Summer School on Formal Methods 2007:
Performance Evaluation

From Annotated Software Designs (SPT/MARTE) to Performance Model Formalisms

Murray Woodside
Carleton University
Ottawa, Canada

Carleton UNIVERSITY

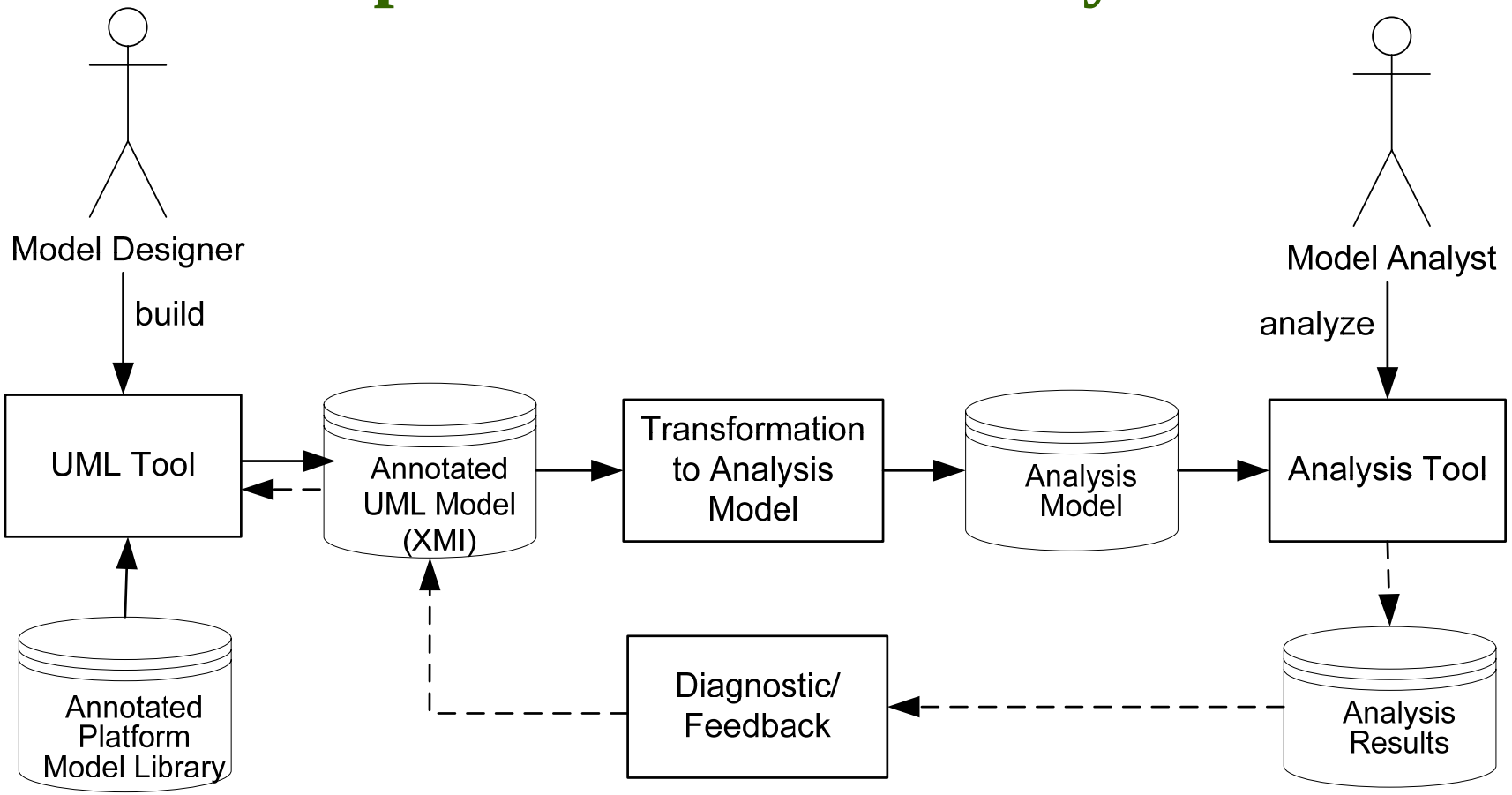
Outline

1. Quick review of UML notation
2. The SPT profile (Schedulability, Performance and Time)
    1. metamodel, concepts, stereotypes and properties, time
3. How UML relates to different target performance models
    1. queueing
    2. layered queueing
    3. petri net
4. MARTE: what is new
5. Examples, annotation details

This lecture includes material originated by many others, including Dorina Petriu, Dorin Petriu, JingXu, and Jose Merseguer.

Carleton UNIVERSITY

Tool Assumptions for Model Analysis

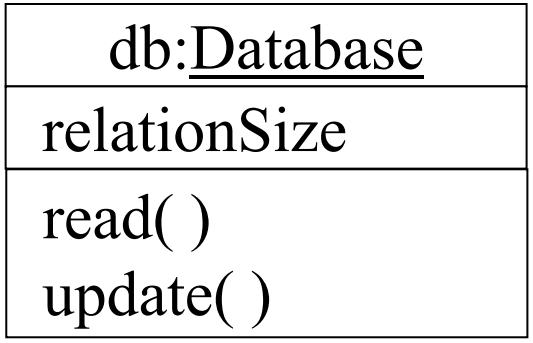


Annotation of real system designs, not just of UML intended to define a performance model

Carleton UNIVERSITY

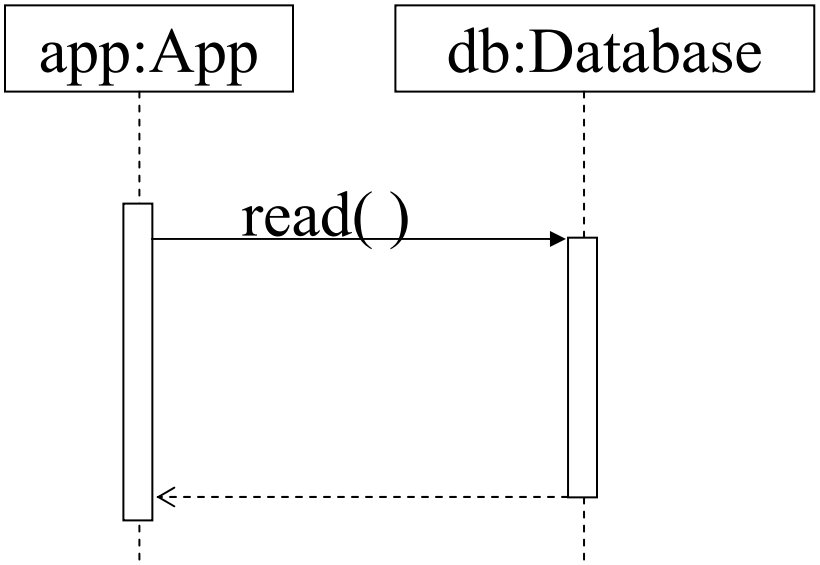
1. UML: quick survey of ideas

- Visual language for design
- Classes of objects
  - operations
  - attributes
  - associations
- Deployment to platforms
- Behaviour
  - State machines
  - Sequence Diagrams
  - Activity Diagrams
- Scenario
  - sequence, including alt and par



Class

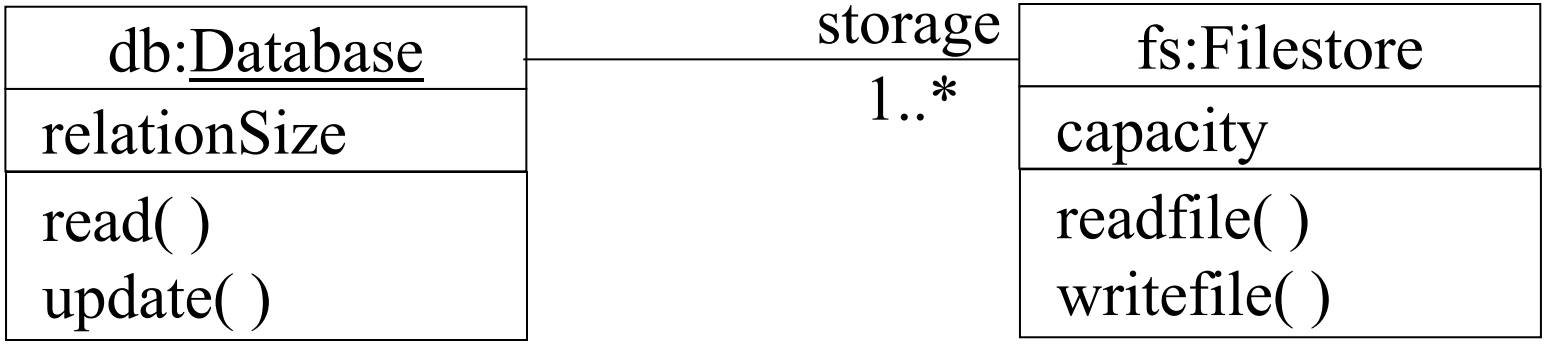
Sequence Diagram



Carleton UNIVERSITY

Class Diagram

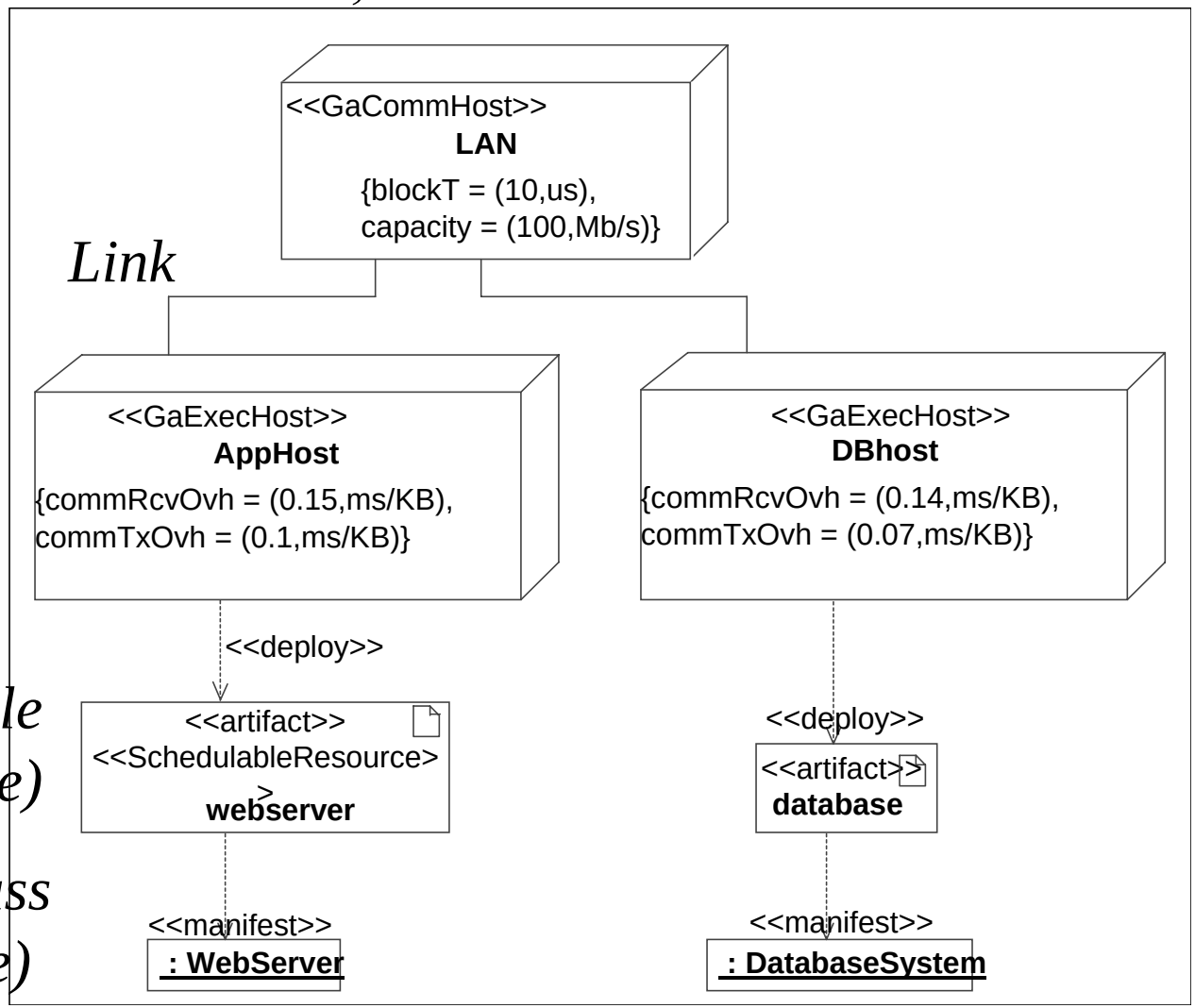
- classes (Database),
  - inheritance specializes a class (SQLDatabase)
  - instances create copies (mydatabase:Database)
- properties = data (mydatabase.relationSize)
- operations = methods (mydatabase.read( ))
- associations provide structuring
  - mydatabase.storage



