLAs

- Typical runtime service level parameters
  - User perspective on performance
    - Interactive responsiveness
      - Transaction Response time / Time to accomplish
      - Throughput / How many simultaneous users or how many things can be done in a defined time
    - Batch turnaround
    - Critical deadlines (e.g. end-of-month processing)
  - Availability
    - Percentage of time service is available
    - Maximum limits on service-down times
- Other non-runtime SLA issues
  - Recoverability
  - Data Integrity
  - Problem responsiveness
  - Affordability

Goals of Application Measurement

- Provide Monitors for
  - Service Level management
    - Need information and controls so that analysis can be done and decisions made and implemented
  - Business and Business Process management
- Provide Application Controls for
  - Fault Determination
  - Performance characteristic attribution
  - Application monitor, management and manipulation in terms of application components, aggregation into whole to support SLOs

OR

- Monitor to provide information for SLA reporting
- Provide controls for SLA tuning
- Provide means to find why not meeting SLAs

It is not enough to know you have a problem if you do not know why or how to solve the problem.
It is even more worthless to have a means for defining SLAsAnd SLOs and no means to measure them on the system.

What are Applications?

- Complex collections of software components
- Multilayered functionally
  - E.g. Presentation, application, database, etc.
- Dynamically assembled
- GOALS of DMTF Runtime Management
  - Model the components as viewed in runtime including the interactions
  - Aggregate the information into the whole
  - Disaggregate information from whole into the components

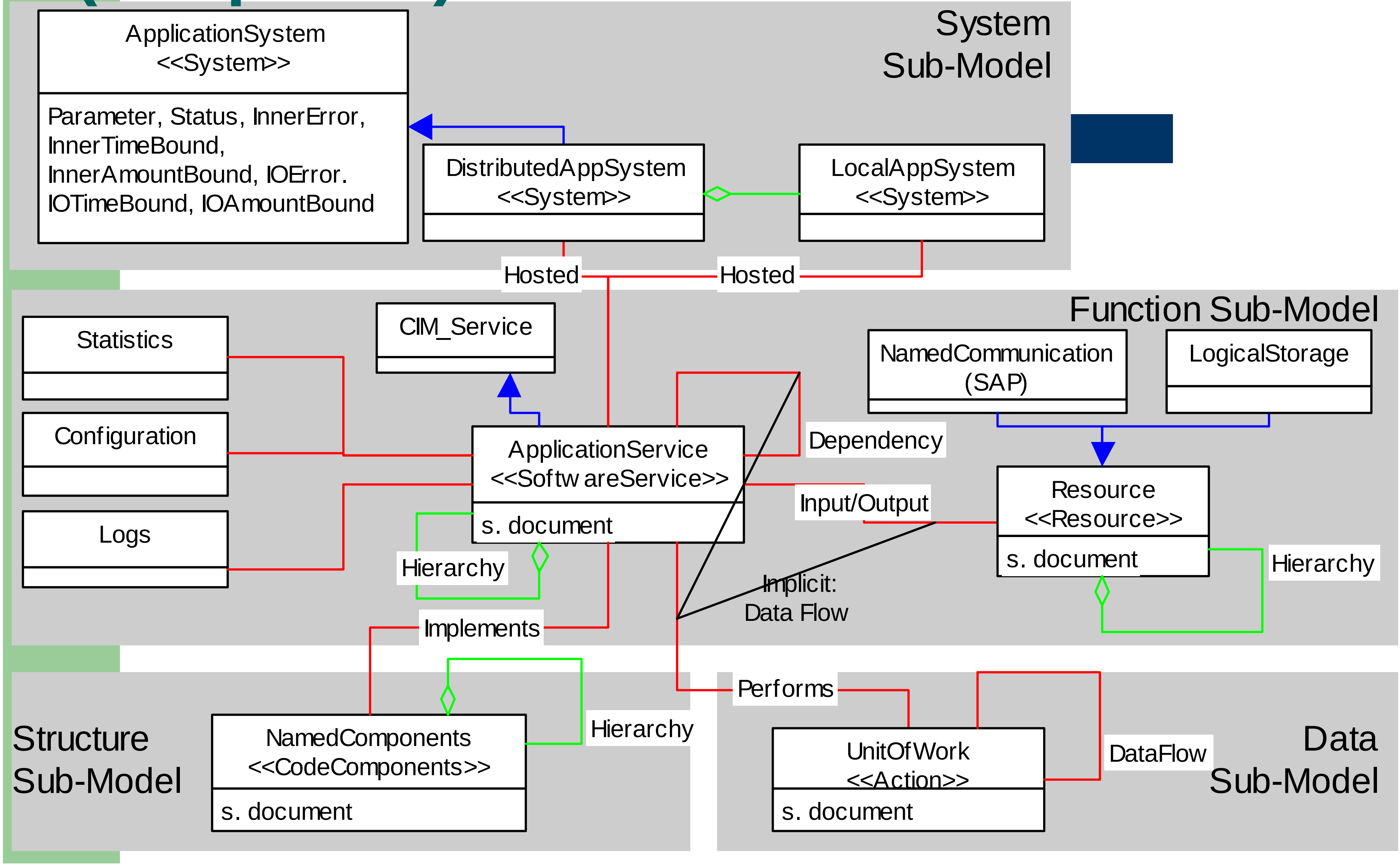
Application Runtime Manageability Requirements

