CIM Overview

Andrea Westerinen

Agenda – Monday, Sept 20th

- 9-9:15am, Introductions (Tom Roney)
- 9:15-10:15am, DMTF Executive Overview (Troy Biegger)
- 10:30am-Noon, CIM Overview and Grid Service Example (Andrea Westerinen)
- 1-1:45pm, Application Management and Behavior and State (Karl Schopmeyer)
- 1:45-3pm, WBEM Architecture and XML Renderings (Jim Davis)
- 3:15-3:45pm, WBEM Open Source Overview (Jim Davis)
- 3:45-4:30pm, Pegasus and WBEM Services Overviews (Karl Schopmeyer and Jim Davis)
- 4:30-5pm, Q&A

Agenda – Wednesday, Sept 22nd CGS Sessions

- 11am-12:30pm, Introductions (Tom Roney) + DMTF Executive Overview (Troy Biegger) + CIM Introduction (Andrea Westerinen)
- 3:30-5pm, CIM Overview and Grid Service Example (Andrea Westerinen) + Application Management and Behavior and State (Karl Schopmeyer)
- 7:30-9pm, WBEM Architecture, Open Source and XML Renderings (Jim Davis and Karl Schopmeyer)

Differing Aspects of a Model

- Two very different aspects of a model exist – Semantics and rendering
  - Each has their own requirements and restrictions
- Semantics -> Rendering
  - The model (CIM) dictates content and concepts / Ideally have one model
  - Language constructs and rules dictate the rendering / Multiple renderings are possible (from abstract UML to specific XML Schema)

Modeling Considerations

- Scope and coverage
- Modeling concepts and principles
- Using the model (And an example)

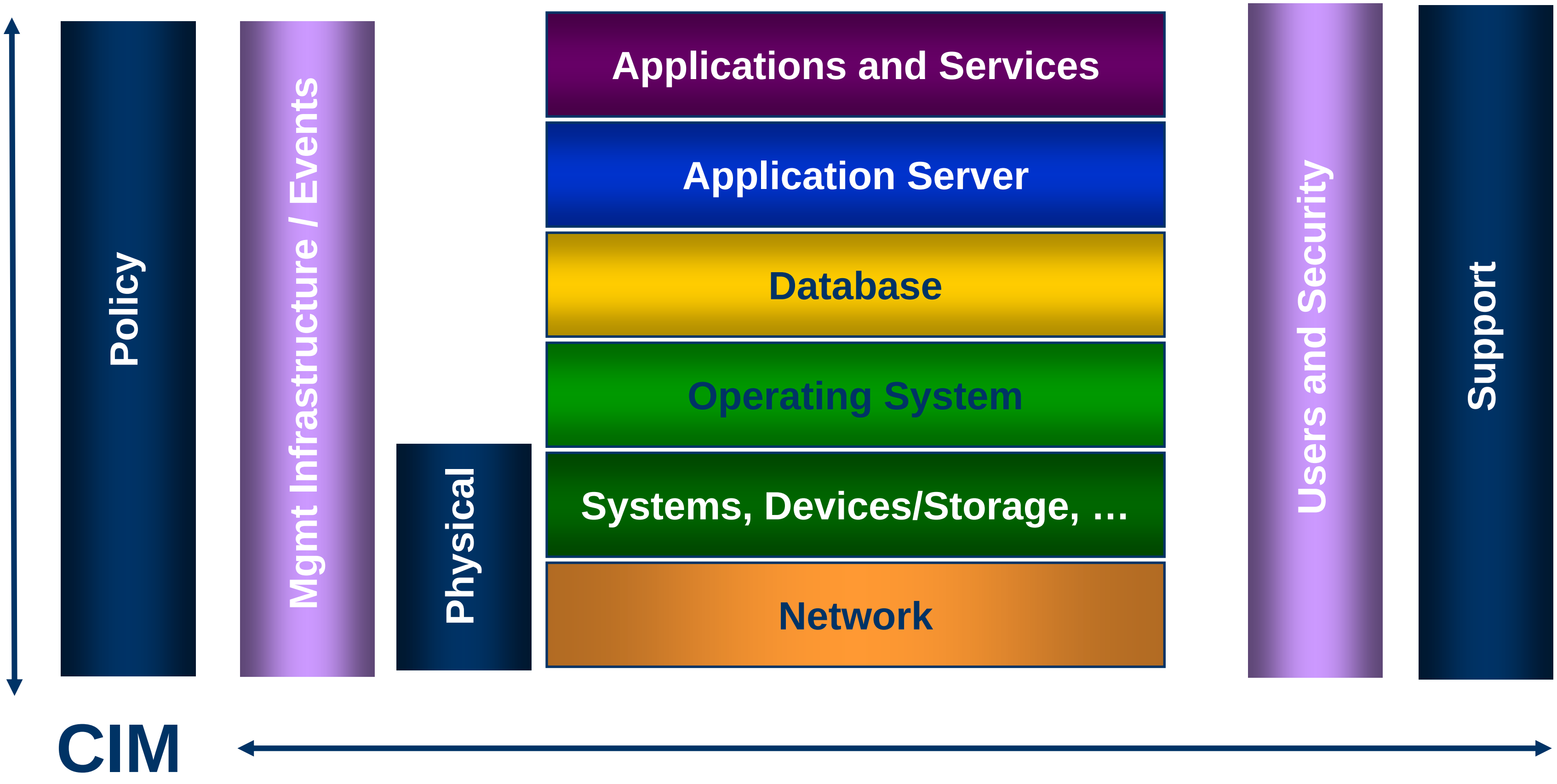
Scope – The Environment AND the Element

- CIM’s scope addresses the "big picture“, but implementation can be limited to a single element
  - Allows dive down into specific components when necessary
  - Example: 20 second access to critical data - Is the problem in the server, the network, the storage or all three?
  - To answer, need element details, and information on the interactions between the elements and business priorities
- Configurations span many elements, to accomplish business goal
  - Desirable for all the elements to understand the "larger" business goals and how they fit into accomplishing these goals
  - Ideally, equipment understands the same config commands
  - Example: Failing over from LA to Chicago

CIM’s Coverage (1)

- Configuration and/or general management data (what is and what is desired)
  - For example, supporting root cause analysis
- Relationships
  - Usage, component, …
  - General abstractions but specific implementations
- Design for evolution and extension (std + proprietary)
- Not only about data, also about operations
  - Domain-specific operations with parameters (ex: CreateOrModifyStoragePool)
- Fits all the pieces together in a single conceptual model

CIM’s Coverage (2)



Policy

Mgmt Infrastructure / Events

Physical

Applications and Services

Application Server

Database

Operating System

Systems, Devices/Storage, …

Network

Users and Security

Support

CIM

Modeling Goals

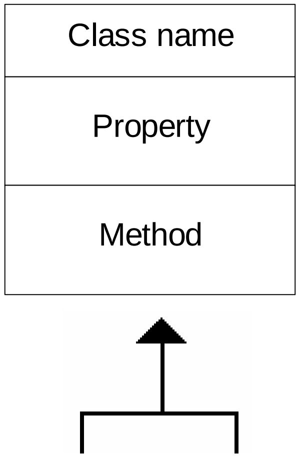
- Predictability
  - Once the model is learned, the location of specific data is maintained and therefore "predictable"
- “Stable” semantics that can be specialized and extended
- Reuse of the model versus redefinition

OO Concepts

- Abstraction (Determination of “essential” characteristics that distinguish and define an object’s conceptual boundaries)
- Modularity (Decomposition of concepts into discrete units)
- Encapsulation (Compartmentalization of structure and behavior; Separation of abstraction and implementation)
- Hierarchy (Ordering of abstractions)