Carleton UNIVERSITY

Deployment

- crudely represented in UML1, better in UML2
- host node,
- link, artifact



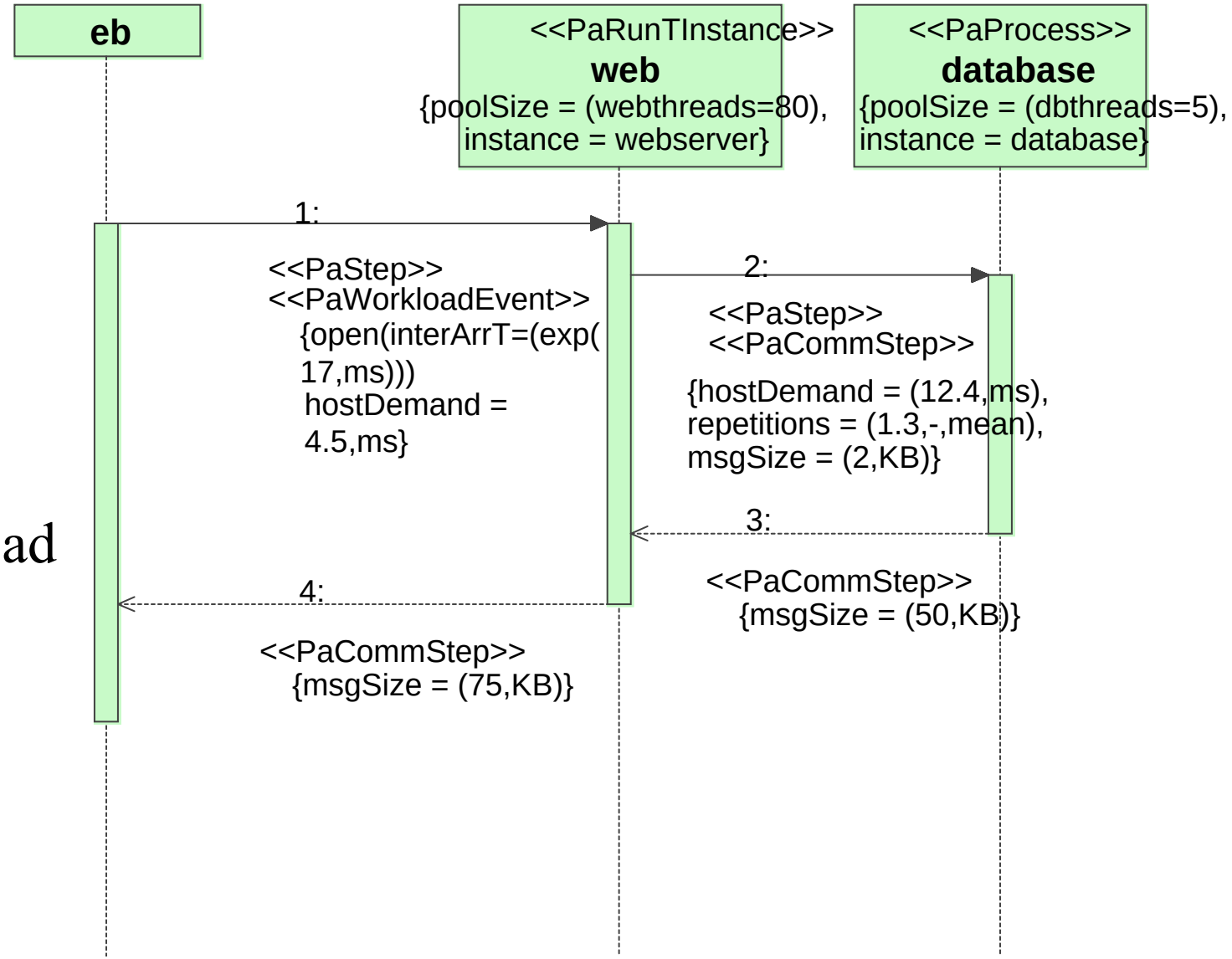
Carleton UNIVERSITY

Sequence or Interaction Diagram: web server

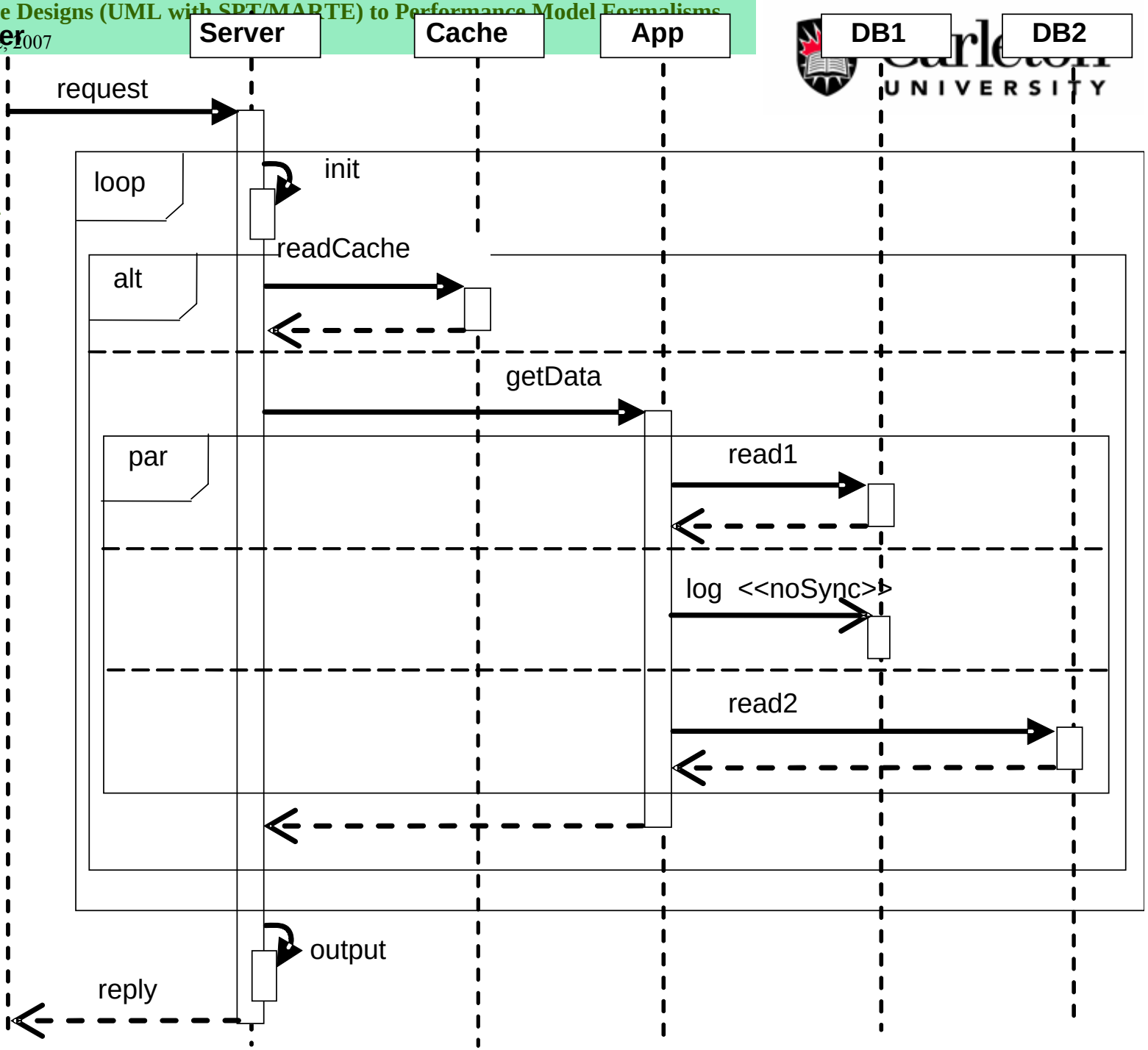
- role,
- lifeline,
- execution

Instance,

- message,
- PaStep,
- PaWorkload

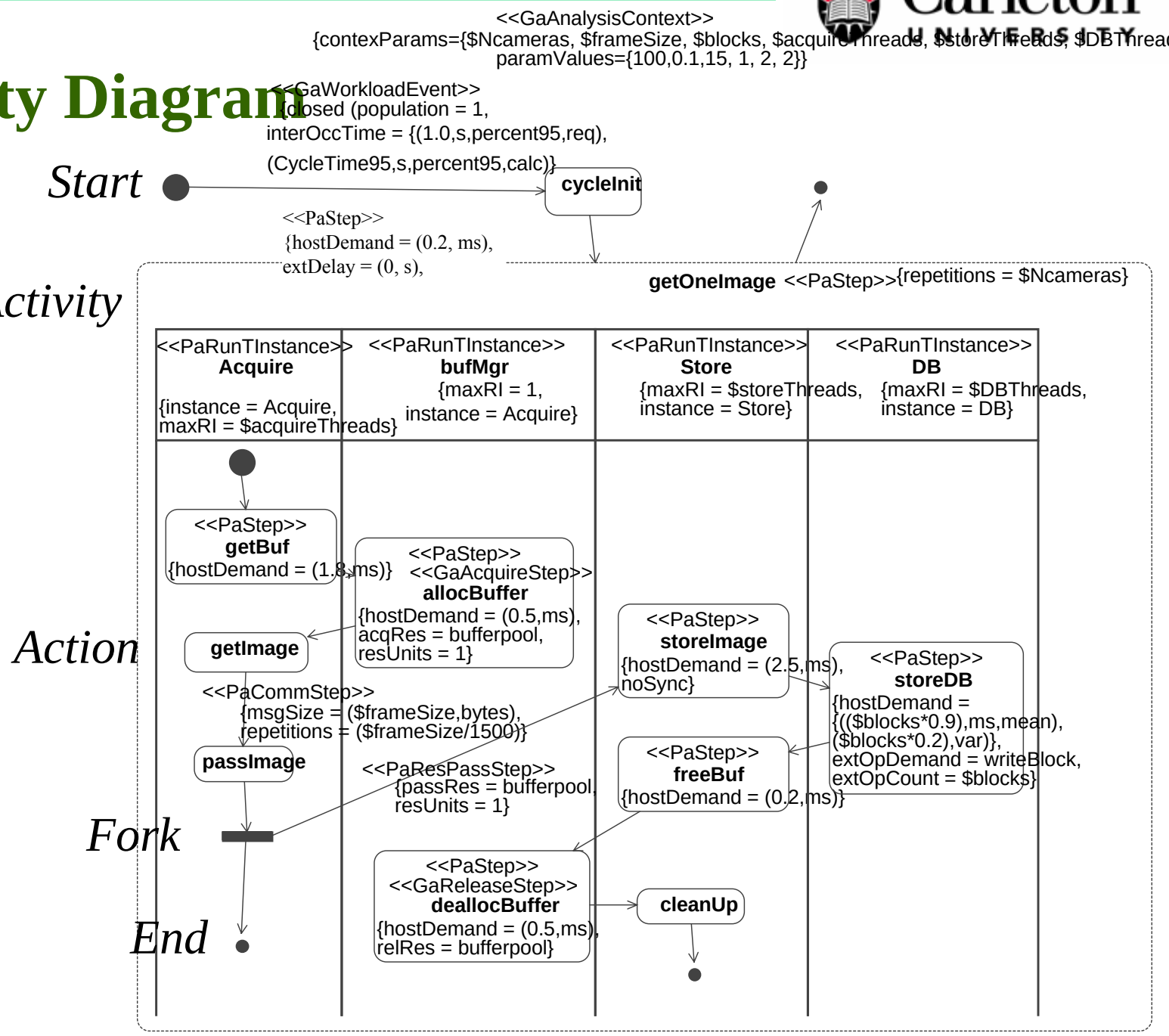


SD summary


User
Server
Cache
App
DB1
DB2
request
loop
init
readCache
alt
getData
par
read1
log <<noSync>>
read2
output
reply

Carleton UNIVERSITY

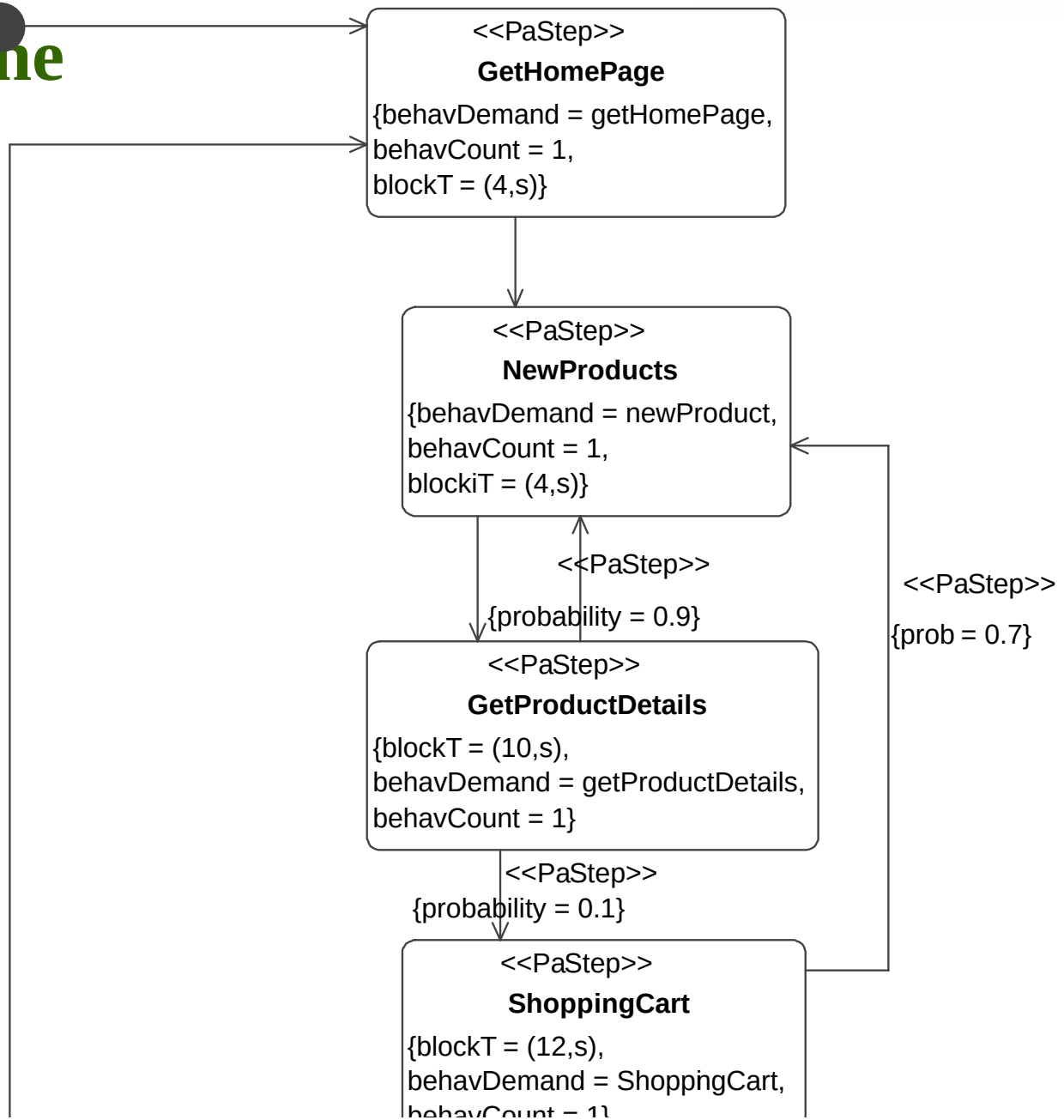
Activity Diagram



Carleton UNIVERSITY

State Machine

- state
- transition
- action on state or transition



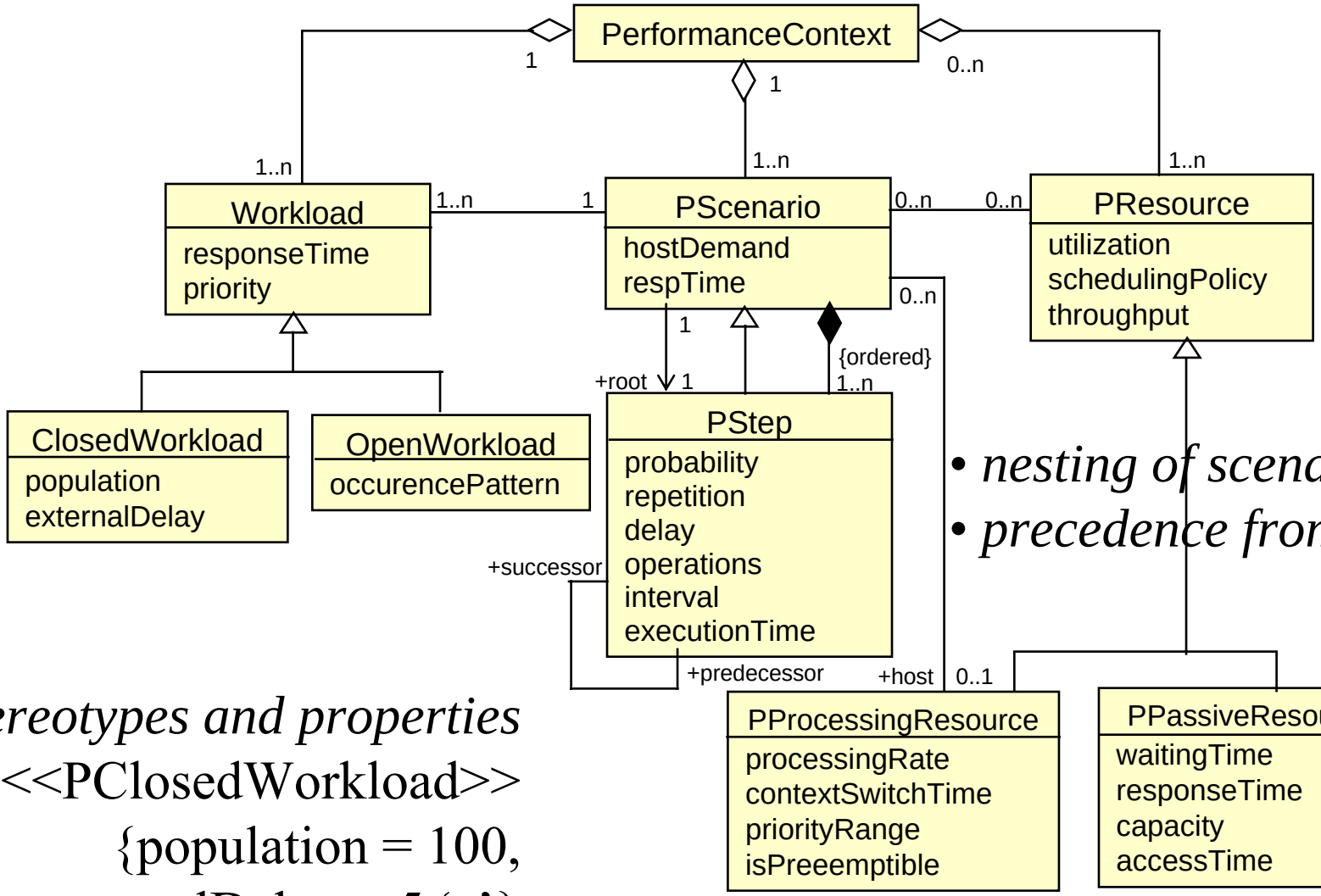
Carleton UNIVERSITY

2. SPT Performance Profile: fundamental concepts

- A Scenario defines a response path through the system, so it's the unit for which performance specifications and predictions are given.
  - however, there is no Scenario stereotype in the Performance Profile to be applied to a UML model element
    - scenario annotations are attached to its first Step instead
  - a Scenario is composed of Steps (which can be simple or composite)
  - a Step stereotype has tags that define performance specifications
    - workload intensity parameters, demands for resource usage, etc.
- Scenarios use the services of Resource instances
  - resource parameters: service policy, multiplicity, operation time
  - resource performance measure: utilization
  - quantitative resource demands given for each step
- Each scenario is executed by a Workload:
  - open workload: requests arriving at in some predetermined pattern
  - closed workload: a fixed number of active or potential users or jobs

Carleton UNIVERSITY

Performance Profile: the domain model



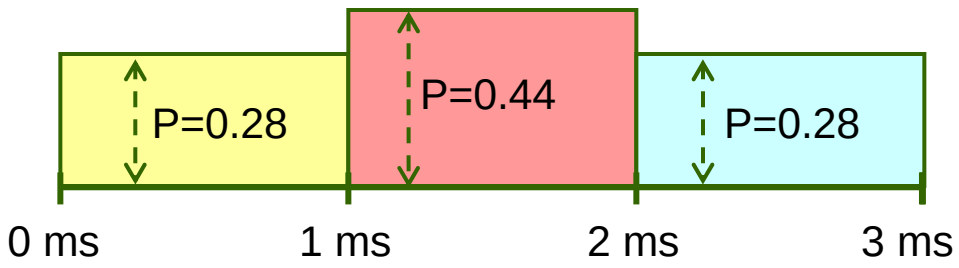
- *nesting of scenarios*
- *precedence from UML*

stereotypes and properties
e.g. <<PClosedWorkload>>
{population = 100,
externalDelay = 5 ‘s’}

Carleton UNIVERSITY

Specifying Time Values

- Time values can be represented by a special Value stereotype «RTtimeValue» in different formats:
  - 12:04 (time of day)
  - 5.3, 'ms' (time interval, unit)
  - 2000/10/27 (date)
  - Wed (day of week)
  - $param, 'ms' (parameterized value, unit)
  - 'poisson', 5.4, 'sec' (time value with a Poisson distribution)
  - 'histogram' 0, 0.28 1, 0.44 2, 0.28, 3, 'ms'



Carleton UNIVERSITY

Specifying Performance Values

- A complex structured string with the following format

<performance-value> ::="(“<kind-of-value> “,“ <modifier> “,“ <time-value>”)”

where:

<kind-of-value> ::= ‘req’ | ‘assm’ | ‘pred’ | ‘msr’

required, assumed, predicted, measured

<modifier> ::= ‘mean’ | ‘sigma’ | ‘kth-mom’ , <Integer> |
‘max’ | ‘percentile’ <Real> | ‘dist’

<time-value> is a time value described by the RTtimeValue type

- A single characteristic may combine more than one performance values:

<PAcharacteristic> ::= <performance-value> [<performance-Value>]*

- Example:

{PAhostDemand = (‘pred’, ‘mean’, (20, ‘ms’)) }

{PArespTime = (‘req’, mean, (1, ‘sec’)) (‘pred’, mean, $RespT) }

required ***predicted => analysis result***

Carleton UNIVERSITY

Specifying Arrival Patterns

- Method for specifying standard arrival pattern values:
  - Bounded: 'bounded', <min-interval>, <max-interval>
  - Bursty: 'bursty', <burst-interval> <max.no.events>
  - Irregular:
    'irregular', <interarrival-time>, [<interarrival-time>]*
  - Periodic: 'periodic', <period> [, <max-deviation>]
  - Unbounded: 'unbounded', <probability-distribution>
- Probability distributions supported:
  - Bernoulli, Binomial, Exponential, Gamma, Geometric, Histogram, Normal, Poisson, Uniform
- What happens when other distributions are needed?
  - tradeoff between
    - flexibility (allow users to introduce their own definitions)
    - simplicity/convenience of expression (i.e., provide pre-packaged definitions).

Carleton UNIVERSITY

Performance Stereotypes (1)

Stereotype	Applies To	Tags	Description
«PAclosedLoad»	Action, ActionExecution, Stimulus, Action, Message, Method…	PArespTime [0..*] PApriority [0..1] PApopulation [0..1] PAextDelay [0..1]	A closed workload
«PAcontext»	Collaboration, CollaborationInstanceSet, ActivityGraph		A performance analysis context
«PAhost»	Classifier, Node, ClassifierRole, Instance, Partition	PAutilization [0..*] PAschdPolicy [0..1] PArate [0..1] PActxtSwT [0..1] PAprioRange [0..1] PApreemptible [0..1] PAthroughput [0..1]	A deferred receive
«PAopenLoad»	Action, ActionExecution, Stimulus, Action, Message, Method…	PArespTime [0..*] PApriority [0..1] PAoccurrence [0..1]	An open workload

Carleton UNIVERSITY

Performance Stereotypes (2)

Stereotype	Applies To	Tags	Description
«PAresource»	Classifier, Node, ClassifierRole, Instance, Partition	PAutilization [0..*] PAschdPolicy [0..1] PAcapacity [0..1] PAmaxTime [0..1] PArespTime [0..1] PAwaitTime [0..1] PAthroughput [0..1]	A passive resource
«PAstep»	Message, ActionState, Stimulus, SubactivityState	PAhostDemand [0..1] PArespTime [0..1] PAprob [0..1] PArep [0..1] PAdelay [0..1] PAextOp [0..1] PAinterval [0..1]	A step in a scenario

Carleton UNIVERSITY

Core Scenario Model (CSM)

Since performance modeling is based on scenarios:

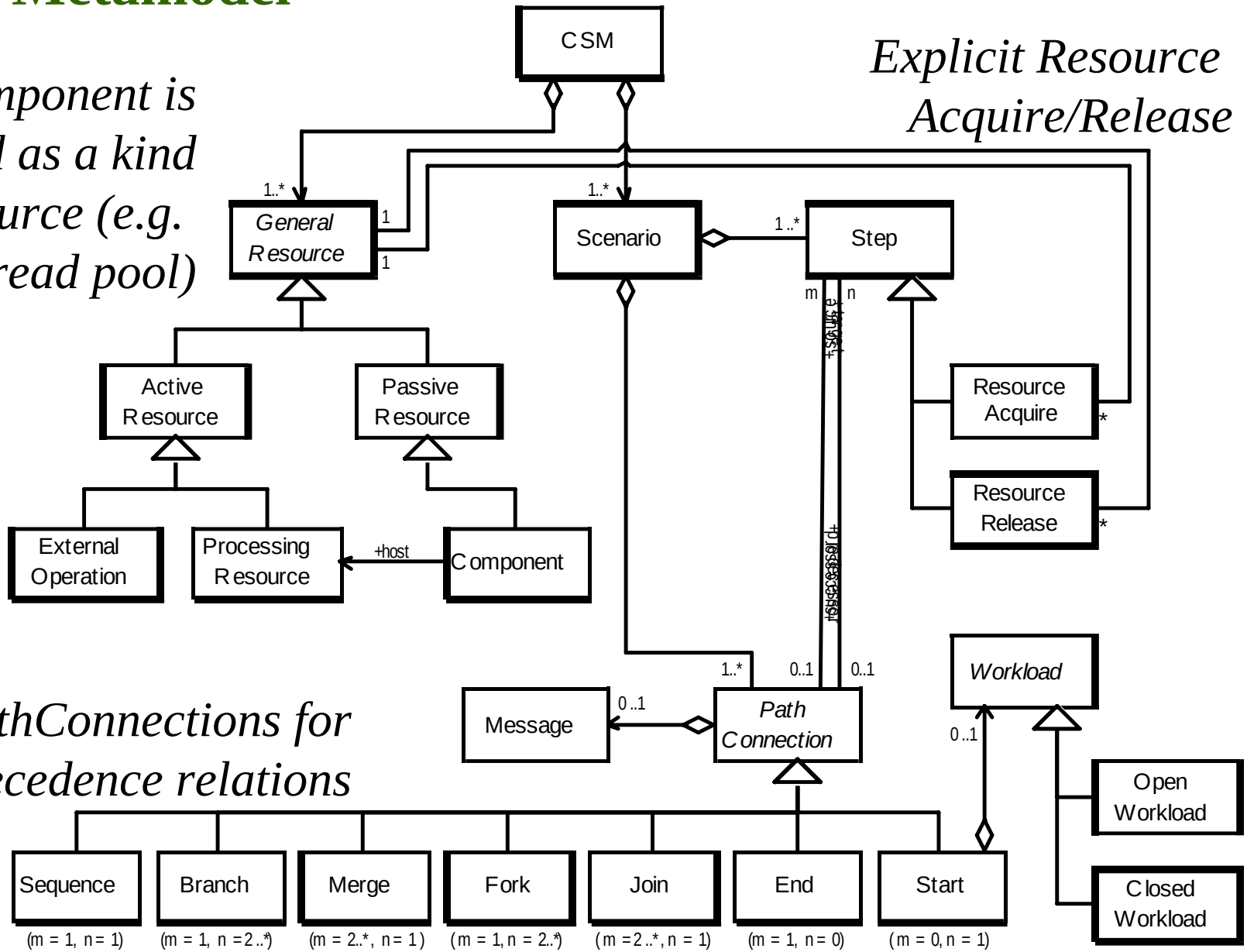
- CSM is a convenient common view of scenarios
  - UML sequence diagram
  - UML activity diagram
  - UML state machine
  - other scenario languages, e.g. Use Case Maps
  - designed for the PUMA project (Performance by Unified Model Analysis)
- based on the SPT metamodel
  - Steps, Components, Resources
  - nesting: scenario within a step
  - added: noSync: a parallel branch that does not join