- Define logical runtime structure of complex applications
- Define Application components/layers
- Support distributed and dynamic applications
- Relate physical structures and logical runtime structures
- Model usage of system resources as viewed by the application
- Model dataflow between components and applications and between applications
- Relate Unit of Work information to runtime structure
- Allow monitor and control of application state
- Support fault management
- Aggregate information from components to the whole

Modeling FCAPS

- Fault
  - Indications
  - Error and status properties (counter, information)
  - Log-entries, traces, etc.
- Performance
  - Base metrics (IO, timebound metrics, etc.)
  - UoW
  - Metric properties
  - Statistics
- Configuration
  - Persistent configuration information: configuration, settings
  - Control: methods
  - Current configuration: object properties, support classes, associations

App Runtime Model Concepts (Simplified)



System Sub-Model

ApplicationSystem
<<System>>

Parameter, Status, InnerError, InnerTimeBound, InnerAmountBound, IOError. IOTimeBound, IOAmountBound

DistributedAppSystem
<<System>>

LocalAppSystem
<<System>>

Hosted

Hosted

Function Sub-Model

Statistics

Configuration

Logs

CIM_Service

ApplicationService
<<Softw areService>>

s. document

Hierarchy

Dependency

Input/Output

Implicit:
Data Flow

NamedCommunication
(SAP)