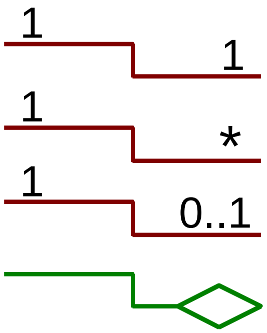
CIM’s Elements

- Classes – Collection of definitions of state, behavior, and identity
  - Properties
  - Methods
- Objects – Instances of a class
- Class hierarchy – Subclassing
- Associations - Relationships
  - Dependency
  - Identity
  - Aggregation
  - Composition
  - And others

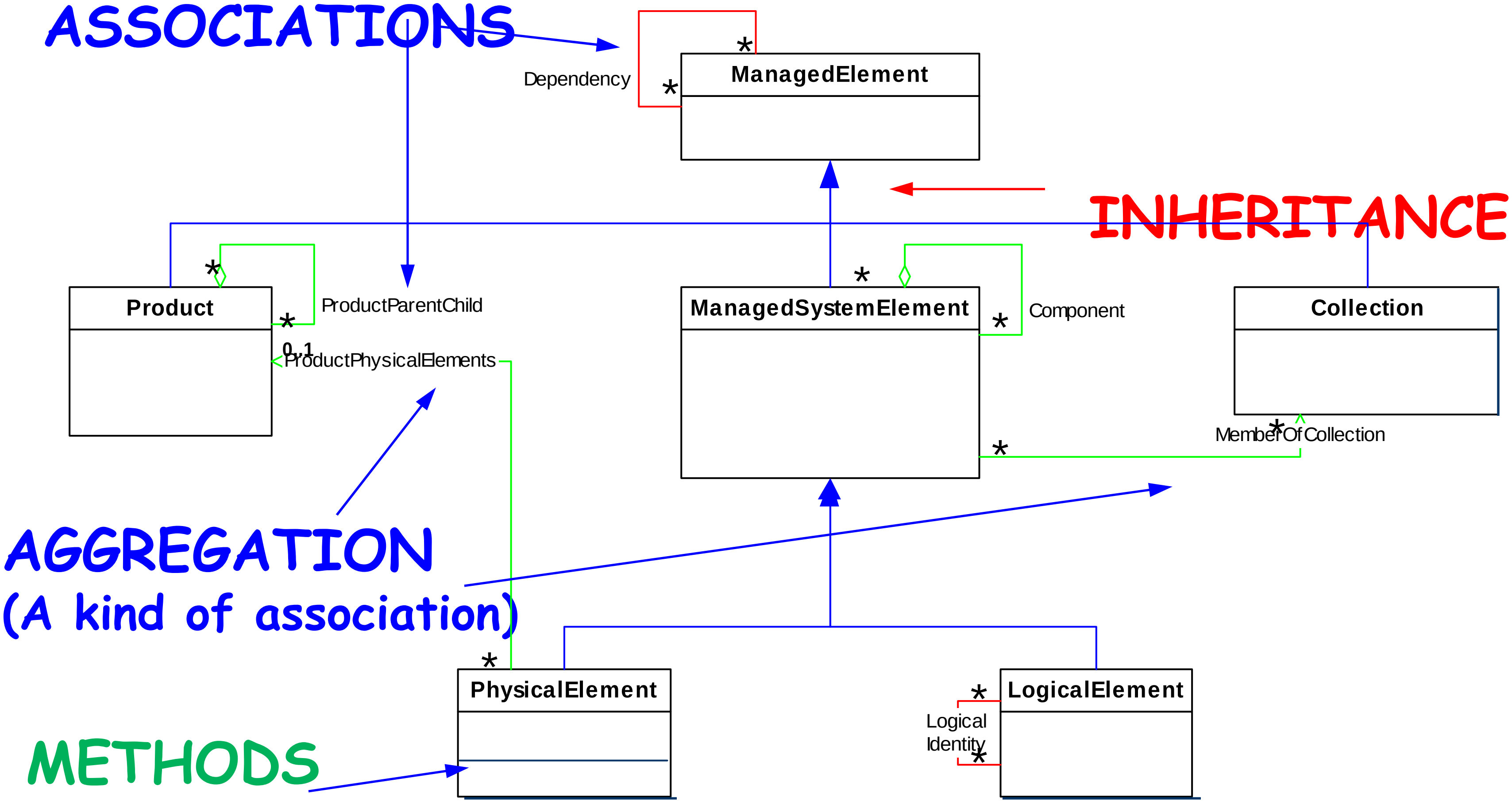

Class name

Property

Method



CIM’s UML



ASSOCIATIONS

INHERITANCE

AGGREGATION
(A kind of association)

METHODS

CIM’s MOF (An Abstract Rendering, Just One of the Possible Renderings)

```
    [Abstract, Description (
     "An abstraction or emulation of a hardware entity, that may "
     "or may not be Realized in physical hardware. ... ") ]
class CIM_LogicalDevice : CIM_LogicalElement
{
. . .
        [Key, MaxLen (64), Description (
         "An address or other identifying information to uniquely "
         "name the LogicalDevice.") ]
   string DeviceID;
        [Description (
         "Boolean indicating that the Device can be power "
         "managed. ...") ]
   boolean PowerManagementSupported;
        [Description (
         "Requests that the LogicalDevice be enabled (\"Enabled\" "
         "input parameter = TRUE) or disabled (= FALSE). ...)" ]
   uint32 EnableDevice([IN] boolean Enabled);
. . .
};
```

Qualifiers

Class Name and Inheritance

Properties

Methods

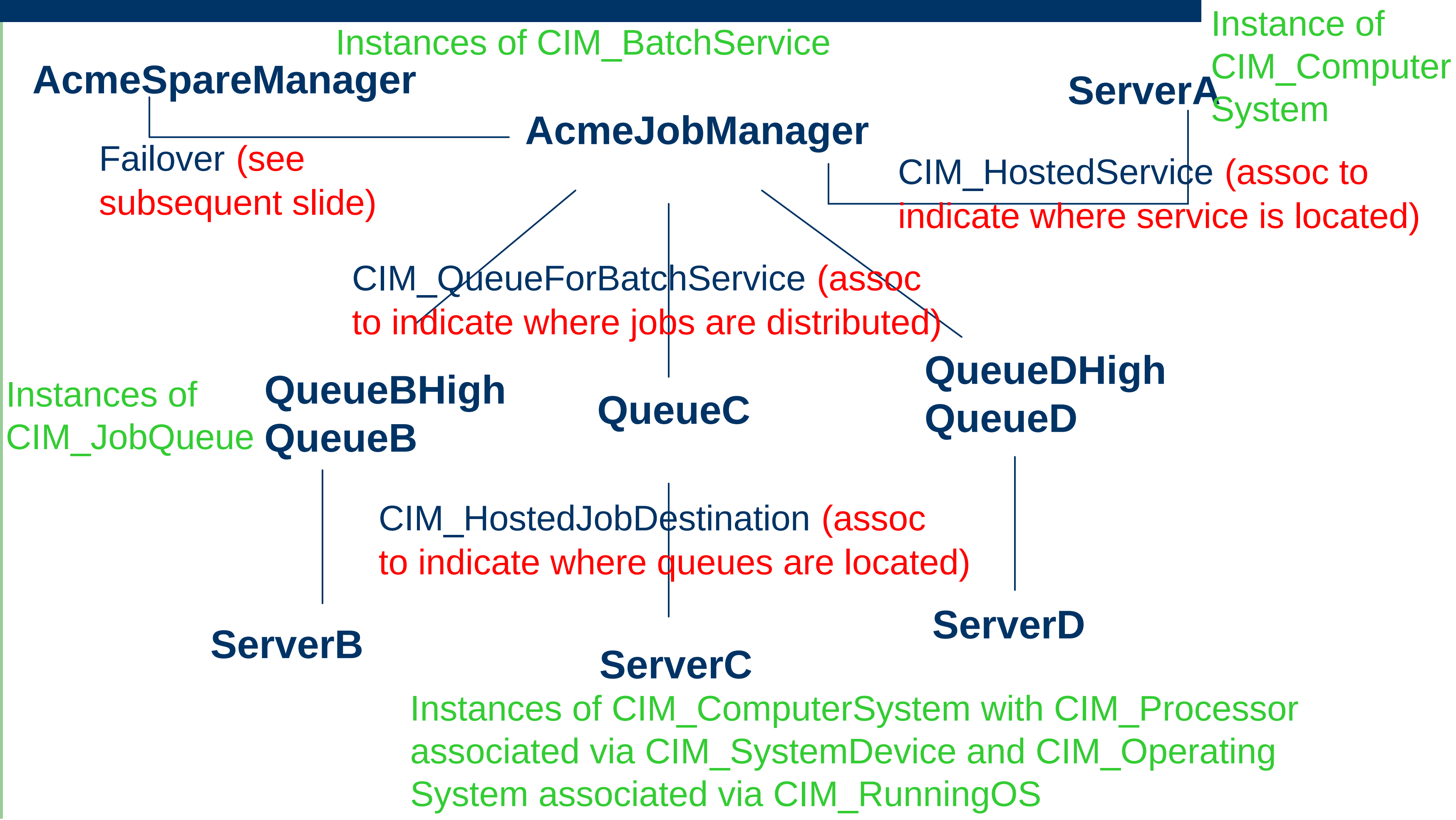
CIM’s Structure – Core and Common Models