Carleton UNIVERSITY

CSM Metamodel

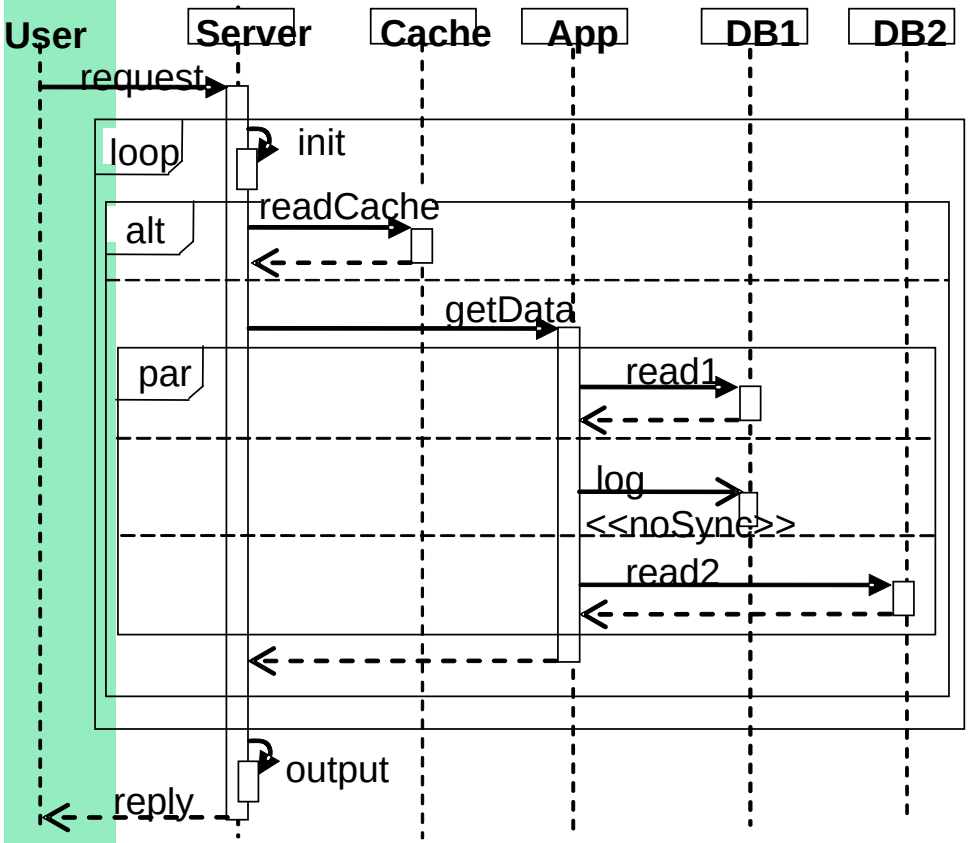
A Component is regarded as a kind of Resource (e.g. process thread pool)

Explicit Resource Acquire/Release

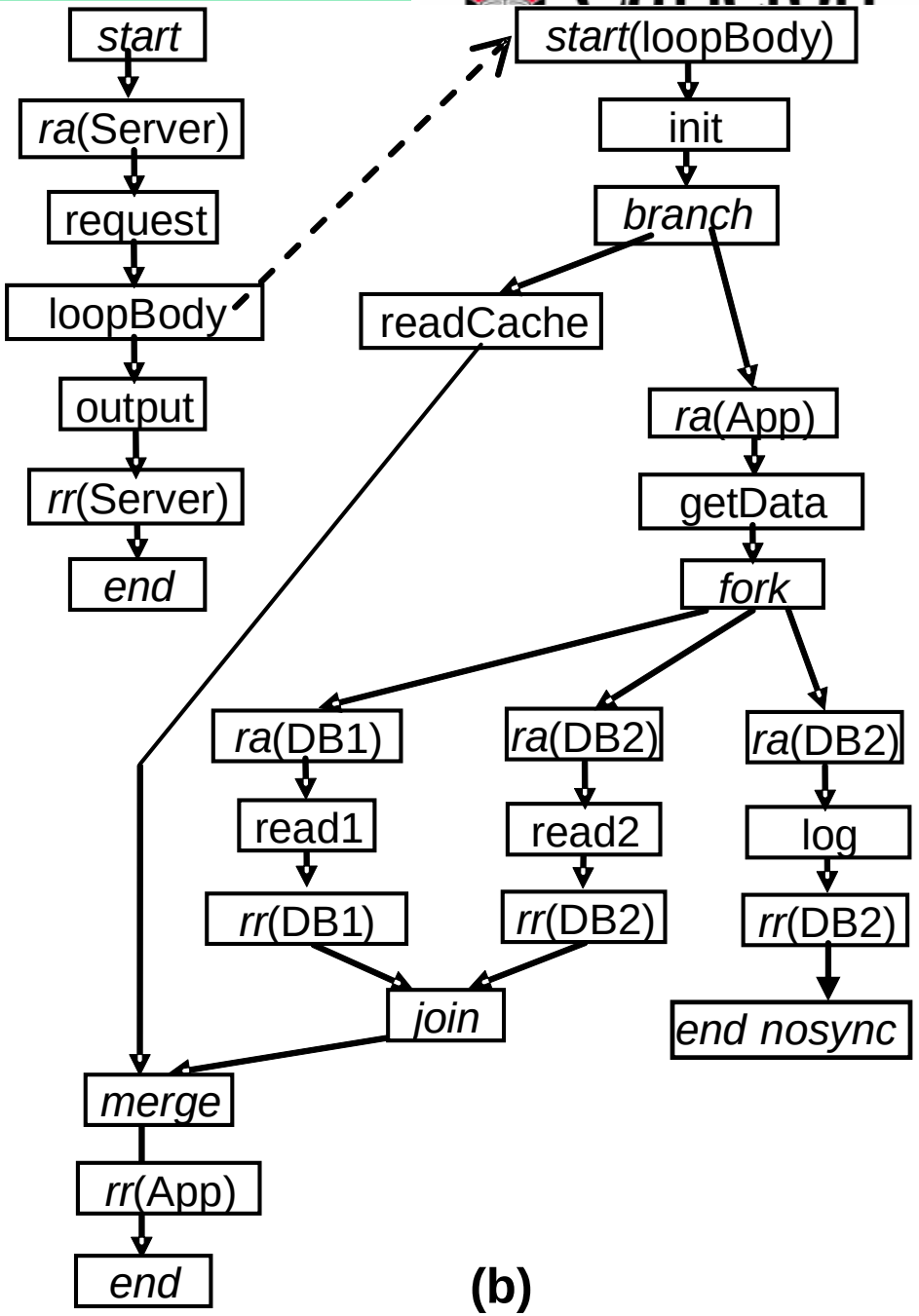


PathConnections for explicit precedence relations

SD to CSM


User
Server
Cache
App
DB1
DB2
request
loop
init
readCache
alt
getData
par
read1
log
<<noSync>>
read2
output
reply

(a)


start
ra(Server)
request
loopBody
output
rr(Server)
end
start(loopBody)
init
branch
readCache
ra(App)
getData
fork
ra(DB1)
read1
rr(DB1)
ra(DB2)
read2
rr(DB2)
ra(DB2)
log
rr(DB2)
join
end nosync
merge
rr(App)
end

(b)

Carleton UNIVERSITY

3. Relationship of SPT to Performance Models

- Deployment specifies the *physical environment* and the *components* (processes) that execute the behaviour
- Behaviour specifies the operations that take time and resources

- The path to obtaining the performance model depends on the model formalism...
  - Queueing network (QN) represents physical resources, with behaviour as a demand pattern
  - Extended QN (EQN) adds selected logical resources
  - Petri net (PN) represents actions and resources in a different way

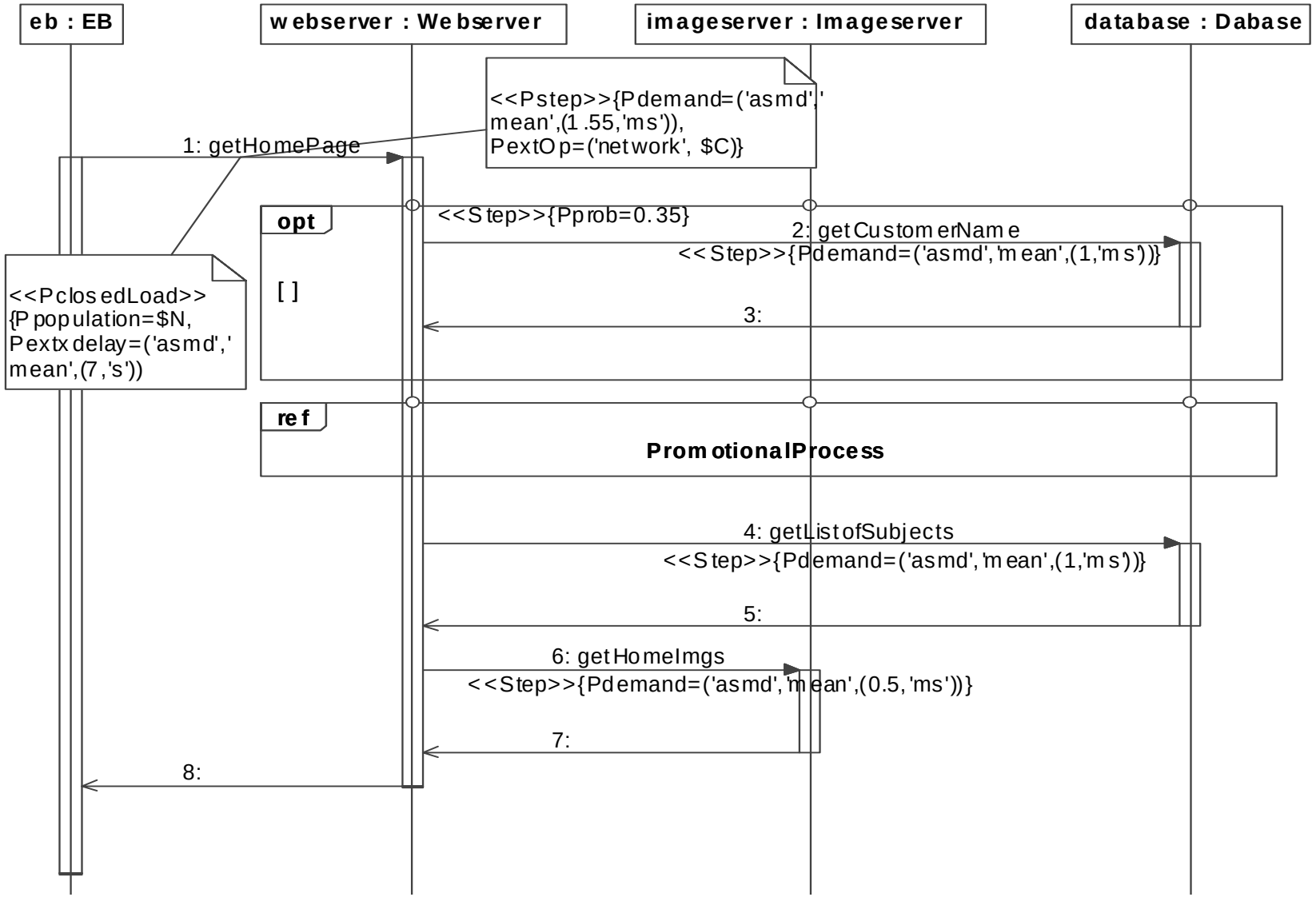
Carleton UNIVERSITY

Target models: plain Queueing Network (QN)

- server = host device
- service class = an operation by Steps on a Host
- service time = CPU demand for the operation
  - summed over all Steps on that host
  - Components are aggregated out
- routing: from Host to Host,from the sequence of Steps
- total demand for one server= sum in scenario, end to end
  - weighted by number of times the Step is traversed
- analytic queueing model: use the demands by host and class

Carleton UNIVERSITY

TPC-W (electronic bookstore-like web benchmark) GetHomePage scenario (one of 14 in the spec)

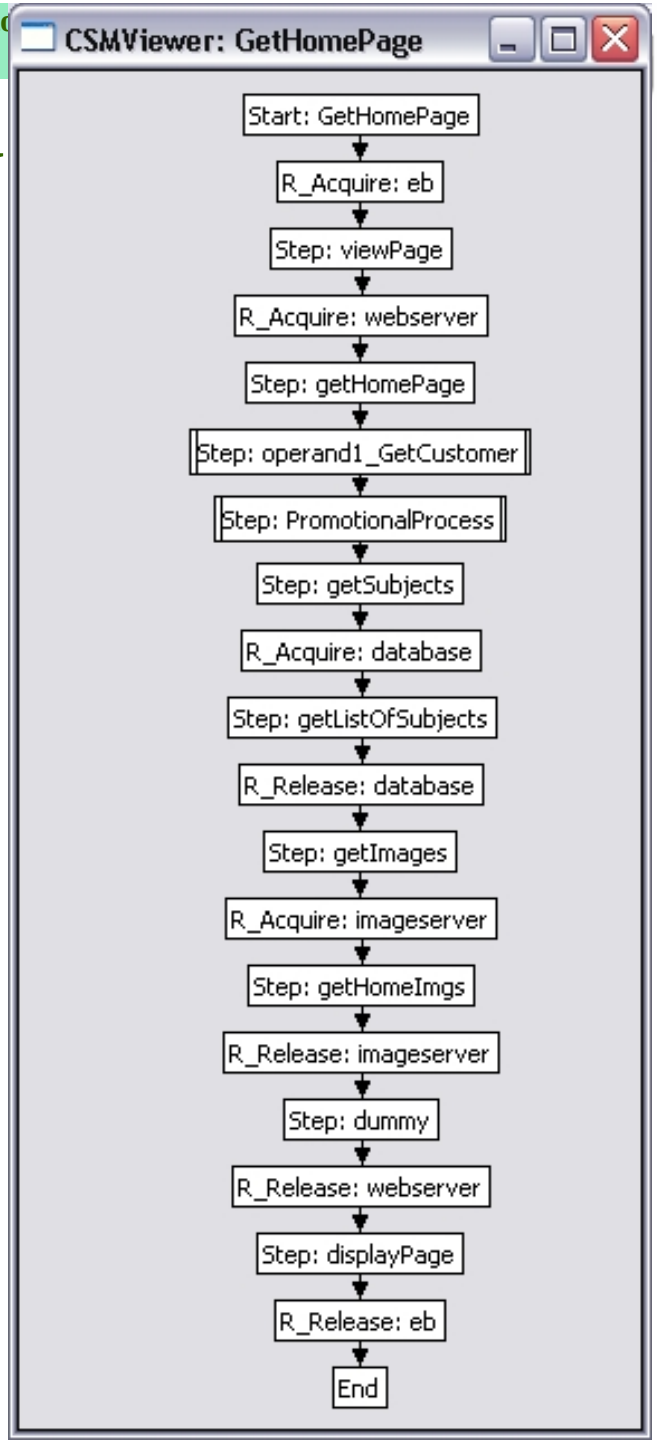


CSM for TPC-W GetHomePage

nested scenarios for

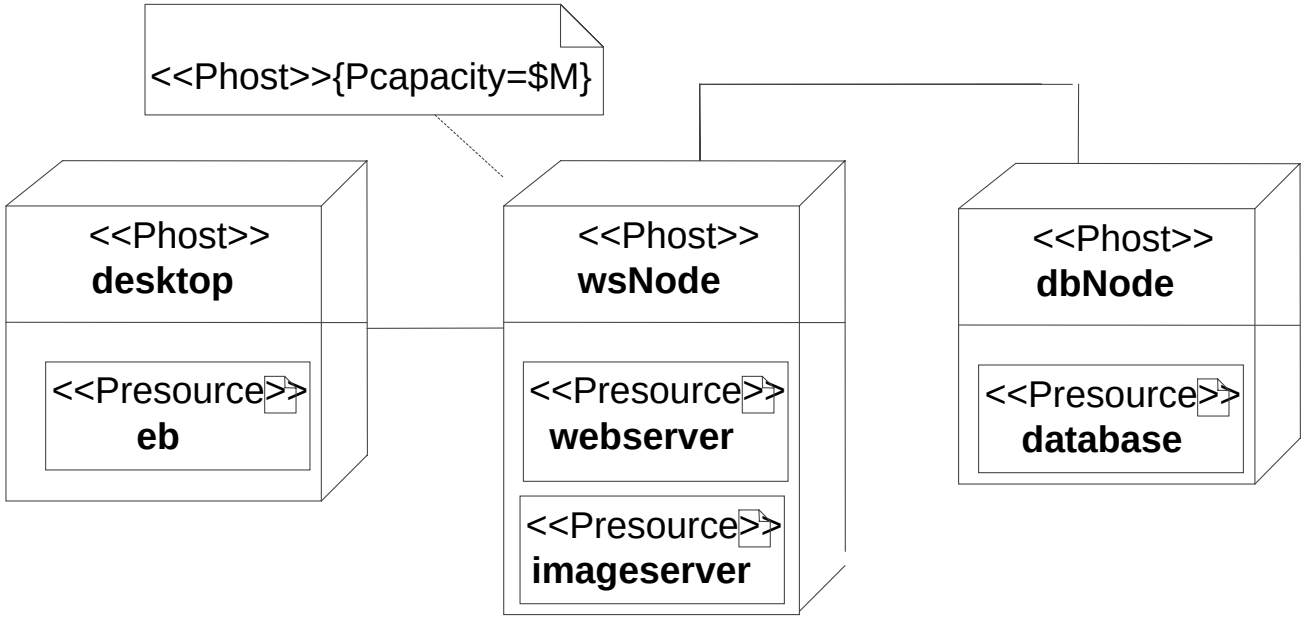
- Get_Customer
- PromotionalProcess

- processor resources are implicit in the components (process resources)
- steps have host demands
- resource holding is nested