LogicalStorage

Resource
<<Resource>>

s. document

Hierarchy

Implements

Structure Sub-Model

NamedComponents
<<CodeComponents>>

s. document

Hierarchy

Performs

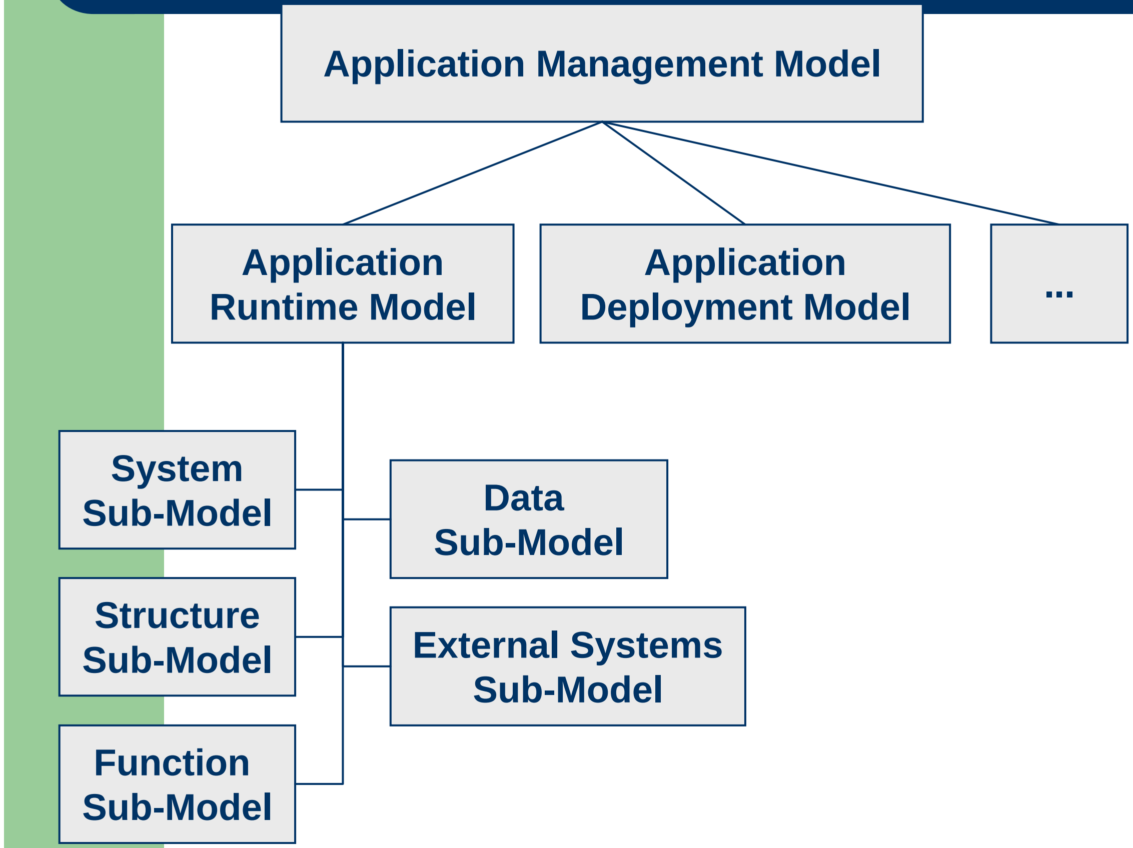
UnitOfWork
<<Action>>

s. document

DataFlow

Data Sub-Model

Application Model Hierarchy



Application Management Model

Application Runtime Model

Application Deployment Model

...

System Sub-Model

Structure Sub-Model

Function Sub-Model

Data Sub-Model

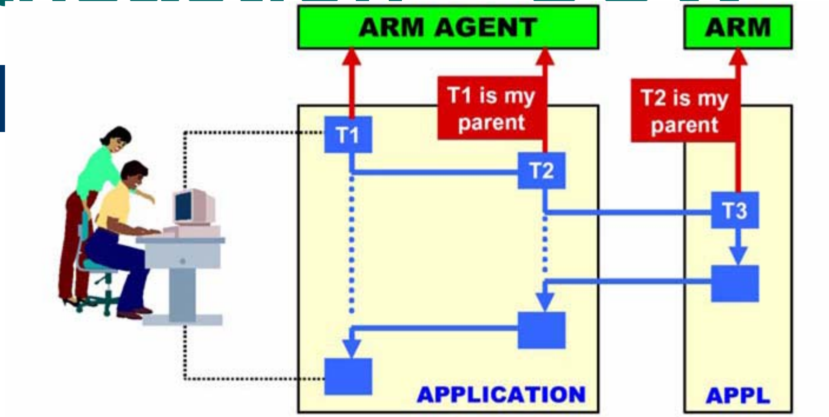
External Systems Sub-Model

- In development today
- Application System submodel (CIM 2.8)
- Components of Function submodel (CIM 2.9 prelim)
- Data submodel, structure submodel planned for future CIM verisons (2.10, etc.)