- Infrastructure Specification
  - “Meta”-model, high level concepts and language definitions
- “Core” and “Common” Models
  - Core Model contains info applicable to all management domains
  - Common Models address specific domains - Systems, Devices, Applications, Networks, Users, ...
    - Subclass from the Core Model
    - Models overlap and cross-reference
  - Vendor extensions encouraged

Using the CIM Schema

- NEVER … “What class(es) do I need?”
- ALWAYS … “What is being managed and modeled?”
  - Who (Users and Security), What (Physical and Logical Elements), Where (Location), When (aspects of time), How (Services and Service Access Points) and Why (ROI !)
  - Do any of the core or common models match the design points?
  - Examine the CIM inheritance tree to find matching concepts / Read profiles or the MOF for details
  - Iterate based on the use cases, data flow and what is found in CIM

CIM Grid Service Example



Instances of CIM_BatchService

AcmeSpareManager

AcmeJobManager

Failover (see subsequent slide)

ServerA

Instance of CIM_Computer System

CIM_HostedService (assoc to indicate where service is located)

CIM_QueueForBatchService (assoc to indicate where jobs are distributed)

Instances of CIM_JobQueue

QueueBHigh

QueueB

QueueC

QueueDHigh

QueueD

CIM_HostedJobDestination (assoc to indicate where queues are located)

ServerB

ServerC

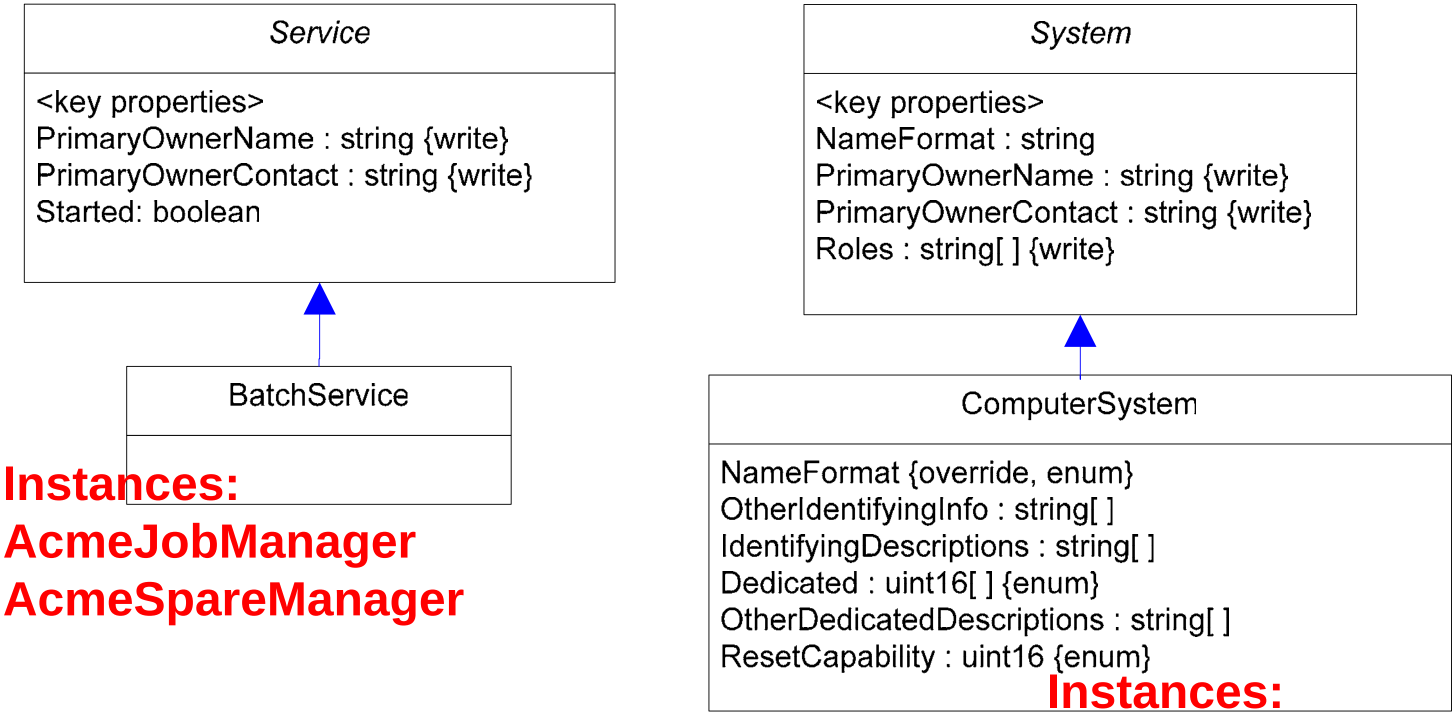
ServerD

Instances of CIM_ComputerSystem with CIM_Processor associated via CIM_SystemDevice and CIM_Operating System associated via CIM_RunningOS

Example – The Job Managers

- "AcmeJobManager" is an instance of BatchService
  - “Submit job" method is part of its functional/business interface, and not its management interface
  - So, BatchService works as defined
- "ServerA" hosts the AcmeJobManager
  - Is an instance of ComputerSystem
  - Used to manage the status of the system and the service
- "RedundantJobManagers” is an instance of a RedundancySet
  - For failover of the AcmeJobManager
  - Associated with the "AcmeSpareManager" (idea that Acme is cheap and only has 1 spare for all its job managers across the Internet)

Service and System Subclasses of EnabledLogicalElement



Service

<key properties>
PrimaryOwnerName : string {write}
PrimaryOwnerContact : string {write}
Started: boolean