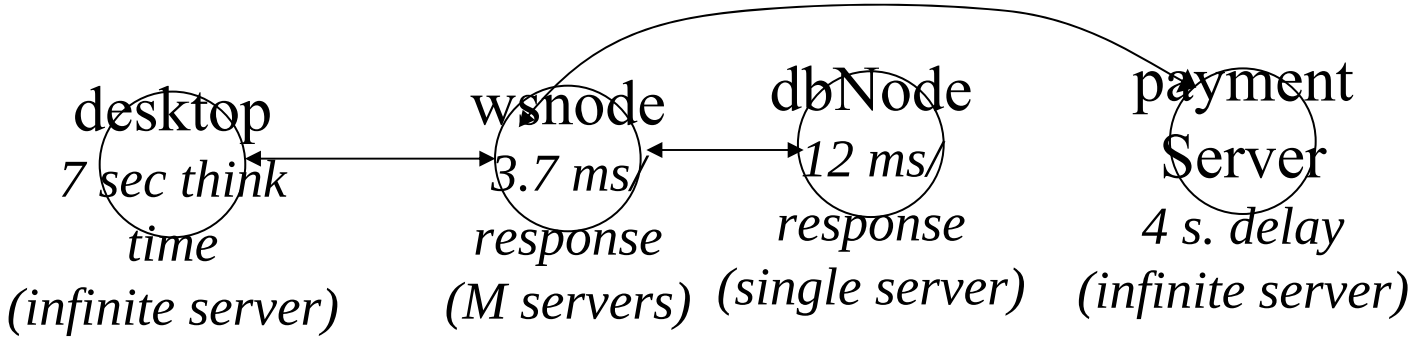
Carleton UNIVERSITY

TPC-W deployment and QN model



payment server is external to the UML spec

Queueing model has servers for the nodes



GetHomePage Service demands

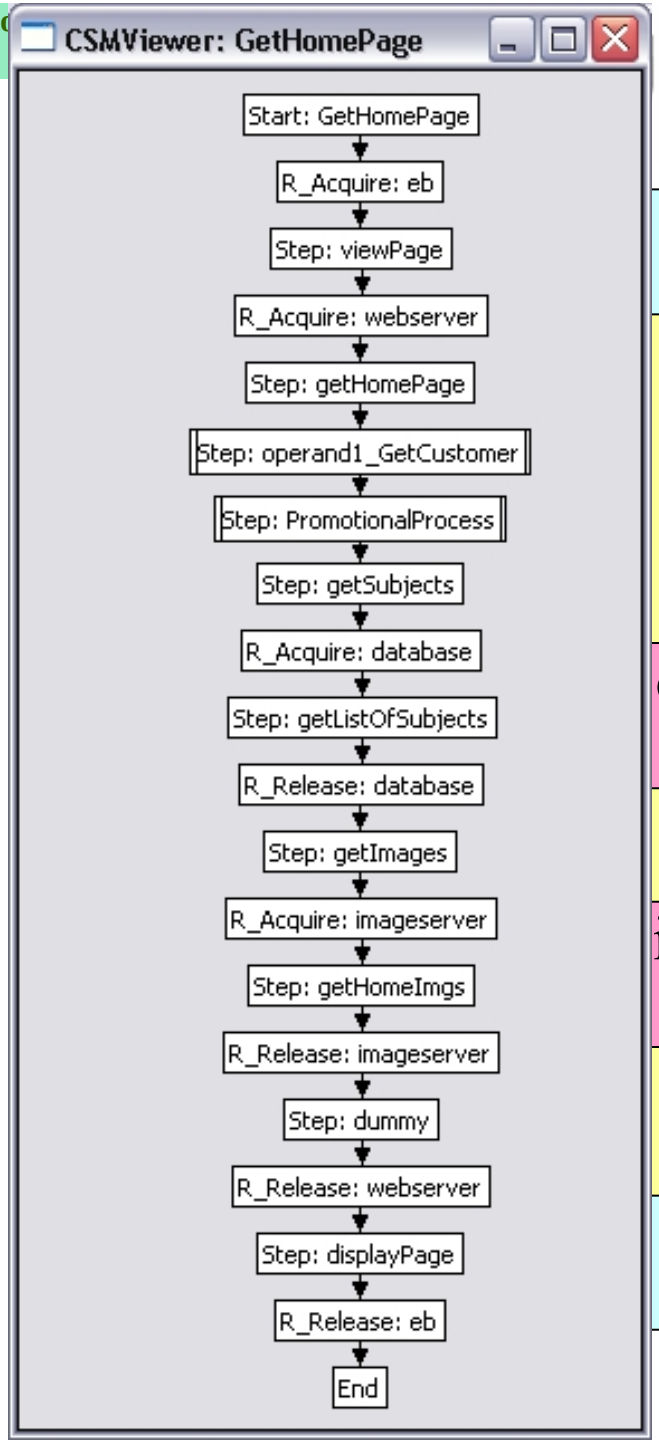
Carleton UNIVERSITY

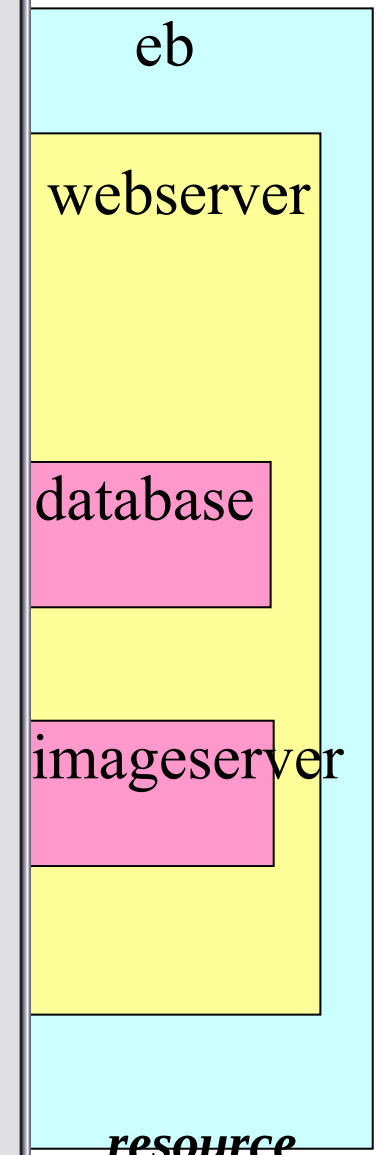
Additional (Logical) Resources

- What resources should be represented?
  - the "right" answer is to represent all those that may limit performance in the expected operating regime
  - but this usually can't be answered exactly
  - rule of thumb: resources with significant utilization
- Examples:
  - process thread pools, if limited
    - some processes are single threaded to provide mutex
    - actual thread pools often adapt in size to the demand, to save memory, but have a maximum
      - another performance question is, static or dynamic pool?
  - buffer pool size
  - critical sections, admission token pools
- SPT/MARTE supports abstract resource concept, explicit acquire/rel

Logical Resources in the CSM for TPC-W GetHomePage:

- logical resources for the processes
- eb, webserver, database, imageserver
  - holding is nested
- processor resources are implicit in the components (process resources)
- steps have host demands

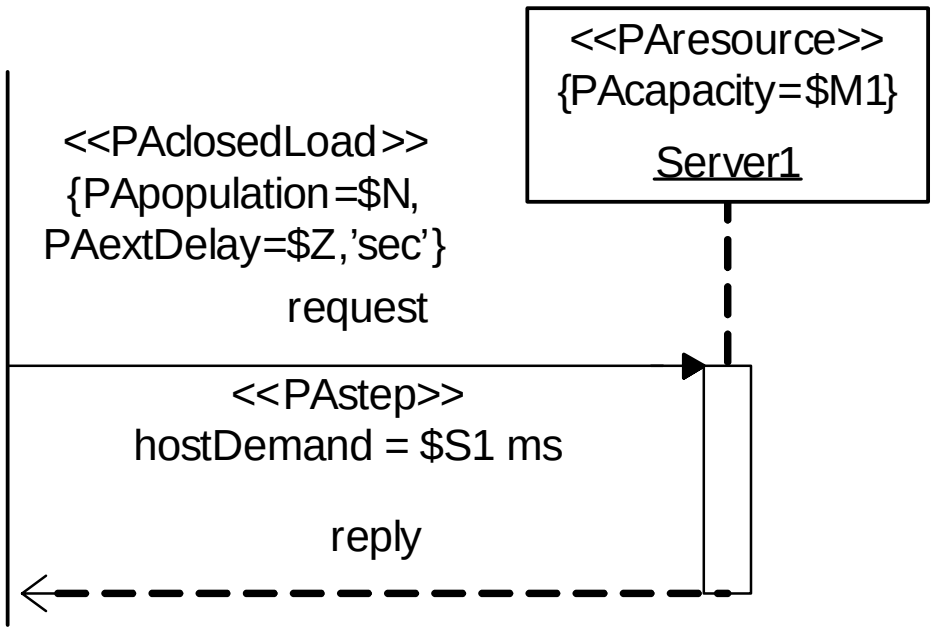


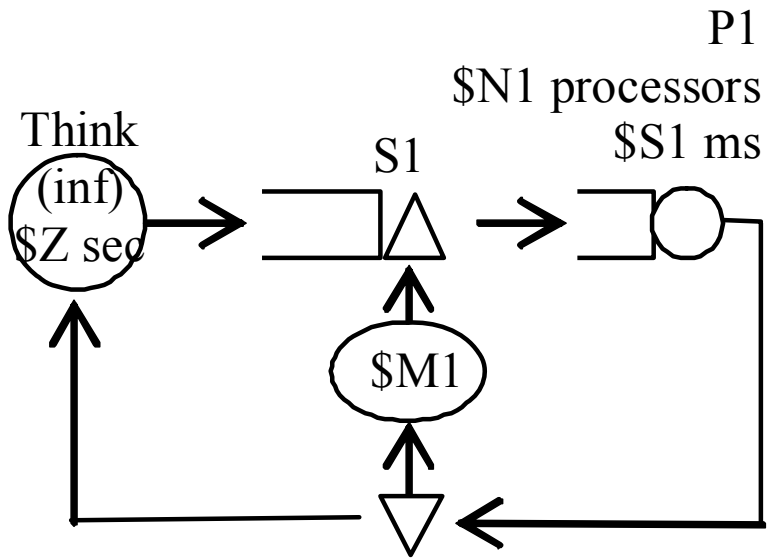


Carleton UNIVERSITY

Extended QN (EQN)

- logical resource is a token pool with a queue and discipline
  - customer must "hold" two resources at once (not classical QN)
- requests may have to wait
  - "service time" of a token, also called the "holding time" of the logical resource, is found by solving the rest of the model
- solution requires iteration between two models that DO NOT have simultaneous resources

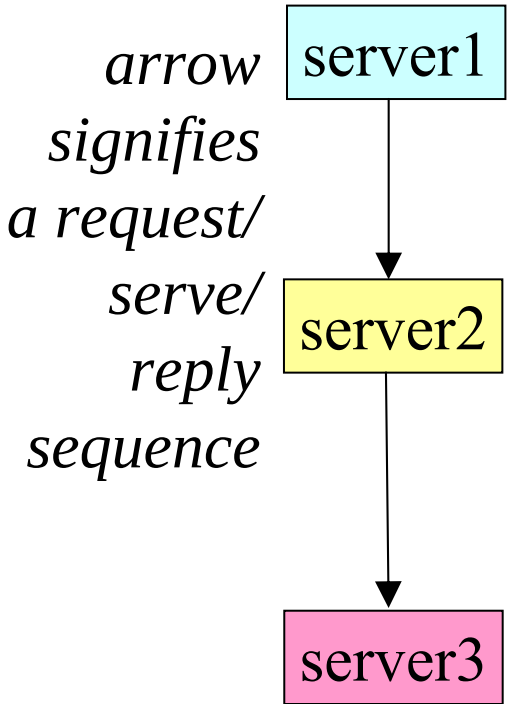


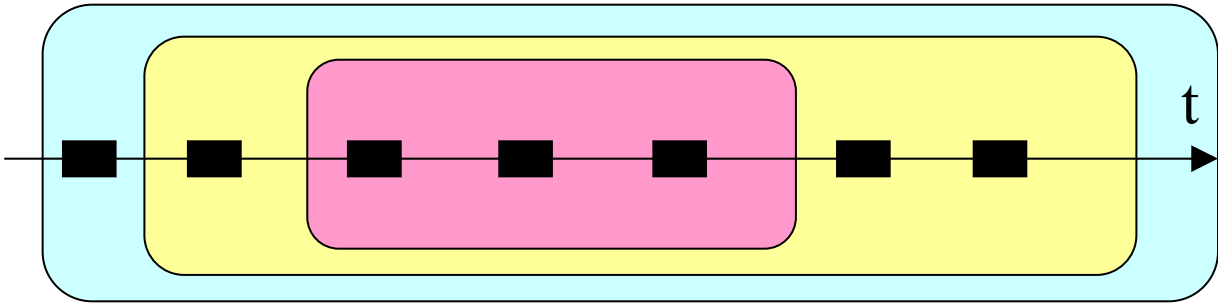


Carleton UNIVERSITY

Layered QN (LQN)

- a kind of EQN in which resource holding is nested
  - "nested resource contexts"
- nested resource-holding gives layered queueing
- the layers organize the solution process (LQNS, etc.)



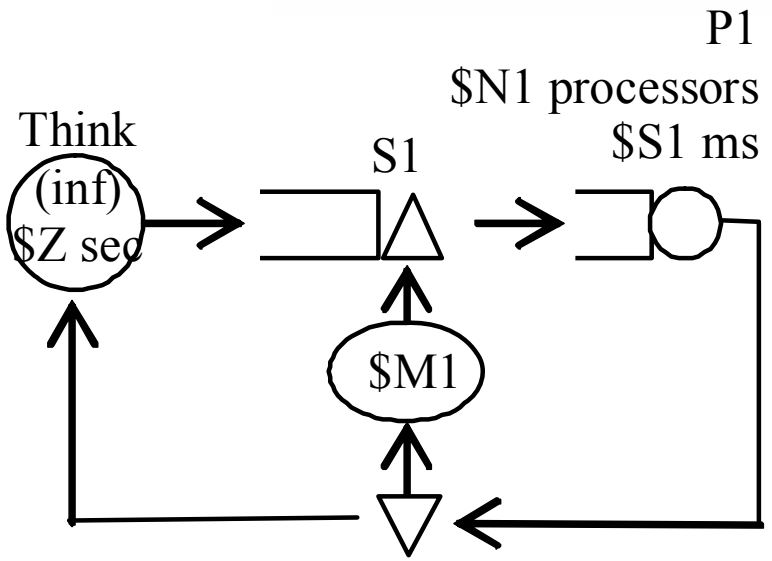


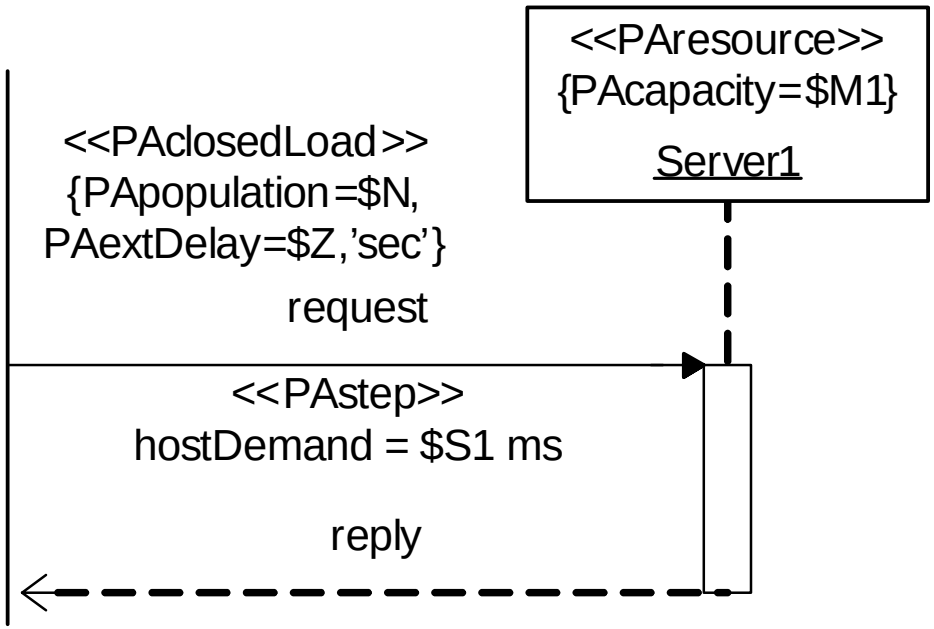
steps executed by different servers, in different nested resource contexts