Measuring Traffic Service

- Goals
  - Identify and measure traffic characteristics (response time, metric information associated with the traffic, etc.)
- DMTF - Unit of Work(UOW)
  - Model dedicated to the concept of modeling time intervals
- Open Group - ARM
  - API dedicated to instrumenting for measurement of time intervals.

Modeling The Transaction - UOW



- Measure a time interval
- Identify the transaction
- Identify the application
- Provides information for correlation of multiple measurements
- Provides information to understand component UofWork (parent/child units of work)
- Provides metric information places for resource, etc. information
- Marry with the instrumentation technology - ARM

Unit of Work

- Defines a type of work
- Represents a UOW that has started and may have completed executing
- Associated to a UOW definition
- Provides information such as:
  - Response or elapsed time
  - Status
    - Active, Suspended, Completed (with status), Aborted
  - Metric Information about the UOW
- Examples
  - Update account balance
  - Execute batch
  - Query Data server
  - Execute subroutine

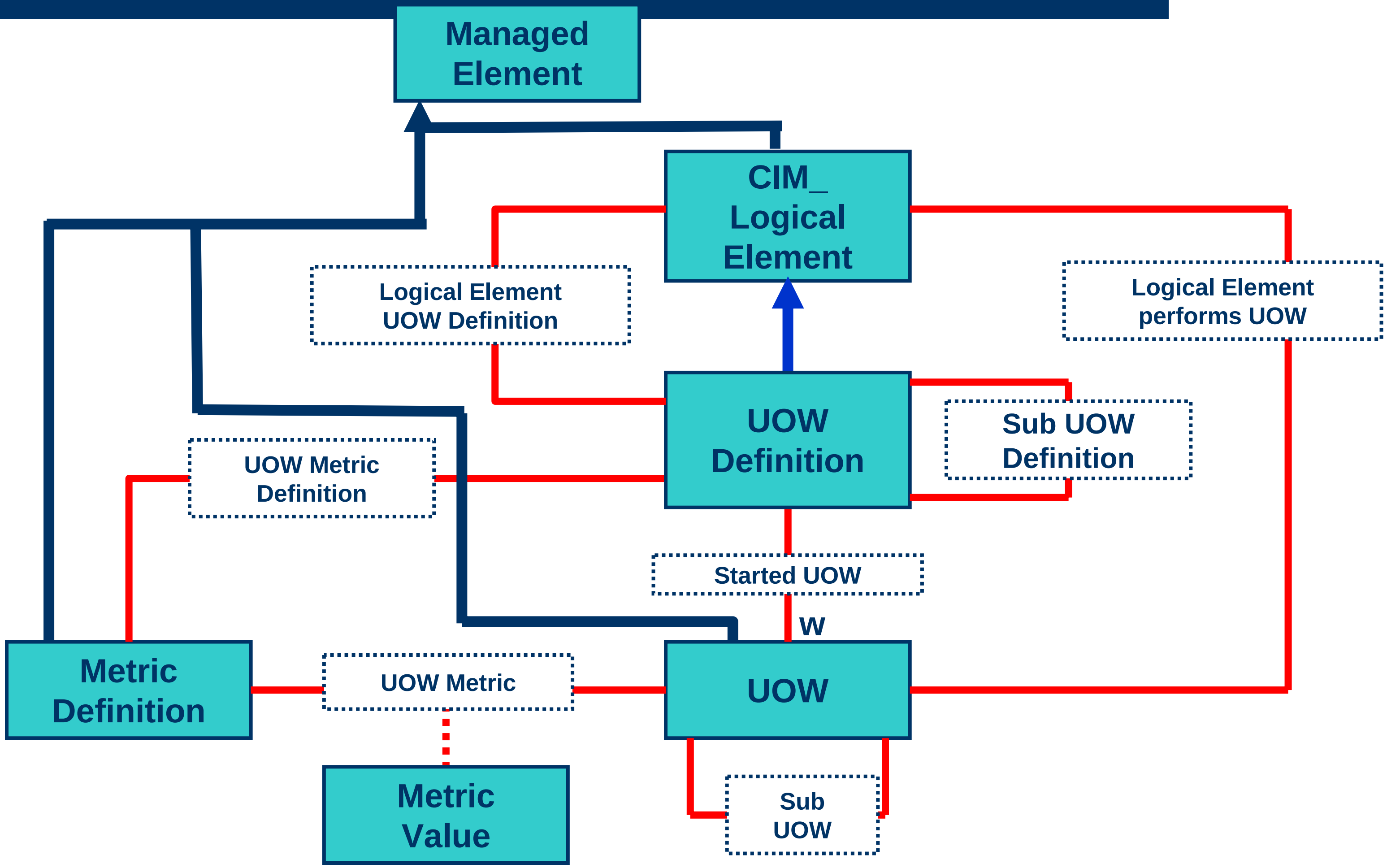
Status

- UOW model
  - Model Developed by DMTF Application Work Group