BatchService

Instances:
AcmeJobManager
AcmeSpareManager

System

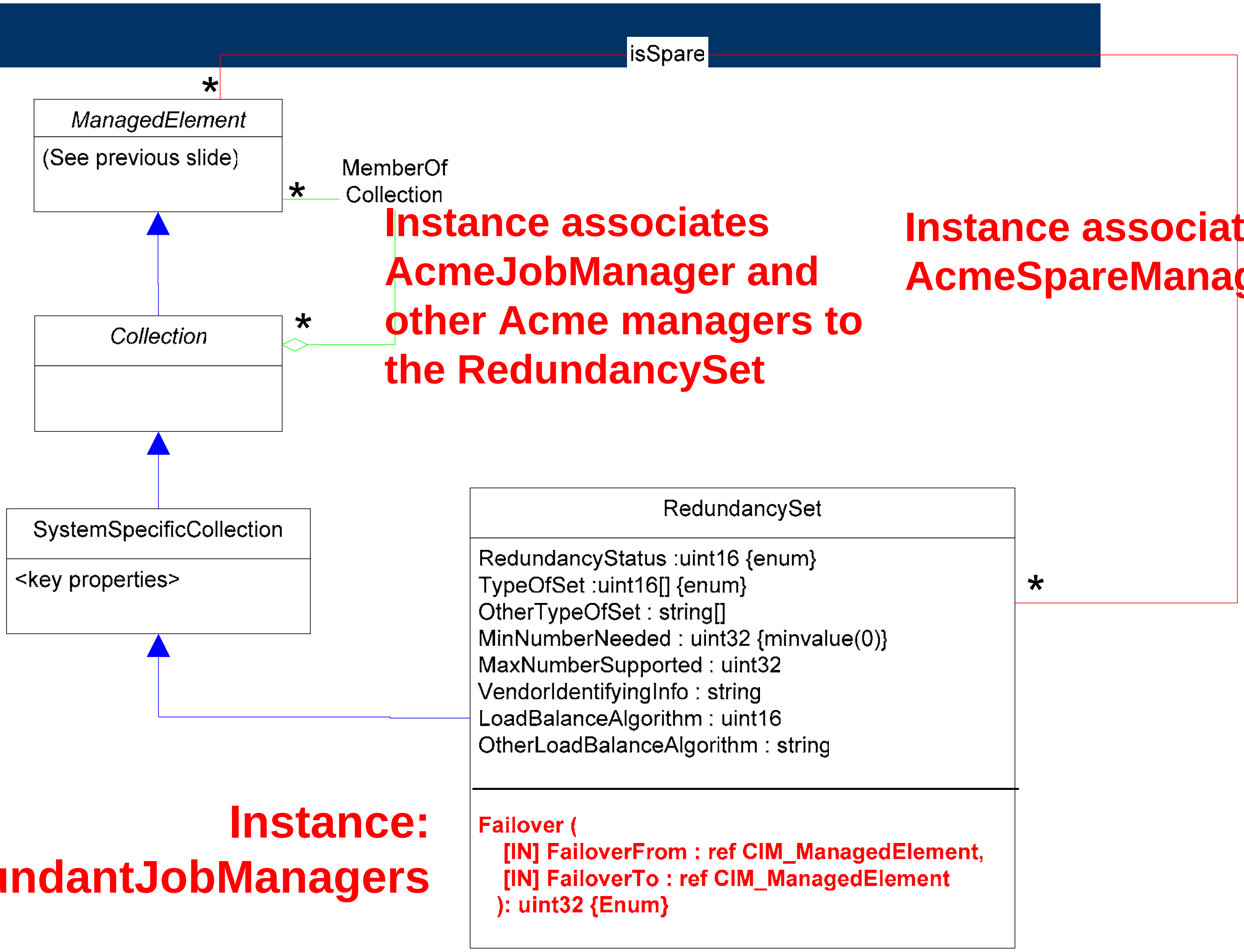
<key properties>
NameFormat : string
PrimaryOwnerName : string {write}
PrimaryOwnerContact : string {write}
Roles : string[ ] {write}

ComputerSystem

NameFormat {override, enum}
OtherIdentifyingInfo : string[ ]
IdentifyingDescriptions : string[ ]
Dedicated : uint16[ ] {enum}
OtherDedicatedDescriptions : string[ ]
ResetCapability : uint16 {enum}

Instances:
ServerA, ServerB,
ServerC, ServerD

CIM Redundancy Modeling



ManagedElement

(See previous slide)

Collection

SystemSpecificCollection

<key properties>

RedundancySet

RedundancyStatus :uint16 {enum}
TypeOfSet :uint16[] {enum}
OtherTypeOfSet : string[]
MinNumberNeeded : uint32 {minvalue(0)}
MaxNumberSupported : uint32
VendorIdentifyingInfo : string
LoadBalanceAlgorithm : uint16
OtherLoadBalanceAlgorithm : string

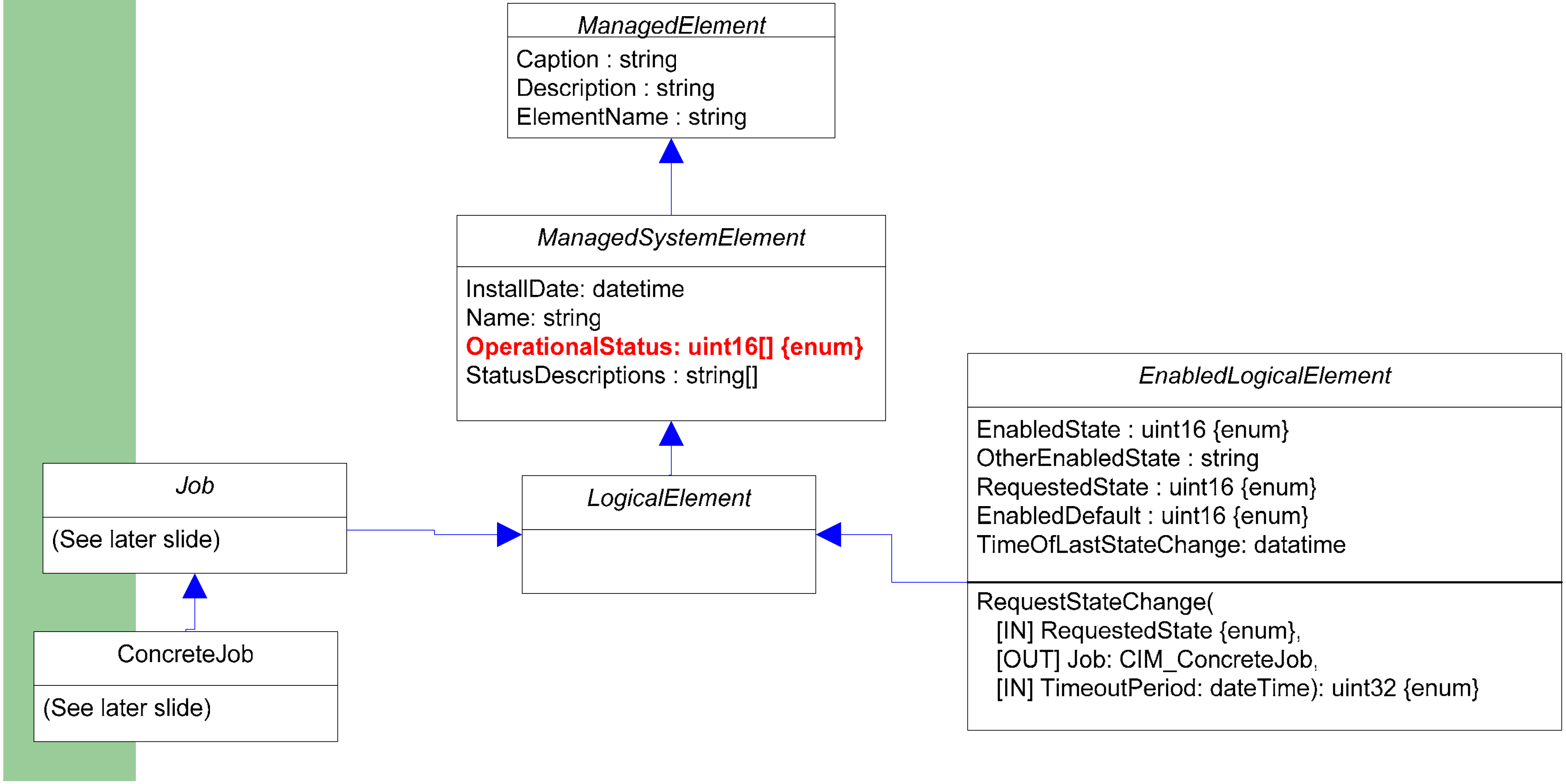
Failover (
[IN] FailoverFrom : ref CIM_ManagedElement,
[IN] FailoverTo : ref CIM_ManagedElement
): uint32 {Enum}