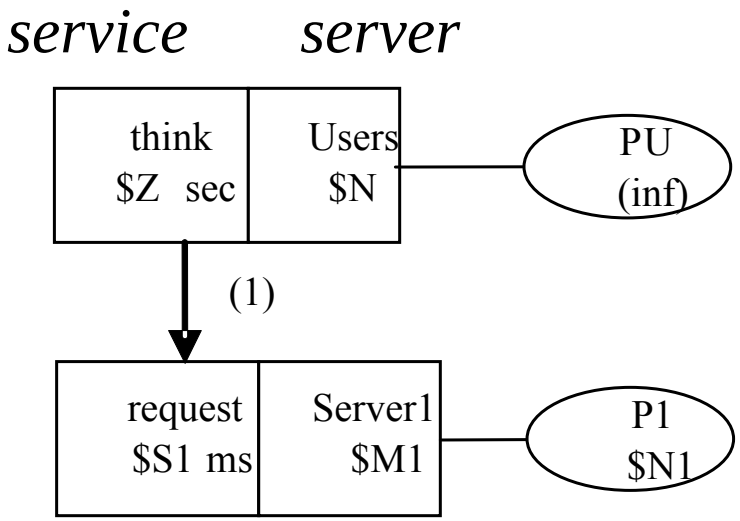
Carleton UNIVERSITY

LQN for one layer

- UML SD
- EQN with extra resource queue
- equivalent LQN



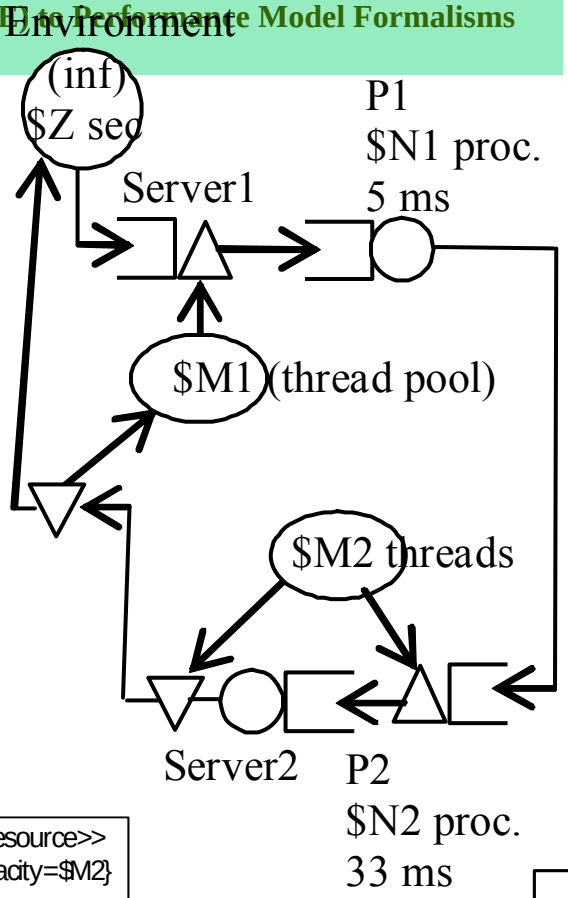


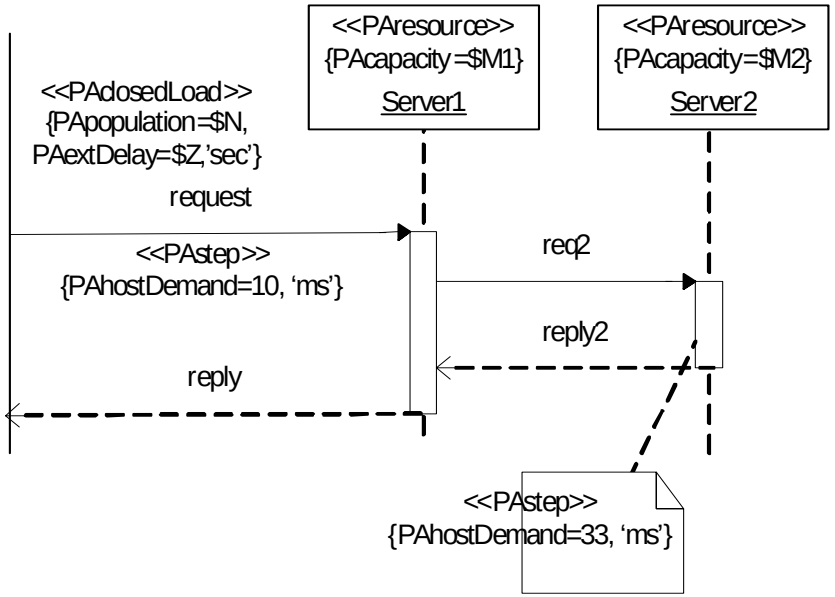


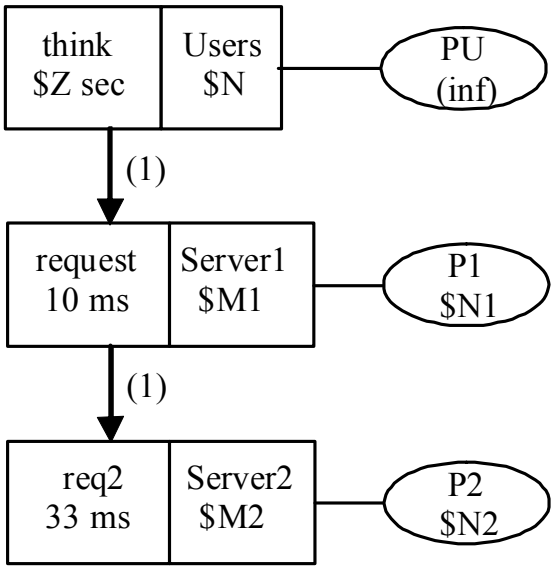
Carleton UNIVERSITY

LQN: two layers

- UML SD
- explicit EQN
- LQN





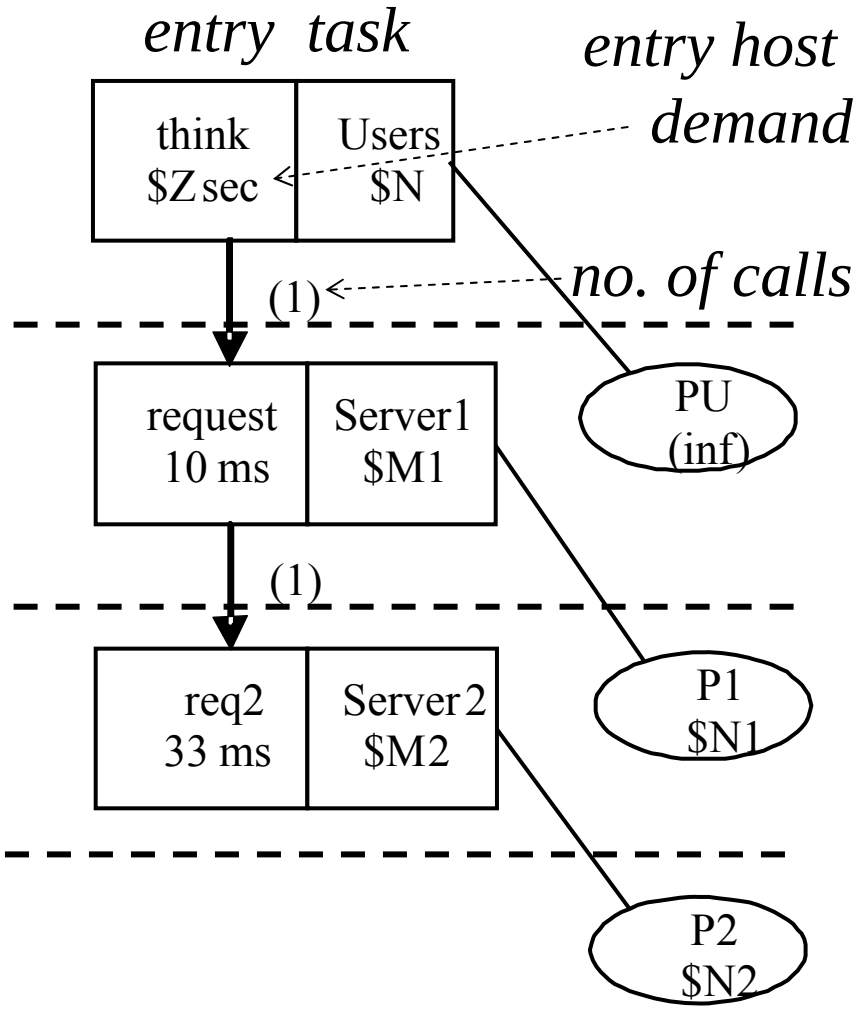


Carleton UNIVERSITY

LQN elements

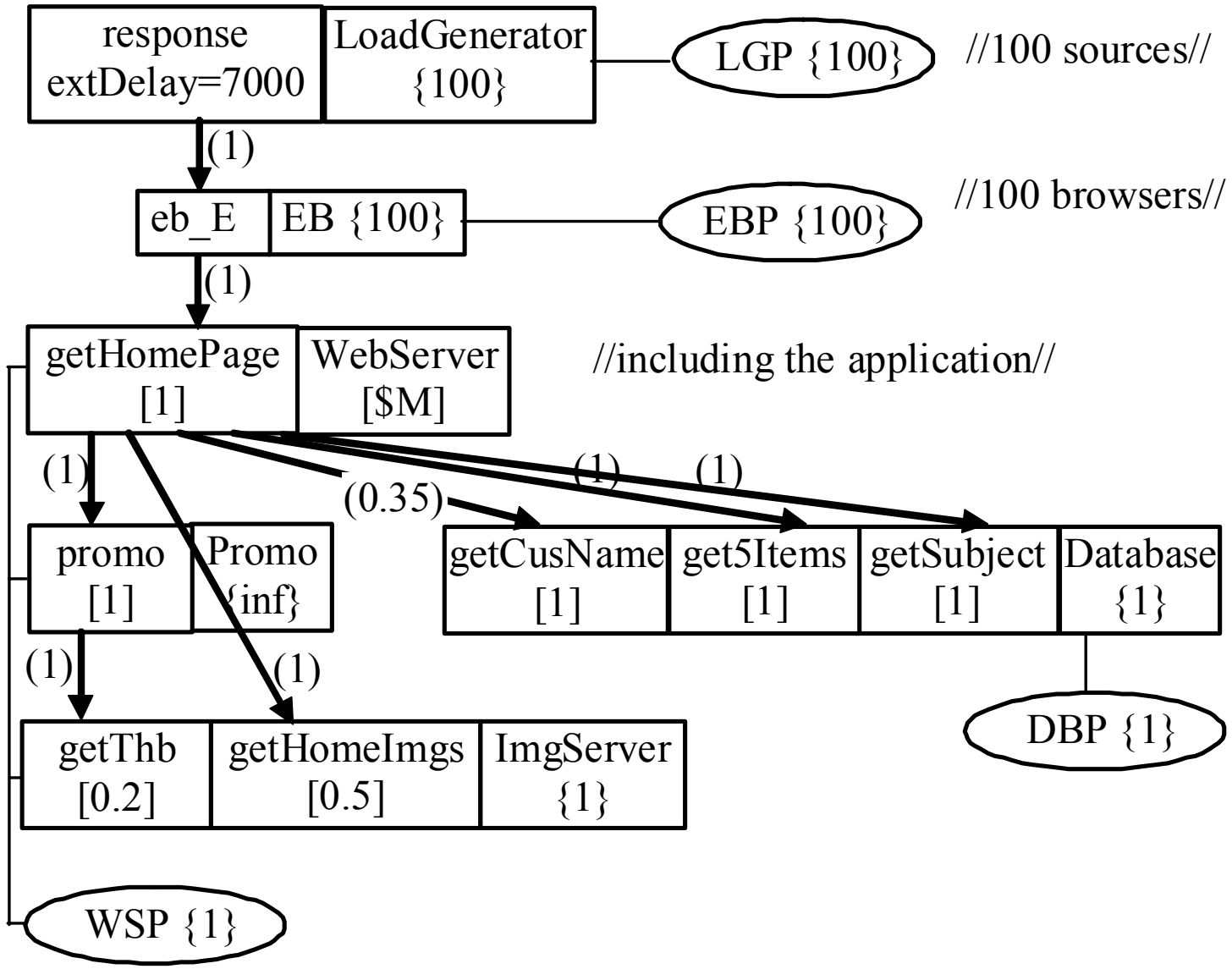
- servers called "tasks"
  - server has a queue
  - may be a multiserver
- offers services called "entries"
  - different entries offer different classes of service
- entries actually run on a lower level of server, the processor
  - task becomes a customer
- a task may also become a customer for a lower layer task
  - the service time of the entry includes delay for queueing for the processor
  - and for the lower server
- blocking: request-reply interactions

Four layers



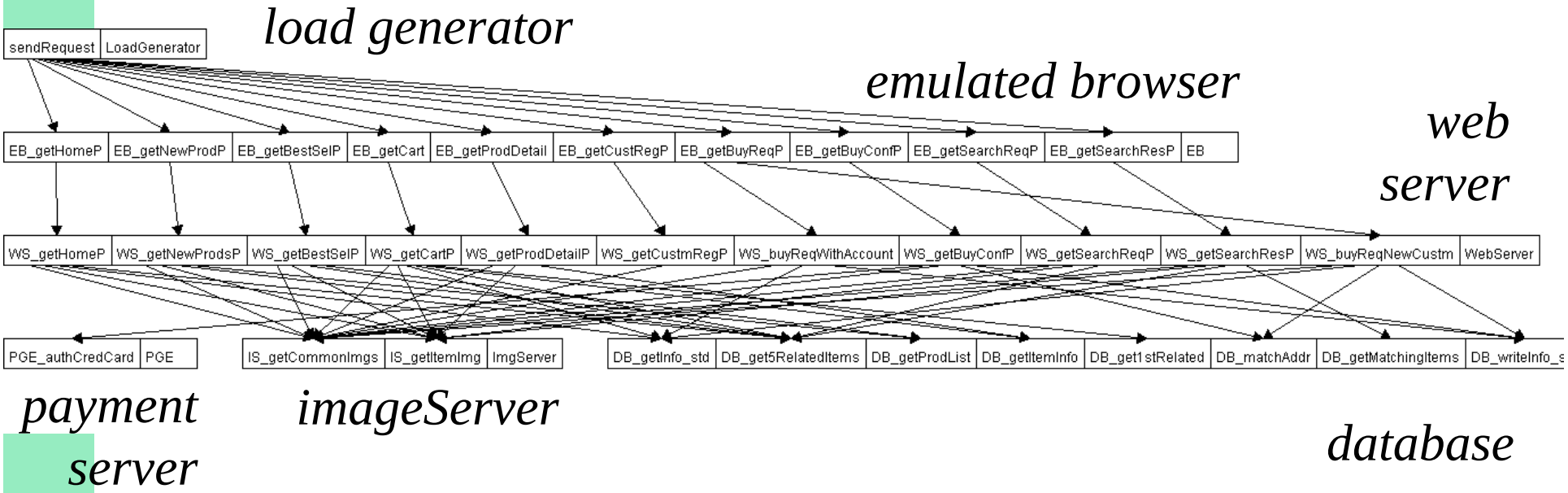
Carleton UNIVERSITY

LQN for the GetHomePage deployment and scenario



Carleton UNIVERSITY

LQN for 10 scenarios

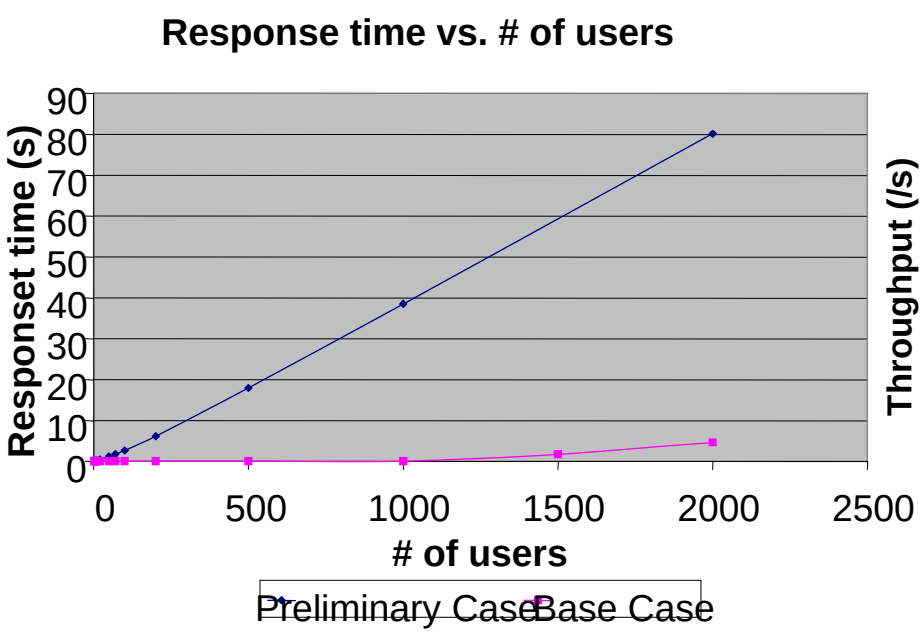


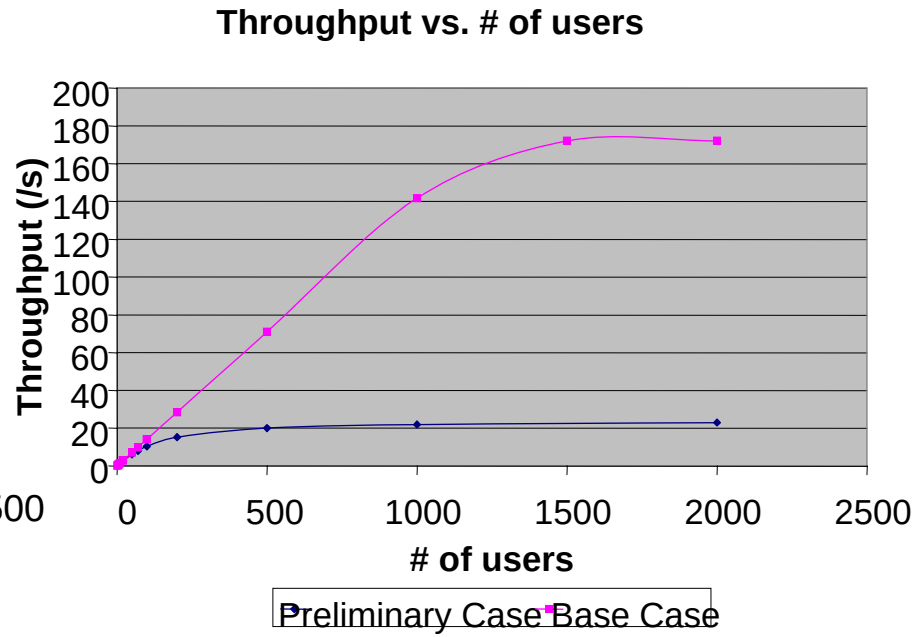
- automatically generated by PUMA tools by Dorin Petriu

Carleton UNIVERSITY

LQN solution results...

- the base case corresponds to the system just described...
  - capacity about 30 users, bottleneck at single-threaded web server
  - low processor utilizations





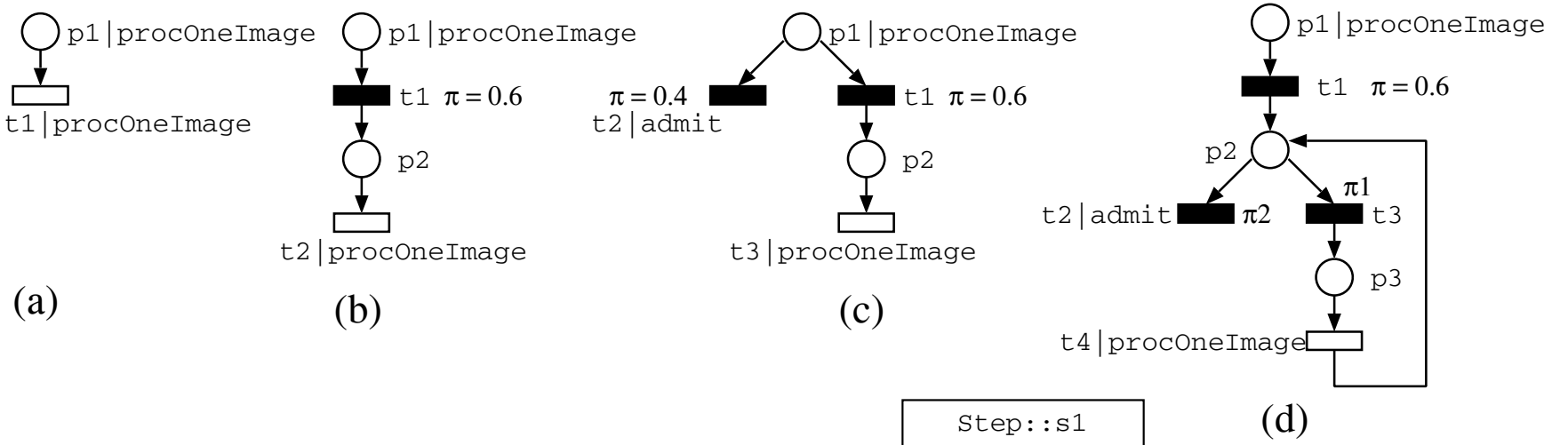
Carleton UNIVERSITY

Petri Net Performance Model

- Tokens move through a series of places by firing transitions
  - timed transitions represent processing delays
  - resources can be represented by tokens

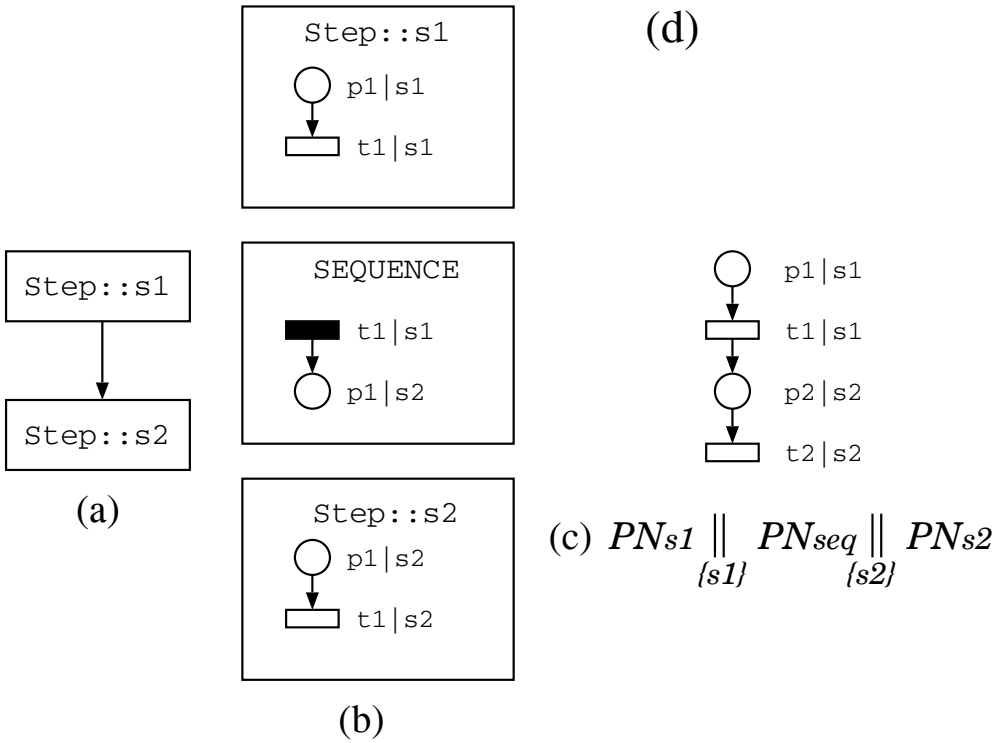
- Jose Merseguer made a tool to translate CSM, using labeled Generalized Stochastic Petri Nets
  - translate elements of CSM directly into corresponding PN subnet elements, using standard PN patterns
  - using labels to glue the successive subnets together