  - Corresponds today to ARM 1, 2, 3
  - Working on ARM 4 equivalent model
- ARM
  - ARM API for C and Java today (Open Group Standard
  - Version 4 extend model to more useful metrics, correlation.

Metrics Model

- Capture dynamic metric information
- Provide means to do predefine structuring and organization of the data
  - Time series
  - Computation such as summing, averaging, etc.

Metrics and UOW model



Other DMTF Work

- Service Level Agreements and Service Level Objectives
- Policy

Modeling Behavior

- Behavior and State, Extending CIM to Behavioral Control

CIM includes Behavior Today

- The model includes methods which represent behavior(ex. Start(), Stop())
- Some specific classes (ex. application model) have been able to model specific behaviorial characteristics (Deployment states and checks and actions classes.

BUT

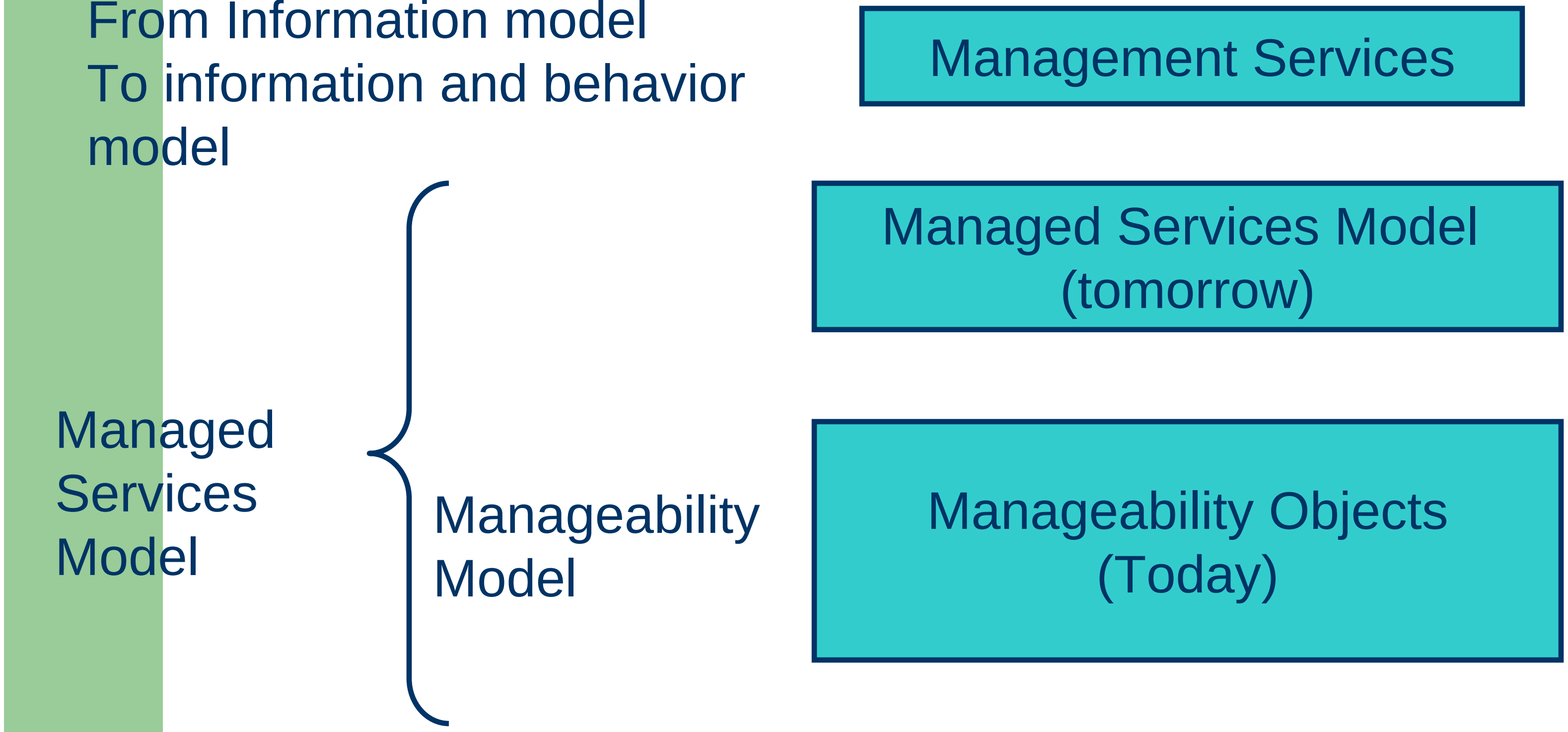
- Cannot define behavioral interactions between classes
  - Change to instance of class A causes creation of instance of Class b and an association to be estabished between A and B.
- Cannot impose behavioral control on instances
  - Ex. Accept this method only when this property set.
  - Model cannot define when a Start() method should be allowed