Instance associates AcmeJobManager and other Acme managers to the RedundancySet

Instance associates AcmeSpareManager

Instance: RedundantJobManagers

Background: CIM Class Derivation



ManagedElement

Caption : string
Description : string
ElementName : string

ManagedSystemElement

InstallDate: datetime
Name: string
OperationalStatus: uint16[] {enum}
StatusDescriptions : string[]

EnabledLogicalElement

EnabledState : uint16 {enum}
OtherEnabledState : string
RequestedState : uint16 {enum}
EnabledDefault : uint16 {enum}
TimeOfLastStateChange: datatime

RequestStateChange(
[IN] RequestedState {enum},
[OUT] Job: CIM_ConcreteJob,
[IN] TimeoutPeriod: dateTime): uint32 {enum}

Job

(See later slide)

LogicalElement

ConcreteJob

(See later slide)

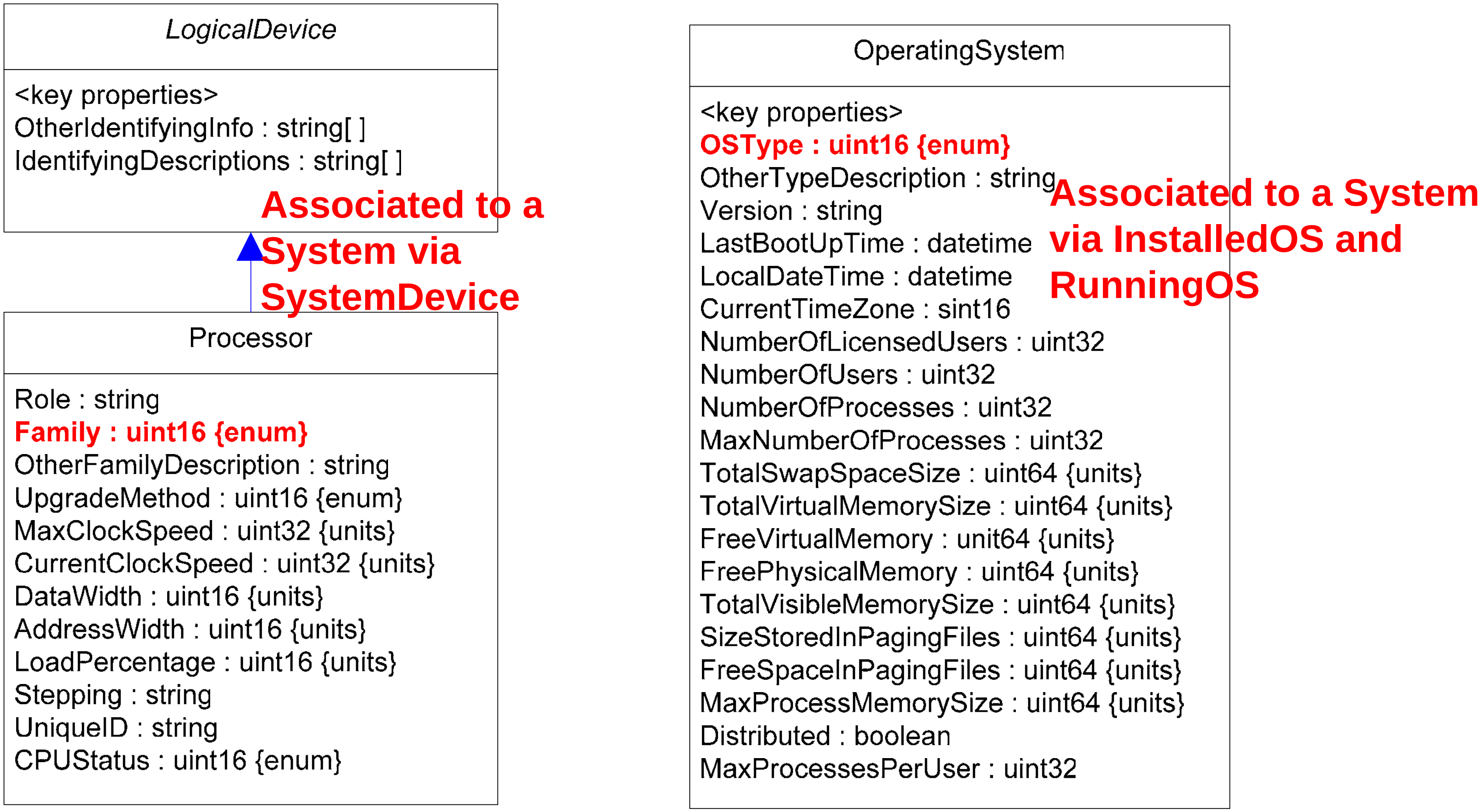
Example – The Execution Servers

- "ServerB", "ServerC" and "ServerD" are instances of ComputerSystem
  - With SystemDevice associations to 4 instances of Processor (on B and D) and 2 instances of Processor (on C) / Servers B and C are running "Linux", while D runs "Microsoft Windows Server 2003"
- "QueueBHigh", "QueueB", "QueueC", "QueueDHigh" and "QueueD" are instances of JobQueue on Servers B (2 queues), C (1 queue) and D (2 queues)
  - Each has an associated QueueStatisticalData
- Queues are associated to AcmeJobManager via the QueueForBatchService relationship
  - AcmeJobManager distributes jobs to Servers B, C and D based on their queue backlogs, and OS

Example – Jobs and Notifications

- Jobs are really instances of BatchJob, and are located in a queue using the JobDestinationJobs association
- AcmeJobManager registers for Indications on all submitted Jobs
  - If the Job's Run* properties indicate a time earlier than StartTime (ie, the job was scheduled to run at a specified time, but did not start on or before that time)
  - Then another Indication is raised regarding an SLA violation

More Subclasses of EnabledLogical Element to Manage OS and Processor



LogicalDevice

<key properties>
OtherIdentifyingInfo : string[ ]
IdentifyingDescriptions : string[ ]