Carleton UNIVERSITY



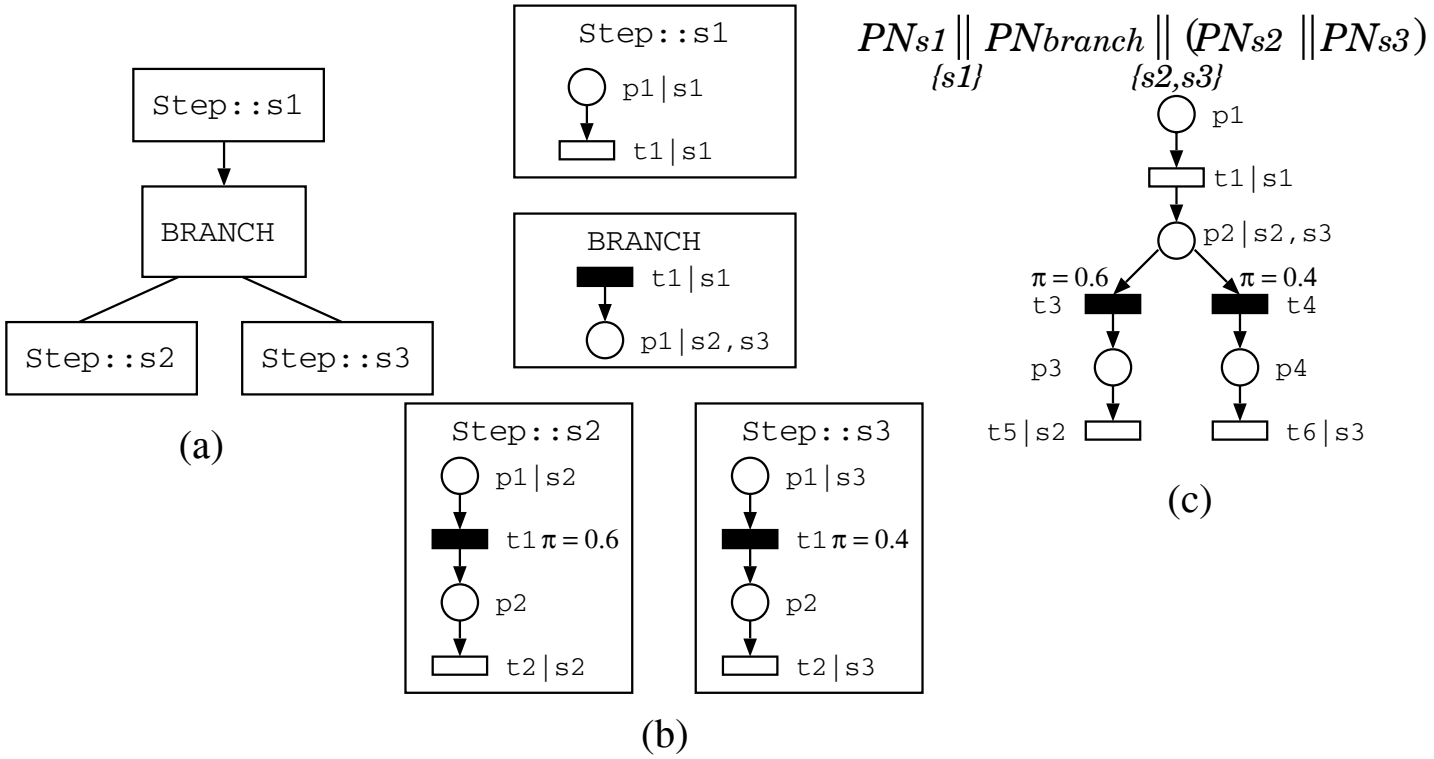
PN Patterns for a Step

...using Labelled GSPNs



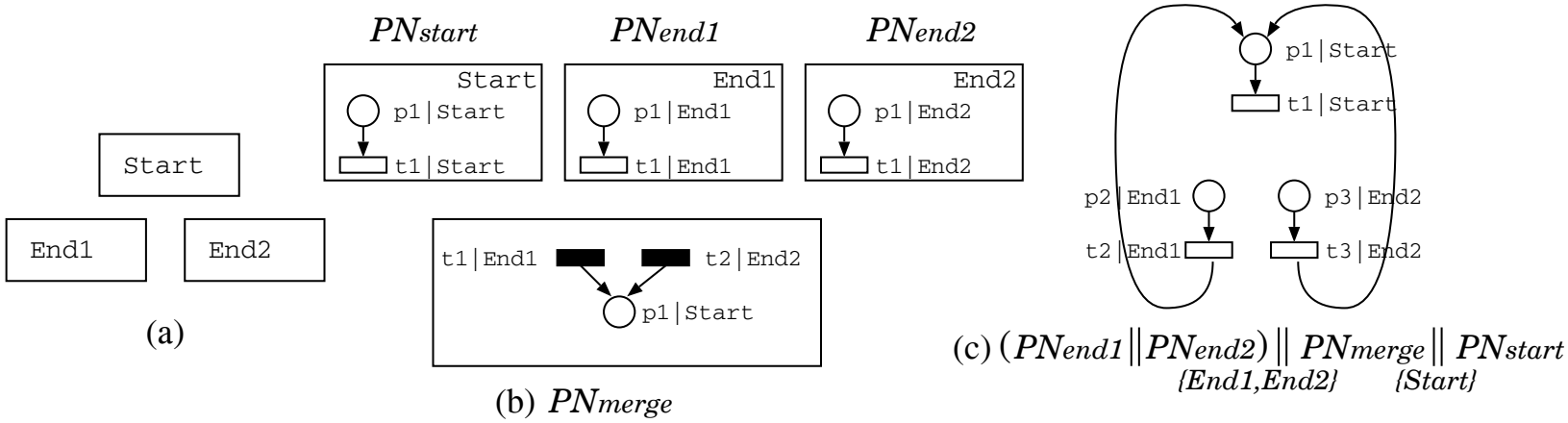
Carleton UNIVERSITY

PN Patterns for a Branch



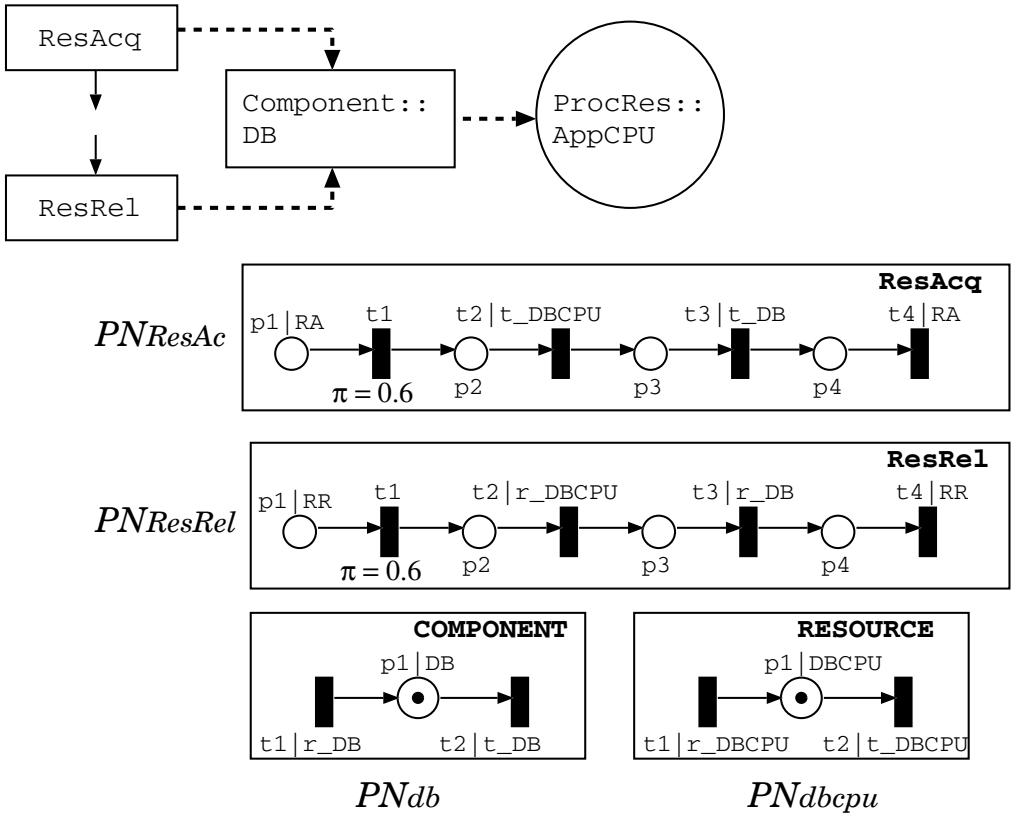
Carleton UNIVERSITY

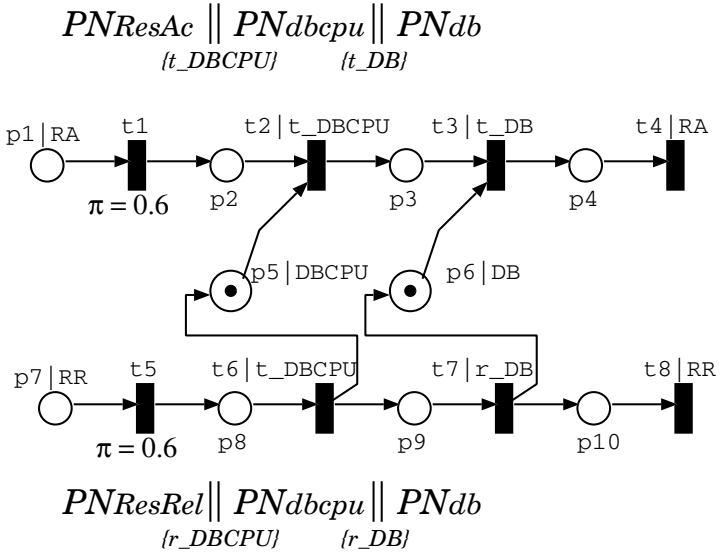
PN Patterns for Start and End



Carleton UNIVERSITY

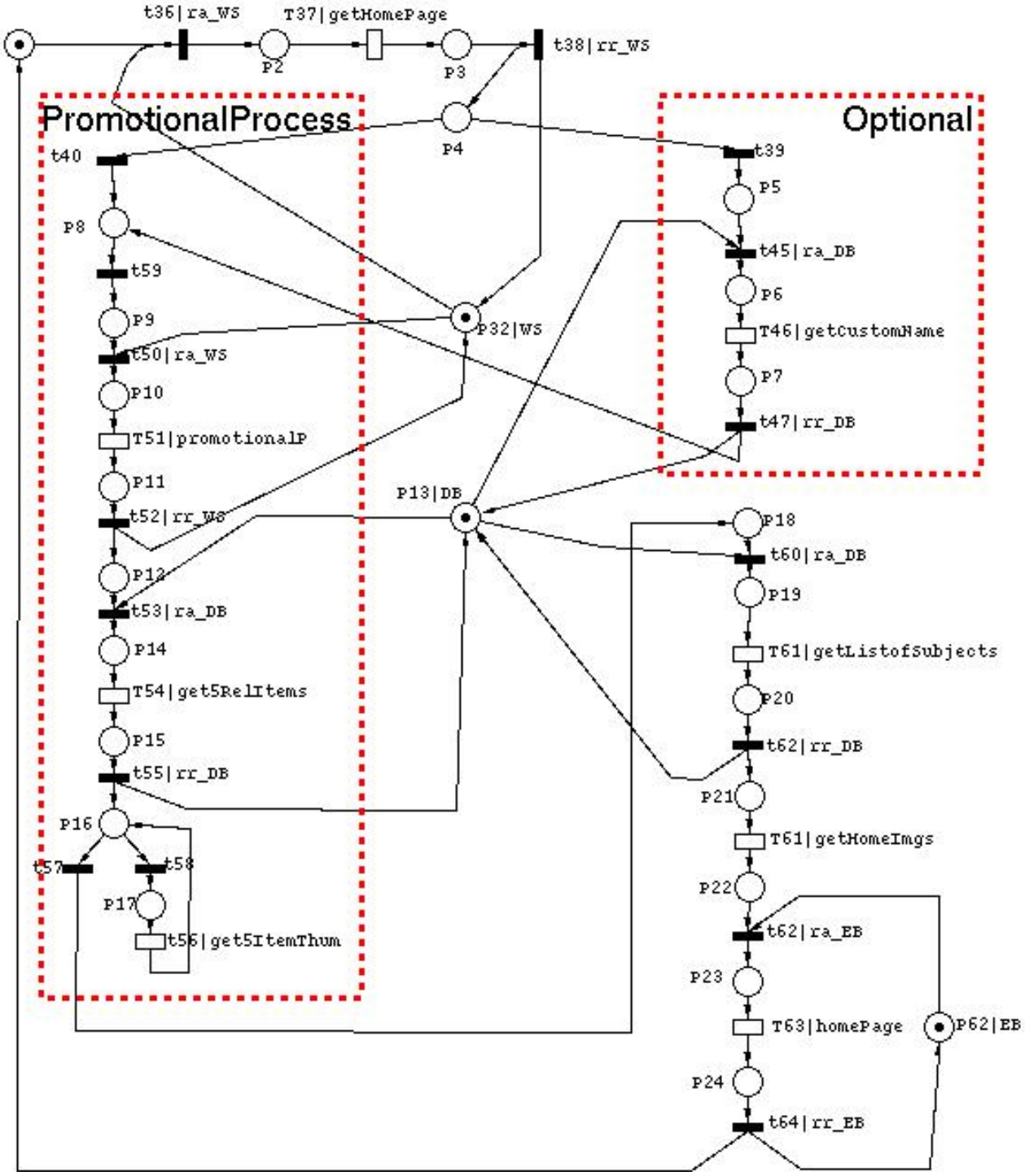
PN Patterns for a Logical Resource





Overall Petri Net Model of GetHomePage (by Jose Merseguer)

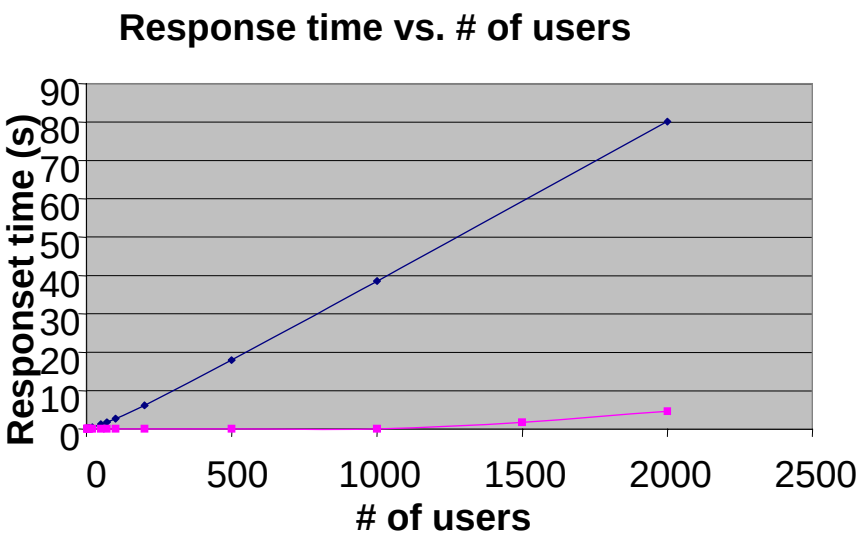
- reasonable complexity
- good traceability to UML naming
- state space a problem only with many customers

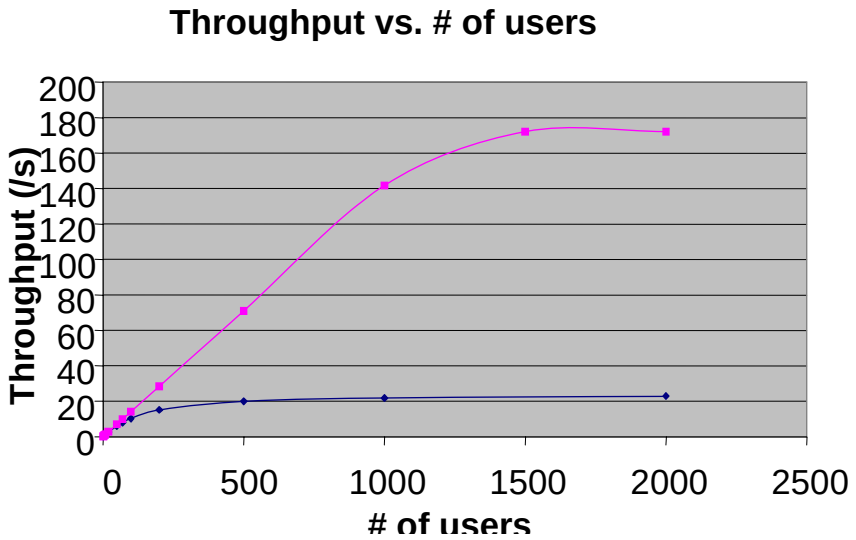


Carleton UNIVERSITY

Performance Improvements: Resources

- Identify performance limitations in the solution
- Identify causes by examining the solution in detail
- e.g., bottleneck at webserver:
  - increase thread pool size to 10
  - also increase webserver processors to 2
  - increase database to 2 replicas
- achieves the improved performance:

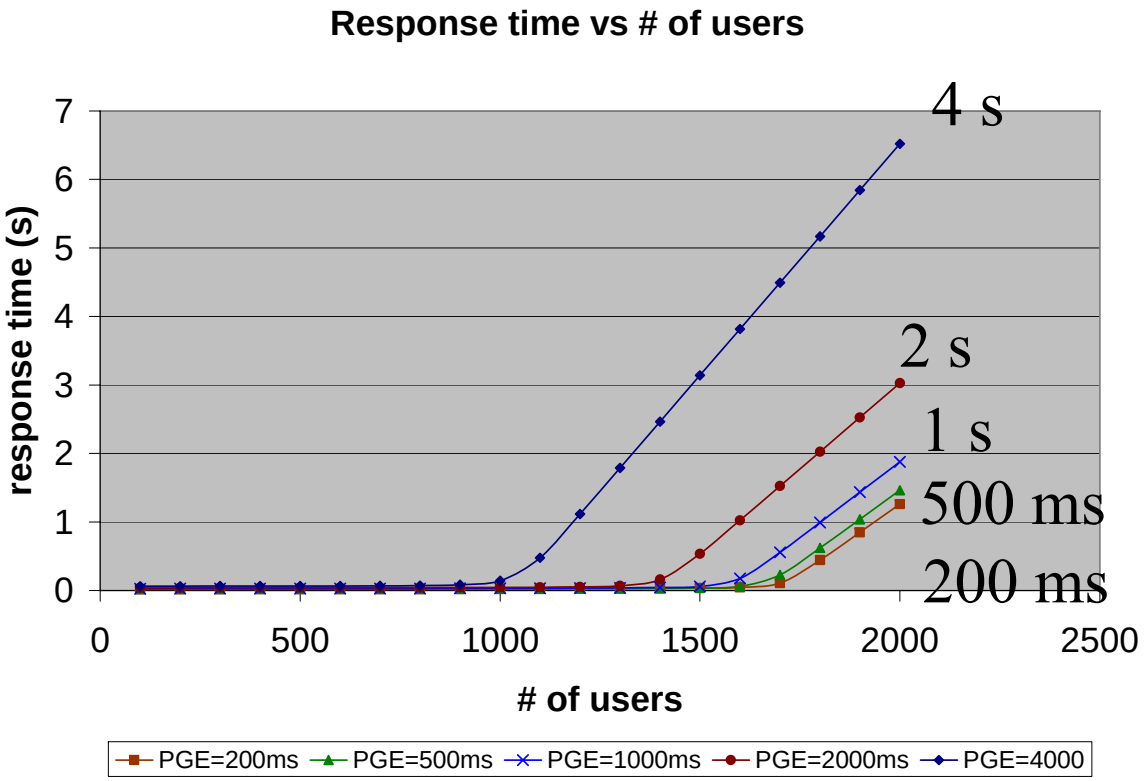




Carleton UNIVERSITY

Performance Improvements: Operations

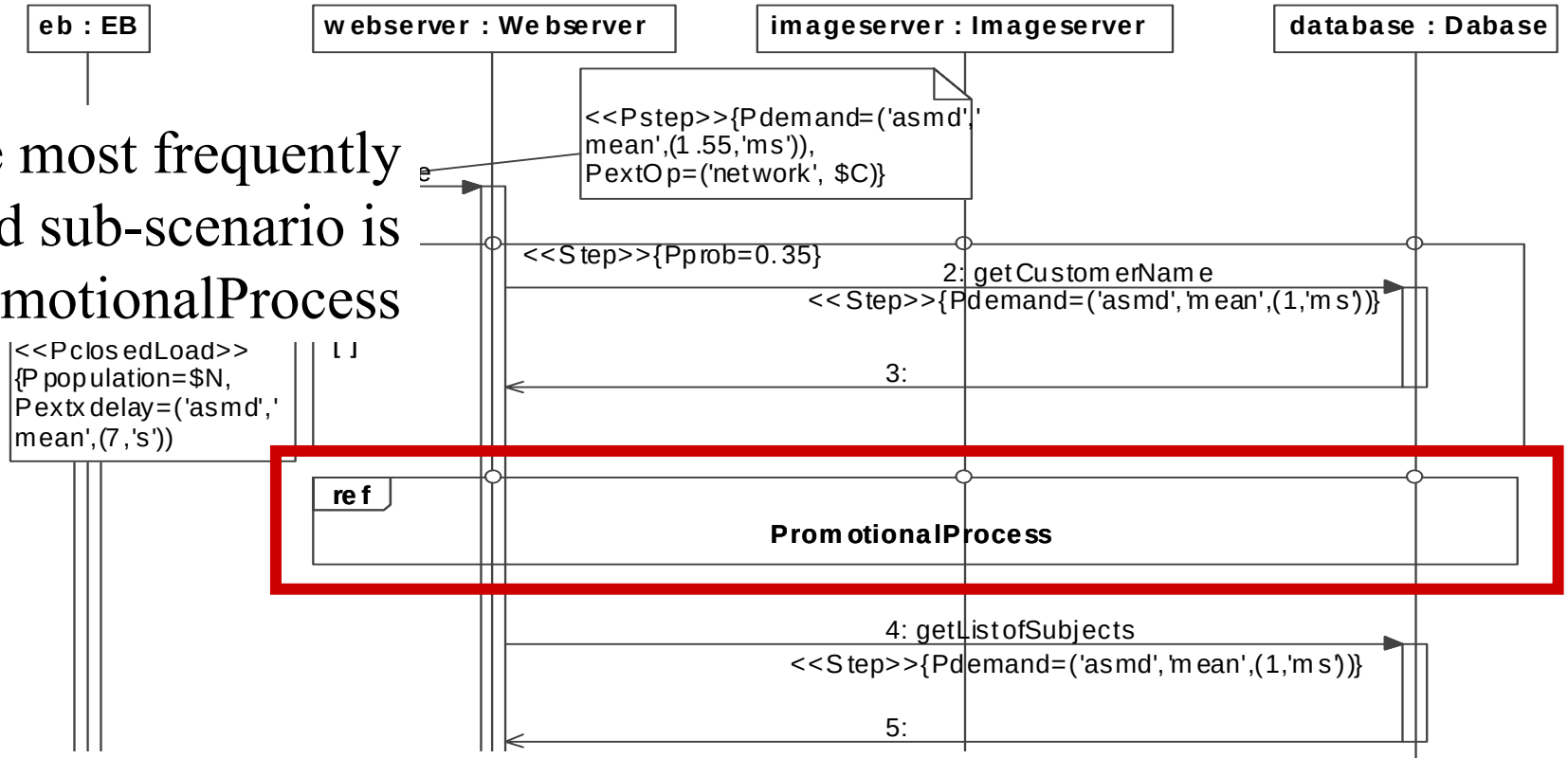
- The subscenario with the longest delay is the one that queries the payments server (with a 4 sec delay)
  - suppose this could be reduced by using high-speed networks
  - capacity increased



Carleton UNIVERSITY

Performance Improvements: Redesign

The most frequently accessed sub-scenario is PromotionalProcess

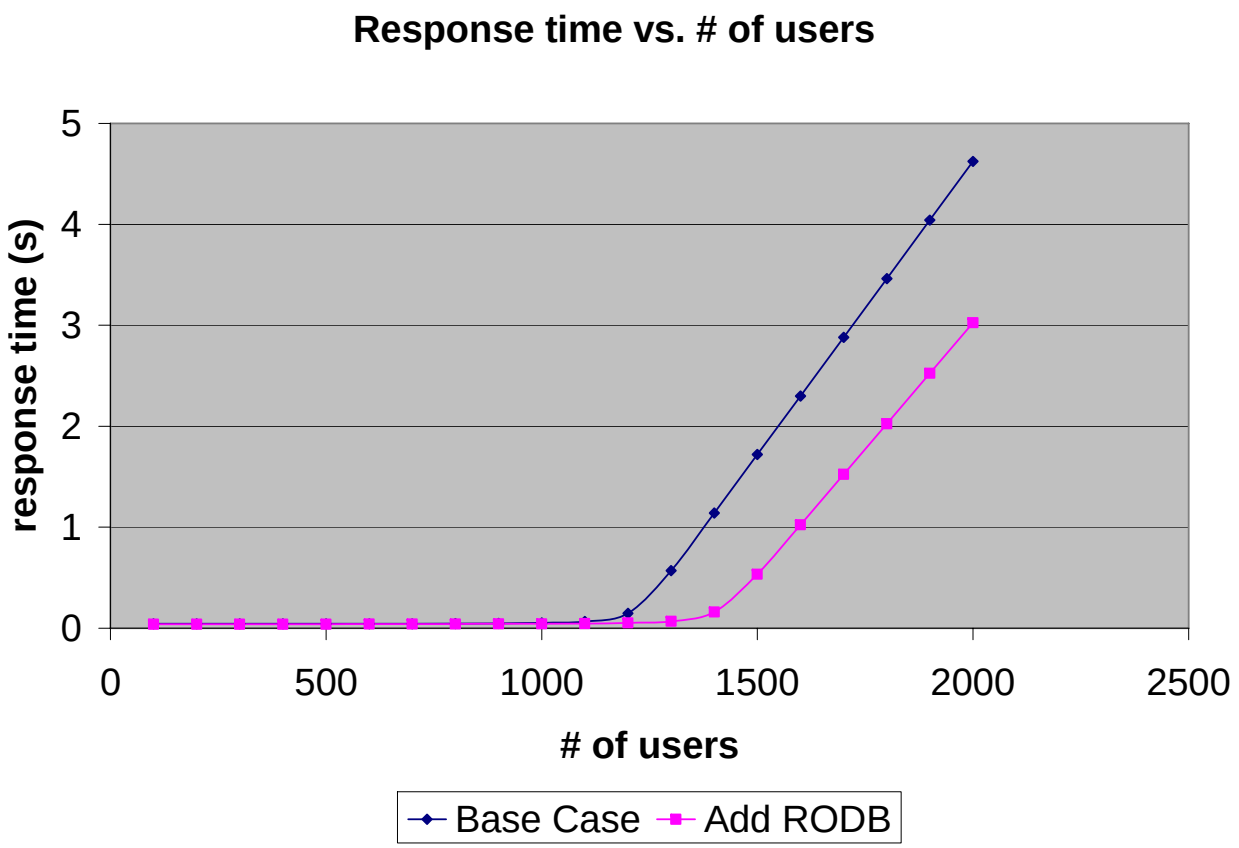


It makes a database demand... so

- suppose a special read-only promotions database with heavy caching,
- with a reduced demand for these accesses (5 ms)

Carleton UNIVERSITY

Read-only database for Promotions...