Modeling Behavior and State

- The Issues
  - Today CIM is an Information model
  - CIM Information plus model behavior = manageability model
- Objectives
  - Allow states and state control on CIM Classes
  - Define inter-object Actions
  - Define state transitions that that invoke actions

Growth of the Information model to a Management model

Managed Services
From Information model
To information and behavior model



Management Services

Managed Services Model (tomorrow)

Manageability Objects (Today)

Managed Services Model

Manageability Model

Requirements for Behavioral Control

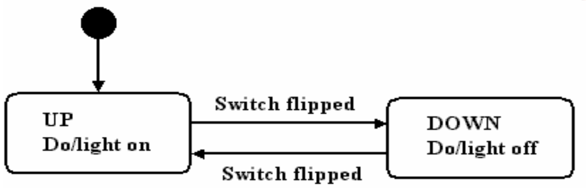
- Define state for CIM Objects
- Define state transitions so that object owners can control state changes
- Define inputs that can control states
- Define Actions that affect other parts of the model
- Provide concepts for hiearchial aggregation and disaggregation of state

A Very Simple Example

A Light Switch Example

- Two states
- One flip switch for control

- Transition Diagram



- State Table

State / Input	Down	Up
Flip Switch	Up/Do Light on	Down/Do Light off

Example (Cont)

```
light
{
        String instanceID;

          [valuemap(“0”, “1”),
          values(“on”, “off”)]
          Uint16 state = 0;

          Uint32 flip();
}
```

TODAY

•Not clear what is a state variable

•Model does not define relation between method and state property.

```
light
{
        String instanceID;

          [State(pointer),
          valuemap(“0”, “1”),
          values(“on”, “off”)]
          Uint16 state = 0;

          Uint32 flip();
}
```