Associated to a System via SystemDevice

Processor

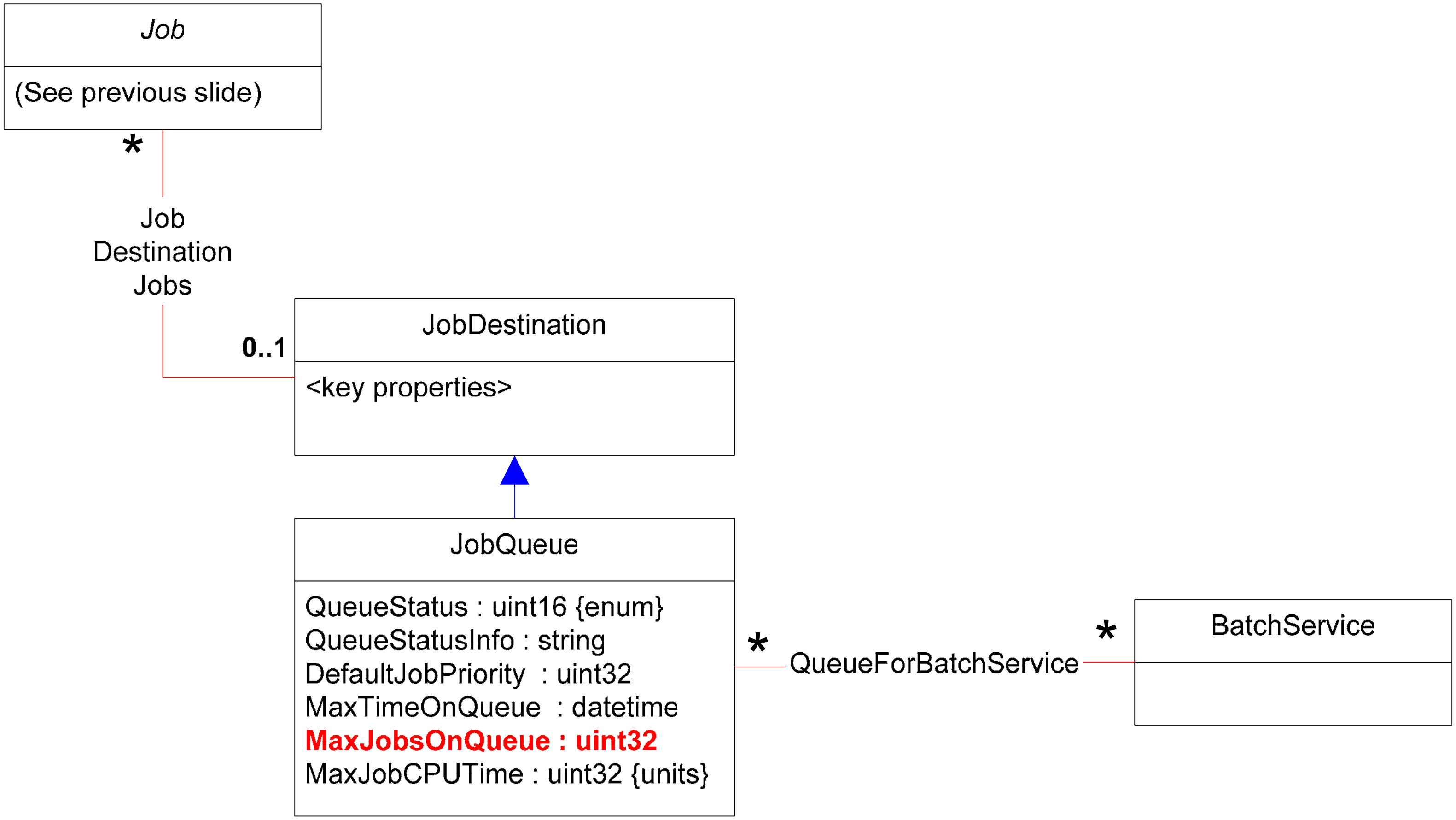
Role : string
Family : uint16 {enum}
OtherFamilyDescription : string
UpgradeMethod : uint16 {enum}
MaxClockSpeed : uint32 {units}
CurrentClockSpeed : uint32 {units}
DataWidth : uint16 {units}
AddressWidth : uint16 {units}
LoadPercentage : uint16 {units}
Stepping : string
UniqueID : string
CPUStatus : uint16 {enum}

OperatingSystem

<key properties>
OSType : uint16 {enum}
OtherTypeDescription : string
Version : string
LastBootUpTime : datetime
LocalDateTime : datetime
CurrentTimeZone : sint16
NumberOfLicensedUsers : uint32
NumberOfUsers : uint32
NumberOfProcesses : uint32
MaxNumberOfProcesses : uint32
TotalSwapSpaceSize : uint64 {units}
TotalVirtualMemorySize : uint64 {units}
FreeVirtualMemory : unit64 {units}
FreePhysicalMemory : uint64 {units}
TotalVisibleMemorySize : uint64 {units}
SizeStoredInPagingFiles : uint64 {units}
FreeSpaceInPagingFiles : uint64 {units}
MaxProcessMemorySize : uint64 {units}
Distributed : boolean
MaxProcessesPerUser : uint32

Associated to a System via InstalledOS and RunningOS

CIM Job Queues



Job

(See previous slide)

Job Destination Jobs

JobDestination

<key properties>

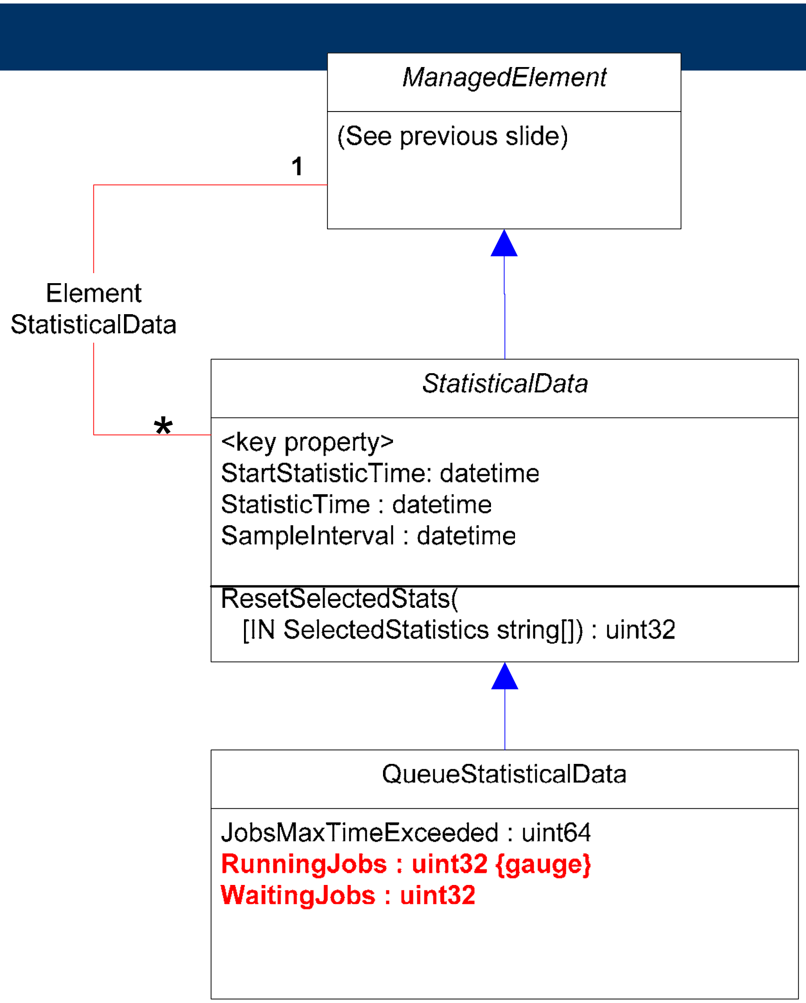
JobQueue

QueueStatus : uint16 {enum}
QueueStatusInfo : string
DefaultJobPriority : uint32
MaxTimeOnQueue : datetime
MaxJobsOnQueue : uint32
MaxJobCPUTime : uint32 {units}

QueueForBatchService

BatchService

CIM Queue Statistics



ManagedElement

(See previous slide)