- capacity for 1 sec mean response time increased by about 200

Carleton UNIVERSITY

What are we missing?

what about

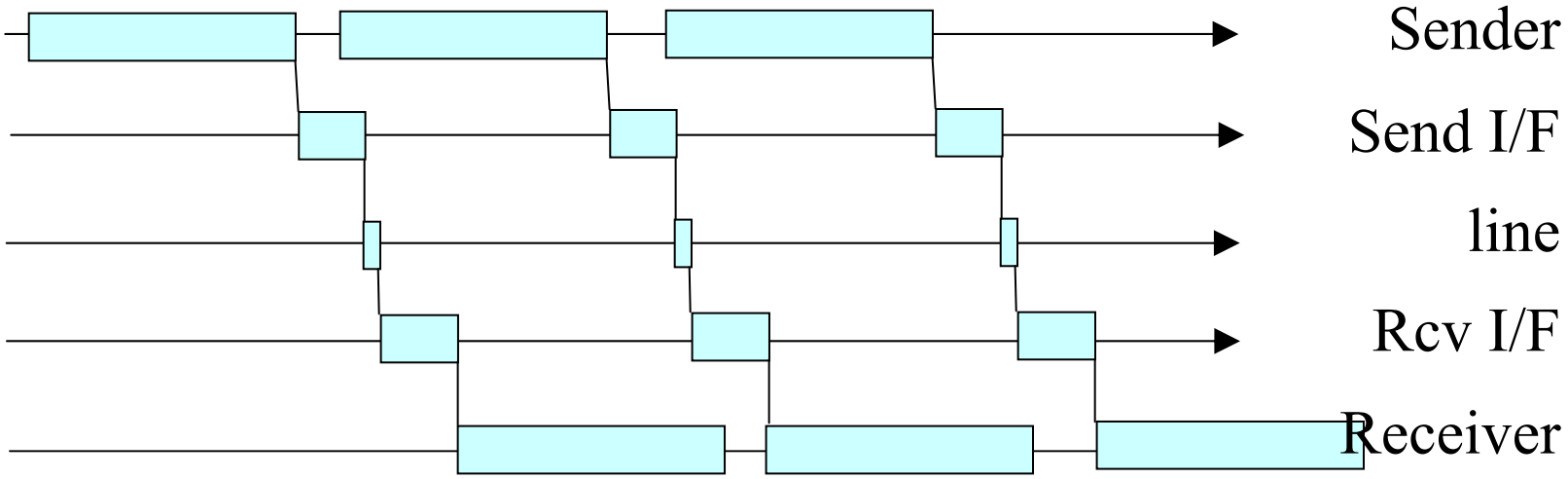
- Operating system overhead?
  - Idea #1: it's included in the Step demands. BUT how to account for a different OS or version
  - Idea #2: OS overhead could be modeled as a node property,
    - roughly, as a % of operation time
    - in detail for each OS call, with Step demands for calls: too detailed for manual modeling
- Middleware costs
  - so–called "web application servers"... most of the execution cost of an operation is shown to be the middleware
  - similar ideas, but now Idea #2 looks better, since middleware operations are more coarse grained
- some extensions in MARTE to describe these.

Carleton UNIVERSITY

Missing (2)

what about

- communications effects?
  - node processing costs, interface processor effect, latency on the channel
  - this is more complicated than it sounds: consider sending a sequence of packets. They are overlapped:



  - we do not specify this kind of complex detail in UML!

Carleton UNIVERSITY

Submodels in the performance modeling tool/domain

It makes sense to use submodels for system components or behaviours if they are available

- communications models
  - for instance for protocols like TCP
- environment components... you might have a library of submodels
  - web server
  - file server
  - file system
  - database
- these are identified as “external operations” in SPT and MARTE

Carleton UNIVERSITY

4. The New Profile: MARTE (being finalized now)

(Modeling and Analsis of Real time and Embedded Systems)

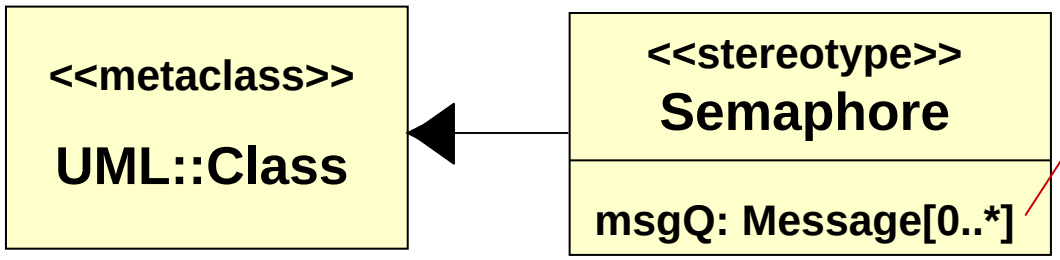
Why?

- upgrade to UML 2.X
- enrich with new Real Time and Embedded Modeling (RTEM) part
  - software and hardware aspects, richer resource types, composite structures
  - concurrency models, real-time mechanisms (e.g. mutex)
  - richer models of time (causal, synchronous, physical)
- overcome some limitations
  - e.g. a point-to-point delay can't be specified in SPT
- Analysis part: with *consistent* analysis domains
  - schedulability and performance analysis as in SPT
  - meant to be a platform for more kinds (power, reliability)
- core Time and Concurrent Resources (TCR) part
  - shared between modeling and analysis
  - plus Non-Functional Properties (NFPs): stronger data definitions

Carleton UNIVERSITY

New Meta-Associations in UML 2.X

- This was not possible in UML 1.x profiles
  - meta-associations represent semantic relationships between modeling concepts
  - new meta-associations create new semantic relationships
  - possibility that some tools will not be able to handle such additions
- UML 2.X capability added via stereotype attribute type
  - type of attribute is another metaclass
  - to be used with care!



creates an association between Class and Message that does not exist in the UML metamodel

Carleton UNIVERSITY

Core Part (TCR): Intent

- support specification of RT systems
- time models:
  - Asynchronous/Causal logical time
  - Synchronous/Clocked models (not in SPT)
  - Real/Continuous time models
- resources (General Resource model)
- a rich set of measures:
  - statistical measures
  - a delay measure shall be applicable to an interval bounded by a defined start event and a defined end event.
  - extensible to user-defined distributions, etc.
  - any measure may have multiple versions (e.g., a required value, an offered value, an estimated value, a test result value, etc.)

Carleton UNIVERSITY

Modeling part (RTEM): Intent

- Provide support to model embedded systems
  - reactive, control/command, multimedia, telecom
- common language for:
  - systems engineers - overall architecture; hardware/software tradeoffs;
  - hardware engineers –high-level description at block level
  - software engineers - detailed software development
- support
  - allocation of resources
  - real-time constraints (e.g. deadline, response time, etc.)
  - embedded constraints (e.g. power consumption, memory size)
- define behaviour
  - data-flow and control-flow
  - deterministic behavior modeling of RTES
- define structures
  - repetitive structures
  - component-based modeling for RTES hardware architectures.

Carleton UNIVERSITY

Analysis part (RTEA): Intent

- SPT plus extensibility to other analyses
- Harmonize S and P
- Support component-based models, by attaching measures to services offered or required at interfaces/ports.
- Point-to-point delays
- Support for state machine spec of behaviour.

Carleton UNIVERSITY

MARTE Core: Non-Functional Properties (NFPs)

- consider a measure, e.g. a certain response time
  - there are sample *realizations* of the measure if we run the system and monitor it
  - there can be *statistics* describing these realizations
    - mean, variance, histogram, 95th percentile, median, max
    - called the "statisticalQualifier"
  - there can be *required values* for these statistics
  - there can be *theoretical assumptions* about them
    - distribution of interarrival times
    - the choice of distribution type is a qualitative parameter
- there can be multiple values coming from different *sources*, such as required, provided, measured... and different statistical qualifiers
- define a property as:
  - property = ((value, units, statisticalQualifier, source, precision, direction),....)
- value can be a variable or an expression, stated in a *value specification language*

Carleton UNIVERSITY

NFP (2)

- e.g. properties of ExecHost (a processor) include
  - utilization:NFP_Real
  - commTxOvh:NFP_Duration
- NFP_Real is a type declared in a library of MARTE data types with fields
  - expr:VSL_Expression : a functional expression, which can also just be a value
  - source (est | meas | calc | req | undef)
  - precision:Real
  - statisticalQ (mean | variance | percentile ....)
  - direction (increasing | decreasing), for relationship to QoS
- NFP_Duration is a subtype of NFP_Real with also
  - unit: TimeUnitKind (us | ms | s | min | hr...)

Carleton UNIVERSITY

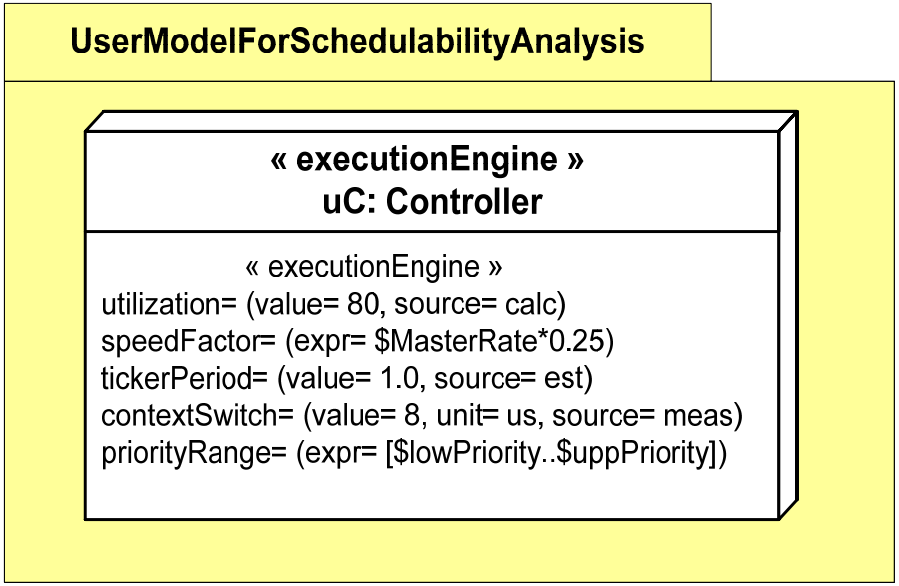
NFP (3) Examples

- Full expression uses the property names:
  - utilization = (expr=$AppProcUtn, source=calc)
  - commTxOvh = (value = 0.1, units = ms, source = est, statisticalQ = mean)
- Shorthand expression gives fields in the order defined:
  - utilization = (-, $AppProcUtn, calc)
  - commTxOvh = (0.1, -, ms, est, -, mean)
- VSL is a powerful language with value types that can be tuples and intervals [a..b],
  - including logical expressions that evaluate to Booleans,
    - V1= ((clients<6)?(exp(6)):1)
  - A context for variables and expressions can be defined to scope their use.
  - special syntax for time expressions including conditions

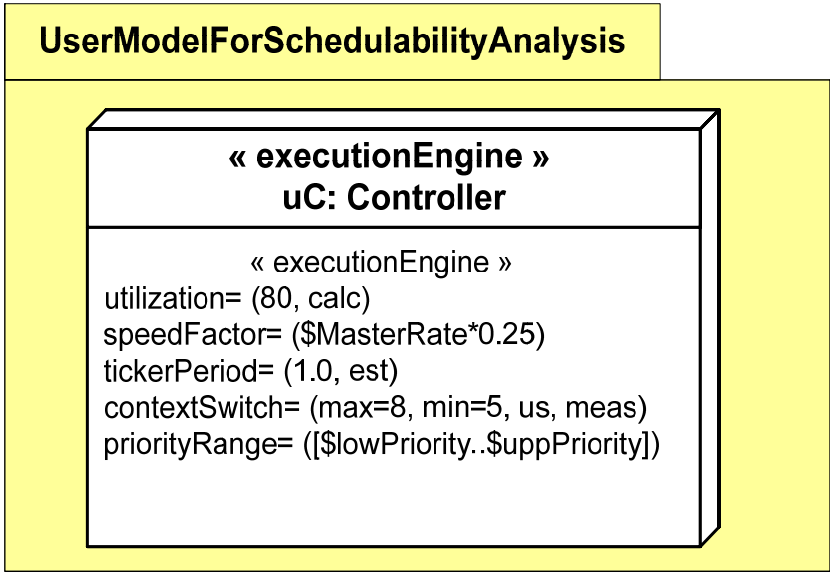
Carleton UNIVERSITY

Example of NFP annotations in a user model

a) Extended Notation

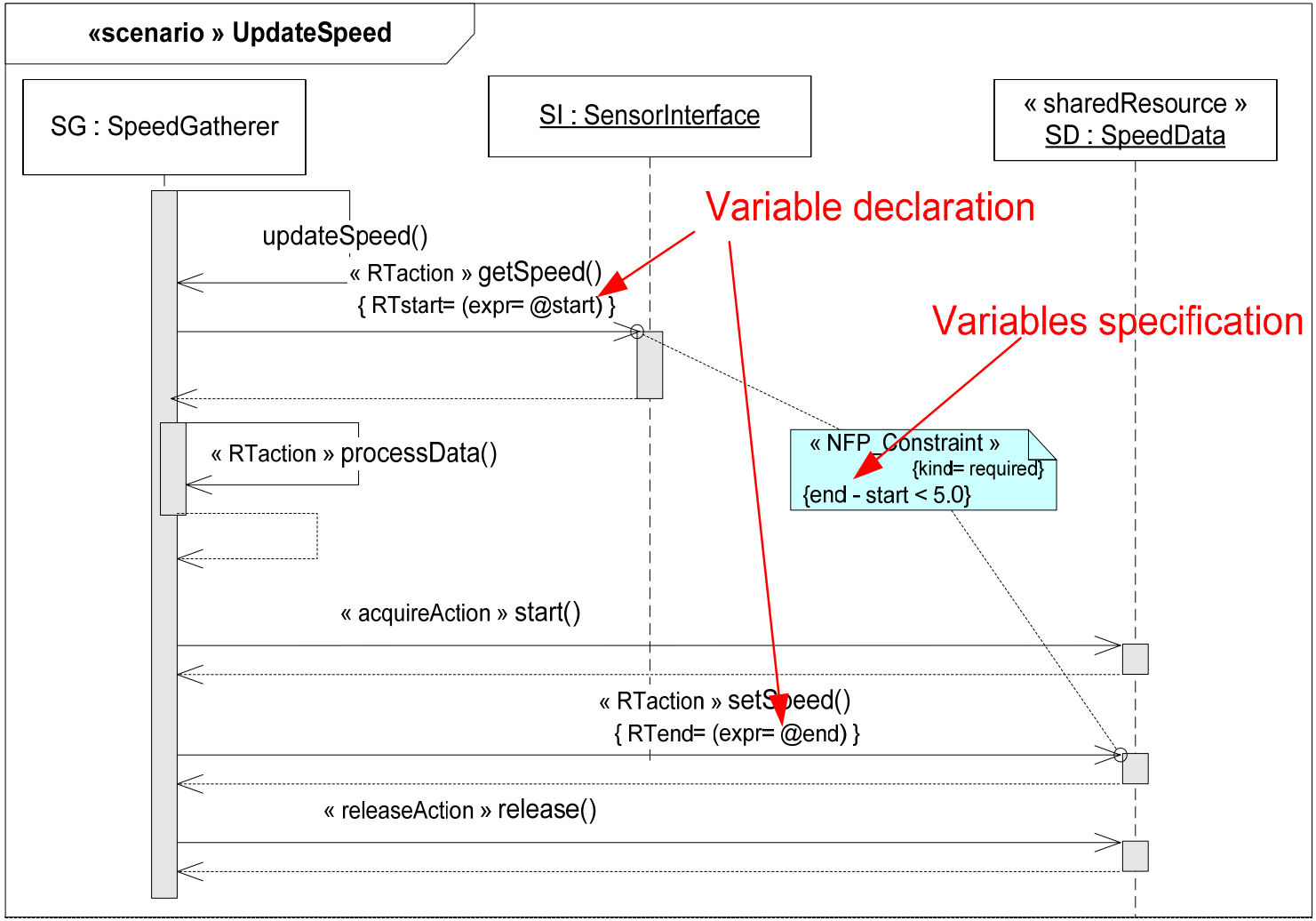


b) Reduced Notation



Carleton UNIVERSITY

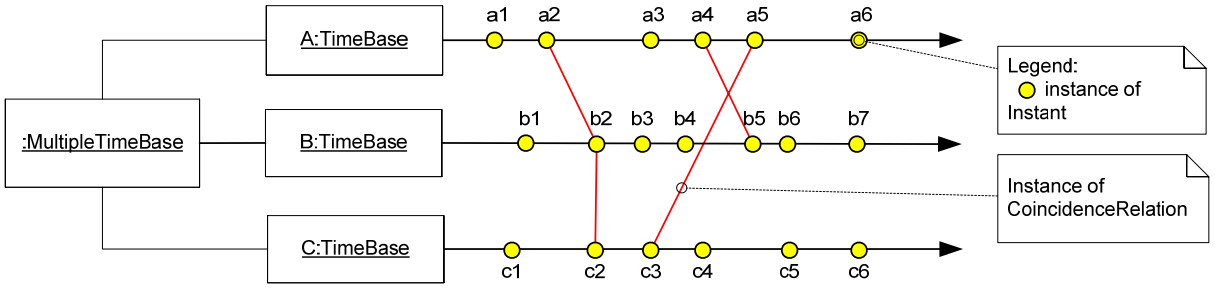
Example of NFP annotations in a Constraint