State Transition Matris

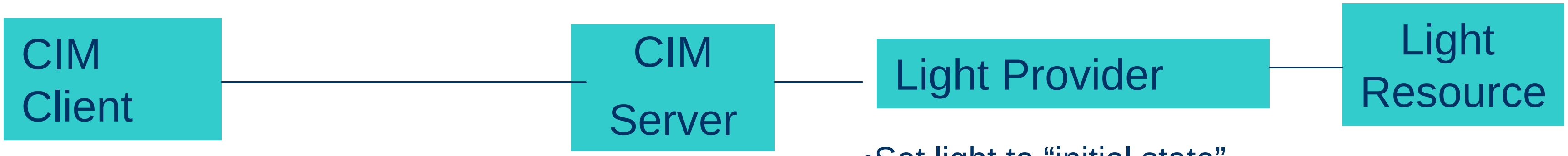
State / Input	Down	Up
Flip Switch	Up/Do Light on	Down/Do Light off

With Behavior Control

•State clearly defined as state property and associated with a particular transition matrix

•Clear behavior relation between method and state property

Example (cont)



CIM Client

CIM Server

Light Provider

Light Resource

- Query state of light instances
- Controls light with “flip” method

- Set light to “initial state”
- Accept “flip” method and control Light resource in accord with input.
- Respond to flip with “good” response if state changed or “error” if it did not.
- Respond to instance requests

Objectives of the Working Group

- Today CIM is an information model
- It does not allow managing behavior
  - Of objects
  - Between objects
- Objective
  - Define mechanisms that would allow behavior CIM objects and between objects to be defined.

Characteristics of a State Model

- Based on OMG UML StateDiagrams
- Able to generate CIM state definitions directly from UML tools

UML State Diagrams

- Hierarchical State Model
  - Hierarchical States (substate model)
  - State Transitions
- Based on event processing architecture
- Features
  - Guards
  - Entry and exit actions
  - Orthogonal Regions - orthogonal regions detect the same events and respond to them “simultaneously”

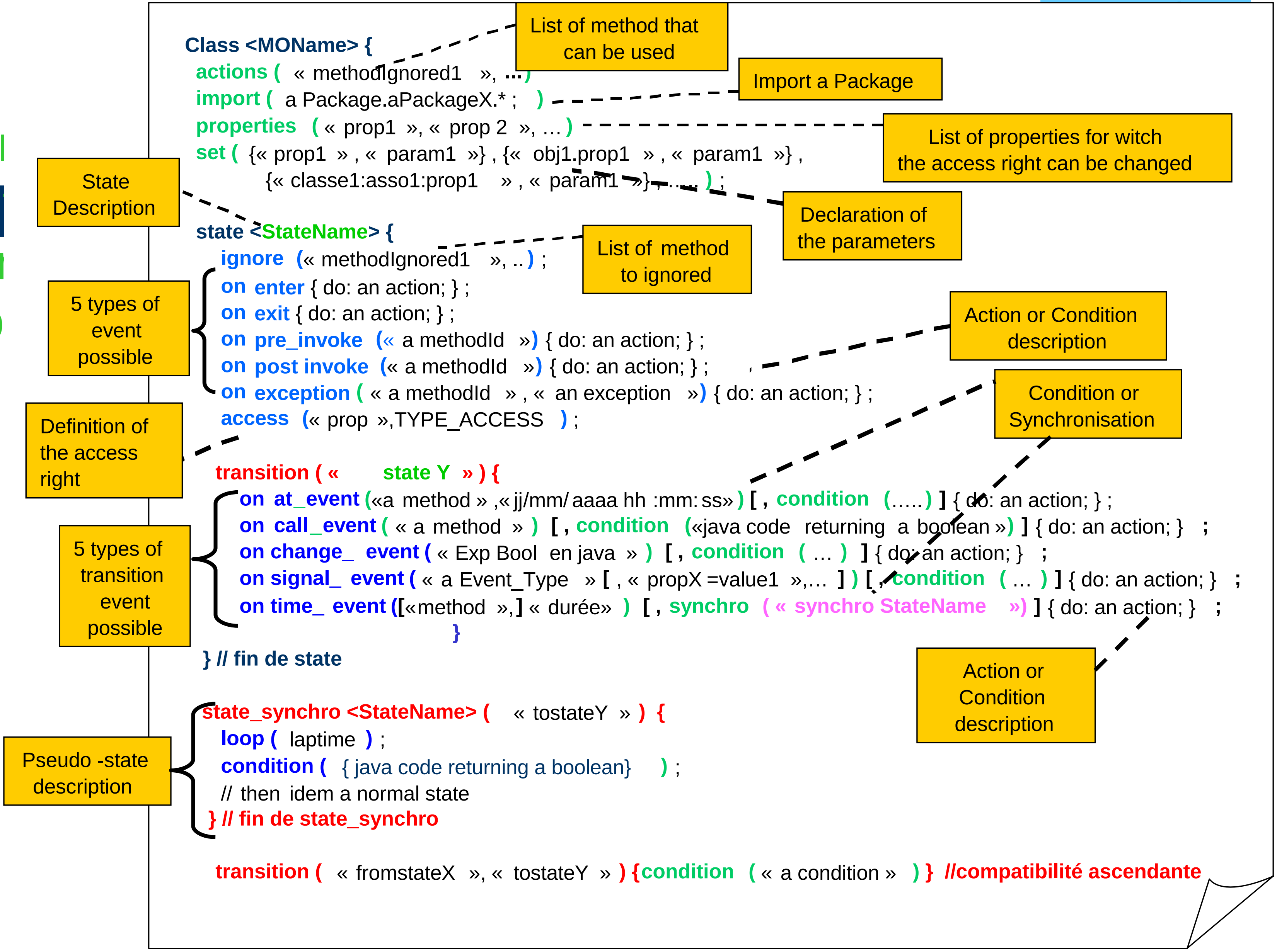
Alternate definitions for State Transitions