StatisticalData

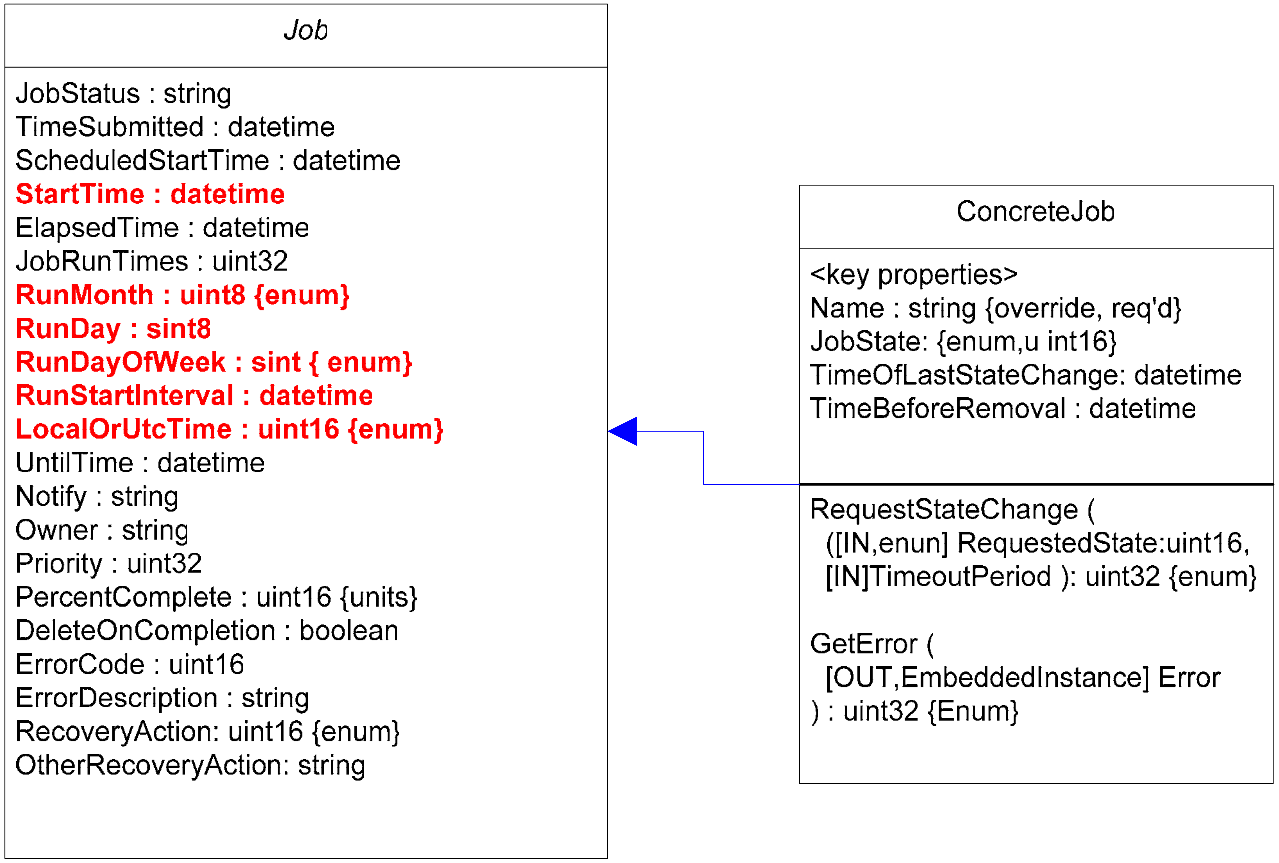
<key property>
StartStatisticTime: datetime
StatisticTime : datetime
SampleInterval : datetime

ResetSelectedStats(
[IN SelectedStatistics string[]) : uint32

QueueStatisticalData

JobsMaxTimeExceeded : uint64
RunningJobs : uint32 {gauge}
WaitingJobs : uint32

CIM Jobs



Job

JobStatus : string
TimeSubmitted : datetime
ScheduledStartTime : datetime
StartTime : datetime
ElapsedTime : datetime
JobRunTimes : uint32
RunMonth : uint8 {enum}
RunDay : sint8
RunDayOfWeek : sint { enum}
RunStartInterval : datetime
LocalOrUtcTime : uint16 {enum}
UntilTime : datetime
Notify : string
Owner : string
Priority : uint32
PercentComplete : uint16 {units}
DeleteOnCompletion : boolean
ErrorCode : uint16
ErrorDescription : string
RecoveryAction: uint16 {enum}
OtherRecoveryAction: string

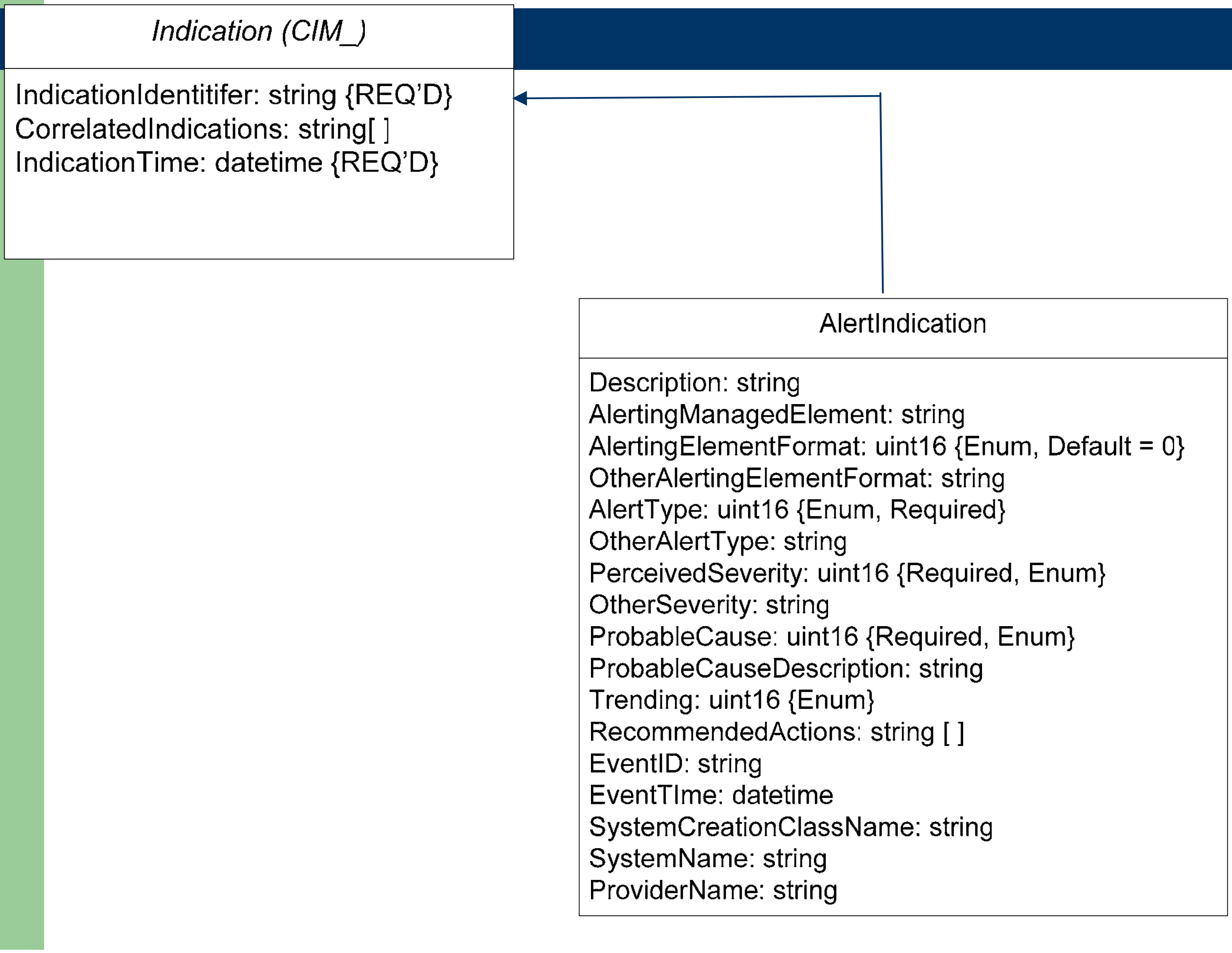
ConcreteJob

<key properties>
Name : string {override, req'd}
JobState: {enum,u int16}
TimeOfLastStateChange: datetime
TimeBeforeRemoval : datetime

RequestStateChange (
([IN,enun] RequestedState:uint16,
[IN]TimeoutPeriod): uint32 {enum}