- Language based Definition
- Extending the CIM MetaModel to include State, Transitions, Actions concepts
  - UML has an existing meta-model as a starting point
- Model State as instances of newly defined classes

Over

char

no



```
Class <MOName> {
  actions (  « methodIgnored1   », ...)
  import (  a Package.aPackageX.* ;  )
  properties  ( « prop1 », « prop 2 », ... )
  set ( {« prop1 » , «  param1  »} , {«  obj1.prop1  » , «  param1  »} ,
        {« classe1:asso1:prop1   » , «  param1  »} , ..... ) ;

  state <StateName> {
    ignore  (« methodIgnored1   », ..) ;
    on enter { do: an action; } ;
    on exit { do: an action; } ;
    on pre_invoke  (« a methodId  ») { do: an action; } ;
    on post invoke  (« a methodId  ») { do: an action; } ;
    on exception ( « a methodId  » , «  an exception  ») { do: an action; } ;
    access  (« prop »,TYPE_ACCESS  ) ;

    transition ( «      state Y  » ) {
      on at_event («a method » ,«jj/mm/ aaaa hh :mm: ss») [ , condition  (.....) ] { do: an action; } ;
      on call_event ( « a method » ) [ , condition  («java code returning a boolean ») ] { do: an action; }  ;
      on change_ event ( « Exp Bool en java » ) [ , condition  ( ... ) ] { do: an action; }  ;
      on signal_ event ( « a Event_Type  » [ , « propX =value1  »,... ] ) [ , condition  ( ... ) ] { do: an action; }  ;
      on time_ event ([«method  »,] « durée» ) [ , synchro  ( « synchro StateName  ») ] { do: an action; }  ;
                        }
  } // fin de state

  state_synchro <StateName> (  « tostateY  » ) {
    loop (  laptime ) ;
    condition (  { java code returning a boolean}   ) ;
    // then idem a normal state
  } // fin de state_synchro

  transition (  « fromstateX  », «  tostateY  » ) {condition  ( « a condition »  ) }  //compatibilité ascendante
```

List of method that can be used

Import a Package

List of properties for witch the access right can be changed

Declaration of the parameters

State Description

List of method to ignored

5 types of event possible

Action or Condition description

Condition or Synchronisation

Definition of the access right

5 types of transition event possible

Action or Condition description

Pseudo -state description

Questions?