GetError (
[OUT,EmbeddedInstance] Error
) : uint32 {Enum}

CIM Notifications



Indication (CIM_)

IndicationIdentitifer: string {REQ'D}
CorrelatedIndications: string[ ]
IndicationTime: datetime {REQ'D}

AlertIndication

Description: string
AlertingManagedElement: string
AlertingElementFormat: uint16 {Enum, Default = 0}
OtherAlertingElementFormat: string
AlertType: uint16 {Enum, Required}
OtherAlertType: string
PerceivedSeverity: uint16 {Required, Enum}
OtherSeverity: string
ProbableCause: uint16 {Required, Enum}
ProbableCauseDescription: string
Trending: uint16 {Enum}
RecommendedActions: string [ ]
EventID: string
EventTIme: datetime
SystemCreationClassName: string
SystemName: string
ProviderName: string

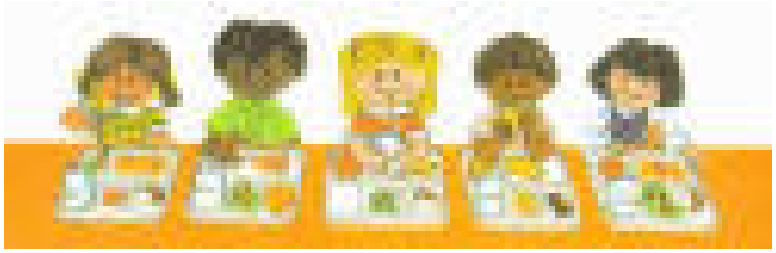
Backup

OO Example - Abstractions

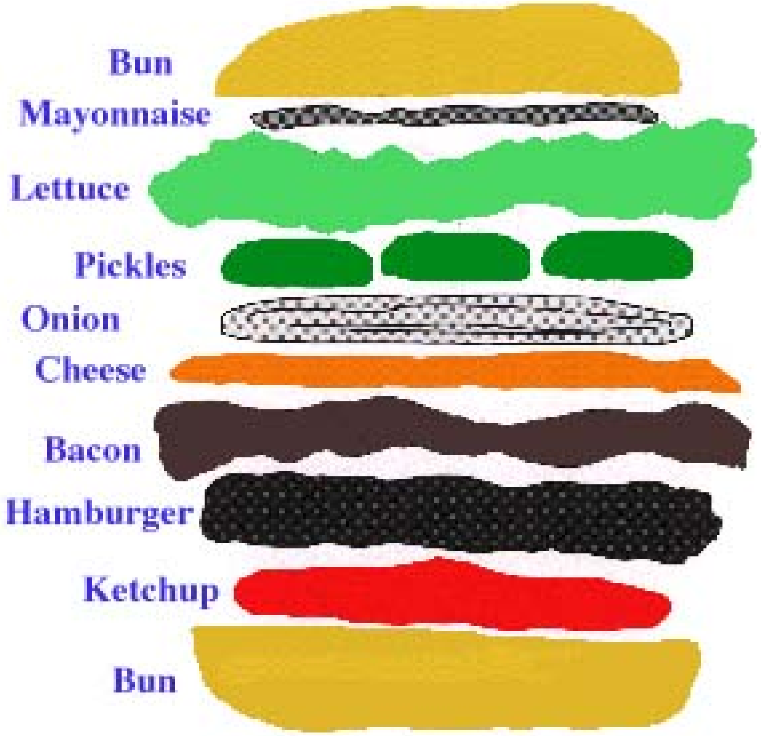
Cheeseburger

Fun to cook!

Good to eat!



OO Example - Modularity

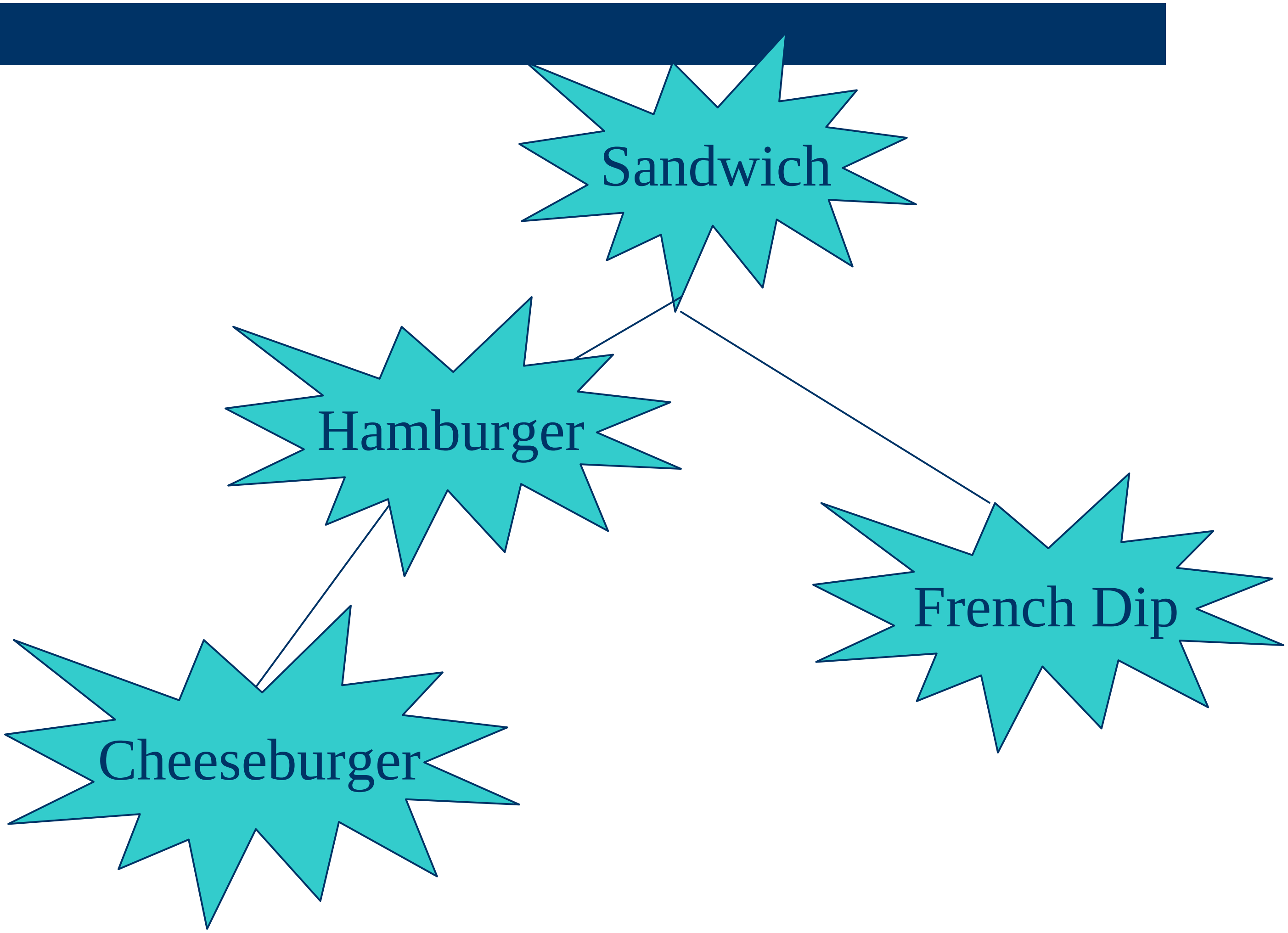


OO Example - Encapsulation

To cook the cheeseburger:
- Is the stove available?
- Are the burners working?
- Are the ingredients available?

To eat the cheeseburger:
- Is it made correctly?
- Is my plate clean or disgusting?

OO Example - Hierarchy


Sandwich
Hamburger
French Dip
Cheeseburger