Carleton UNIVERSITY

MARTE Core: Time Modeling

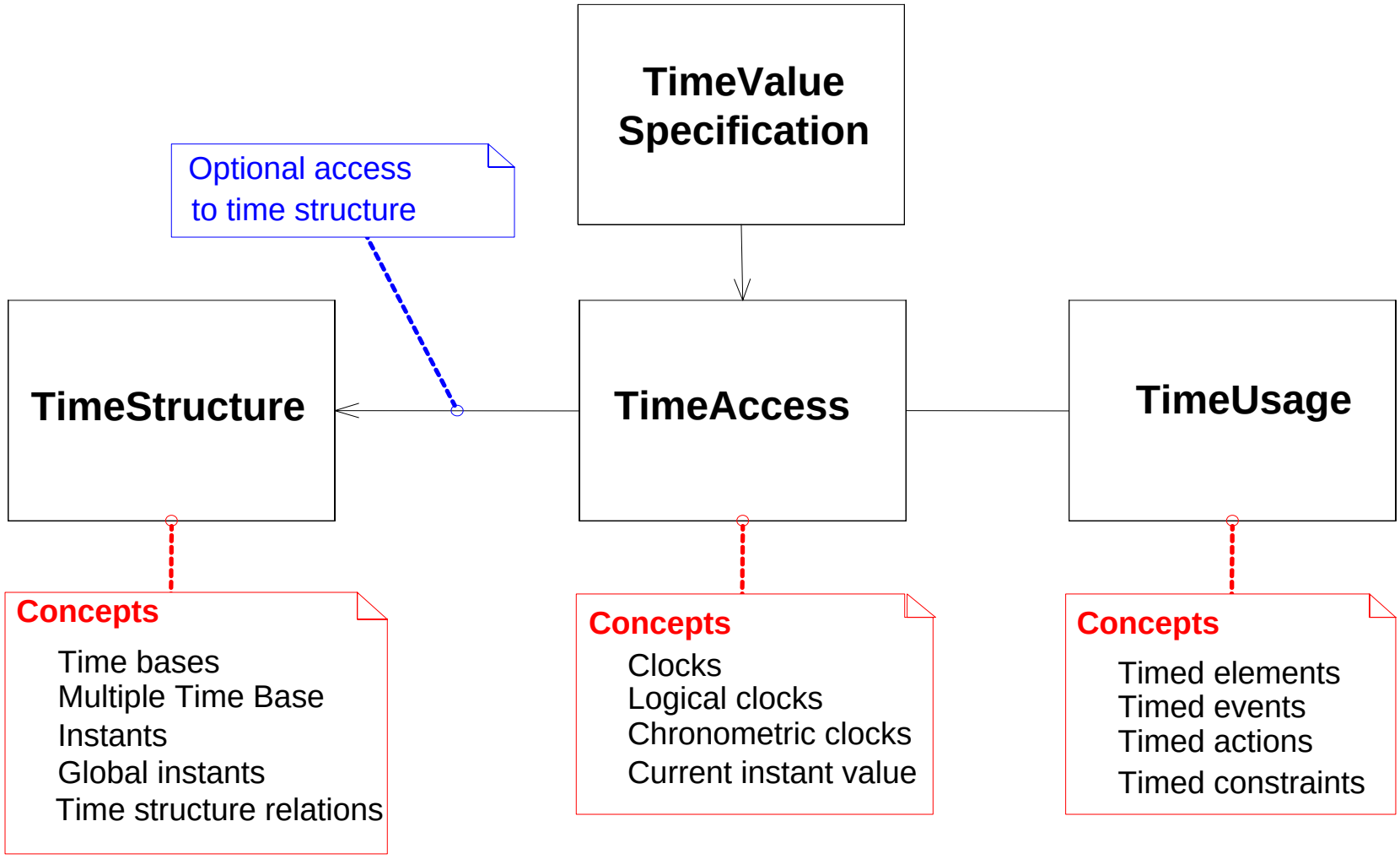
- Defines a set of time-related concepts and their semantics
  - time base, instant, duration, clock, time value, timed event, timed action



- Covers 3 kinds of time models
  - Causal/temporal time model
    - concerned only with precedence/dependency
  - Clocked/synchronous time model
    - divides the time scale in a discrete succession of instants
    - rich causal dependencies can take place *inside* an instant
  - Physical/real-time time model
    - defines precise accurate modeling of real-time duration values

Carleton UNIVERSITY

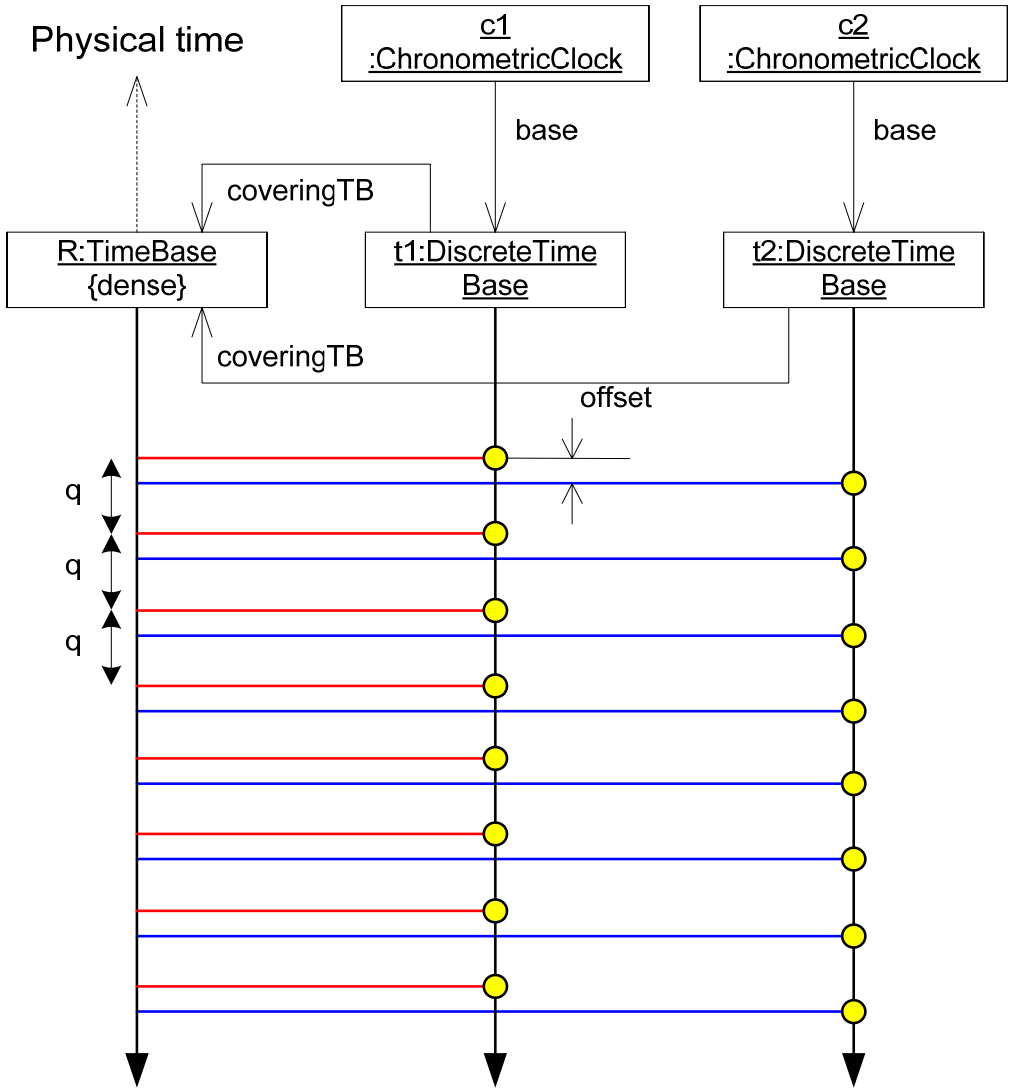
Schema of the time modeling concerns



Carleton UNIVERSITY

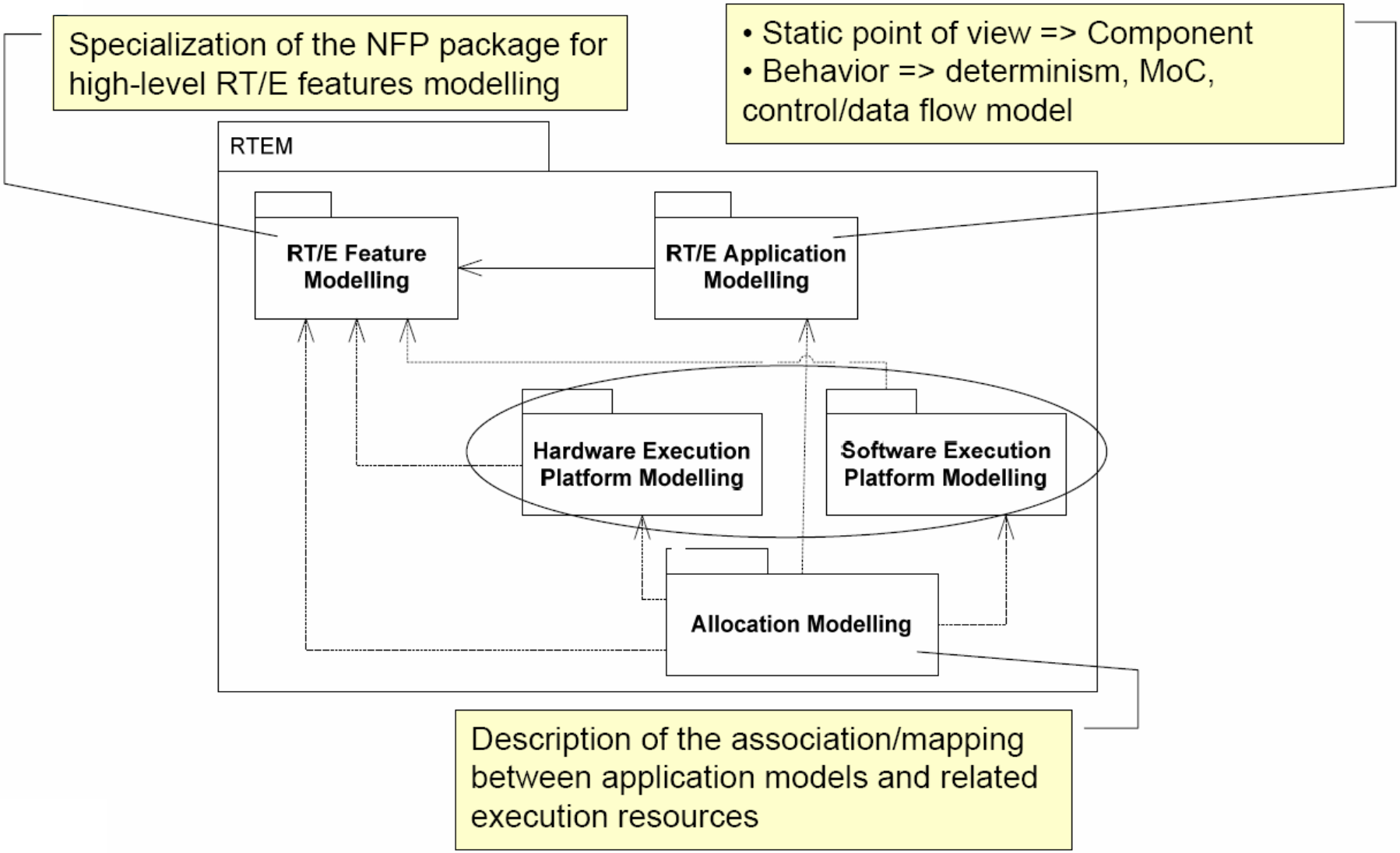
Example: dependency of clocks on physical time

- Different clock NFPs in function of the physical time:
  - resolution
  - stability
  - offset
  - skew
  - drift
- The figure represents examples of:
  - resolution q
  - offset between two clocks



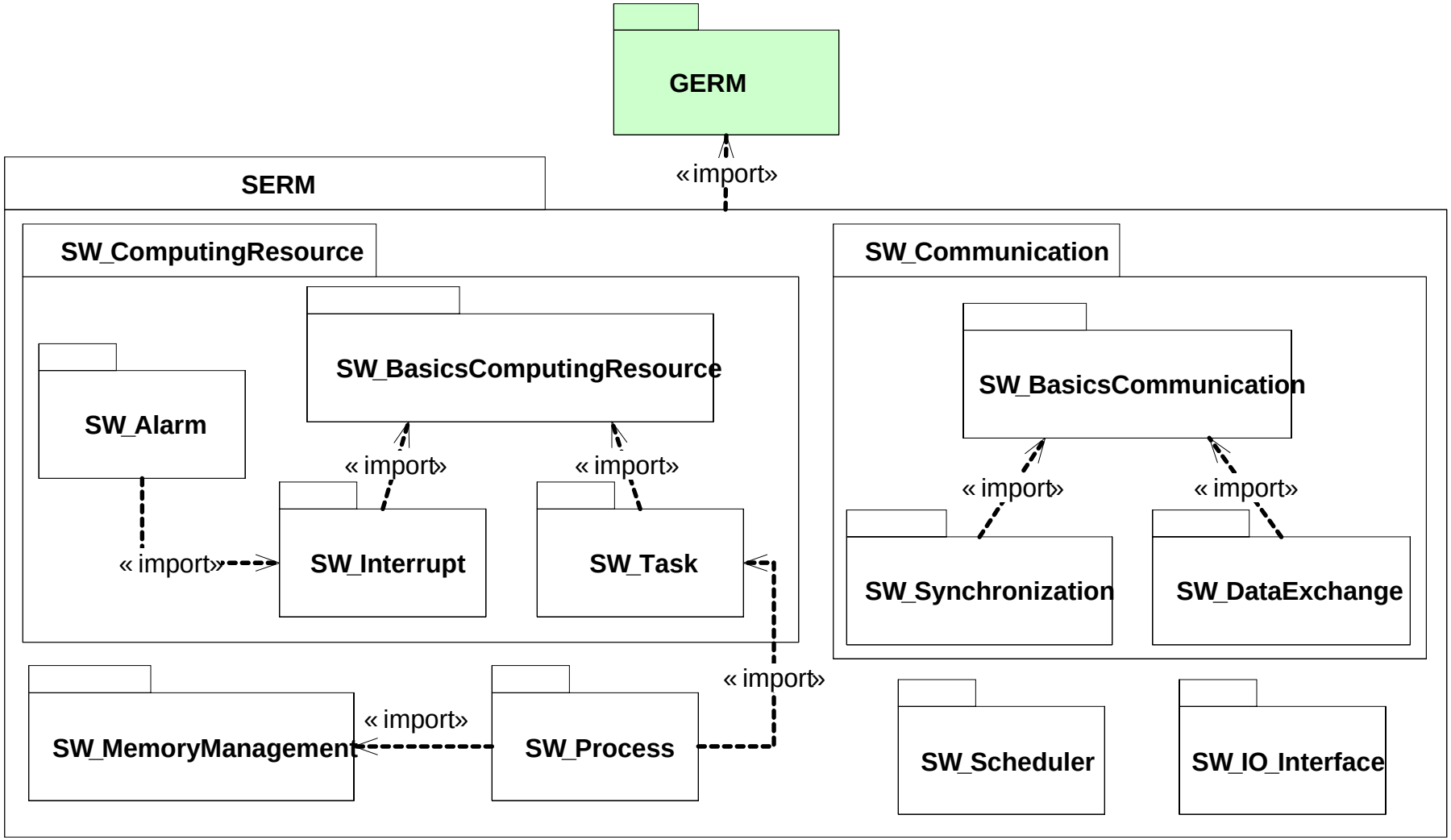
Carleton UNIVERSITY

Real Time Ex Modeling: Structure



Carleton UNIVERSITY

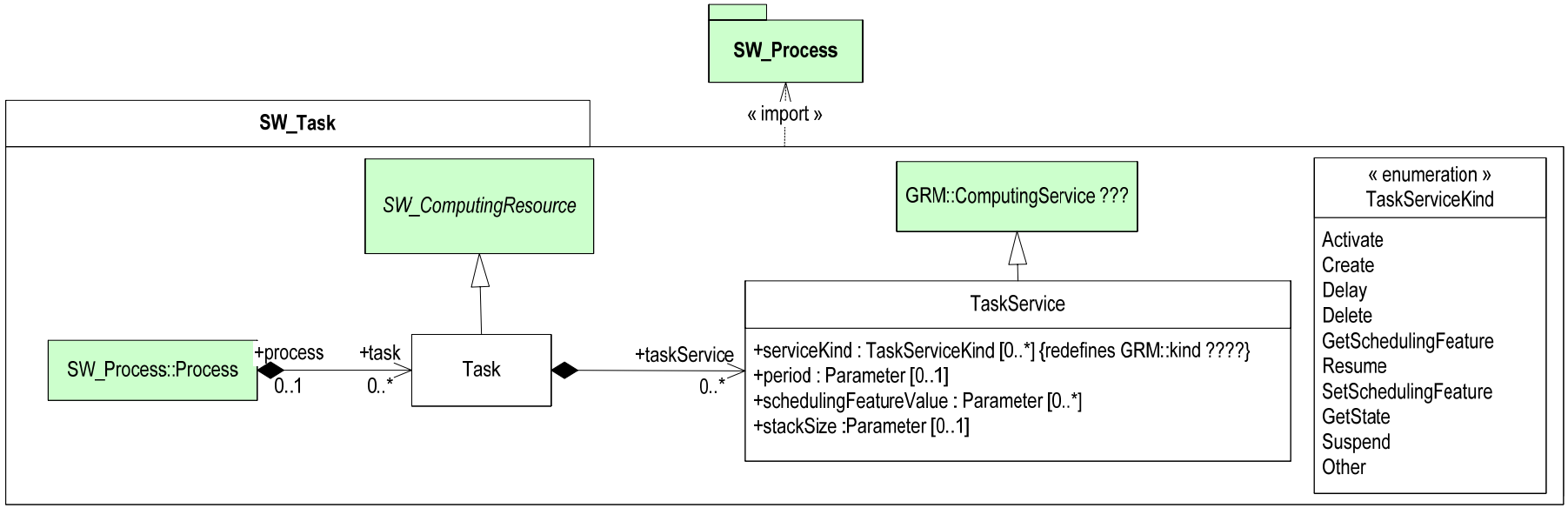
Structure of Software Ex Res Model



Carleton UNIVERSITY

Example: SW_Task package

- Tasks are basic resources for real-time multitasking kernels.
- A Task denotes an encapsulated sequence of actions and executes concurrently to other tasks.
- Tasks share the same address space but preserve their own contexts (program counter, registers, signal mask, stack, etc).
- In this model, there is no distinction between tasks and threads



Carleton UNIVERSITY

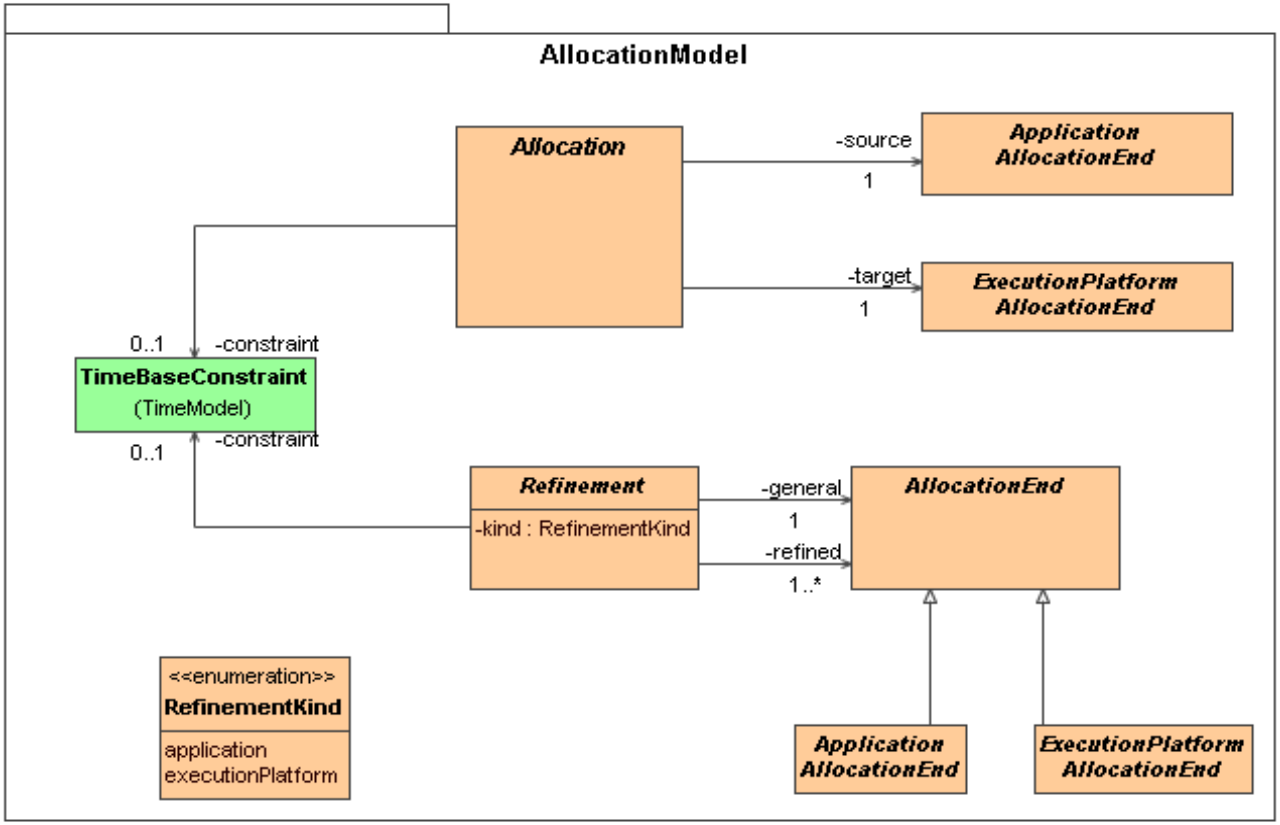
Structure of Hardware Ex Res Model

- Logical view
  - Functional taxonomy of HW
  - 4 kinds of resources:
    - Computing resources
    - Storage resources
    - Communication resources
    - Supporting resources
  - Each kind is refined in its package
- Physical view
  - HW classification in terms of
    - Voltage, area, dissipation…
    - Chips, cards, channels…

Carleton UNIVERSITY

Allocation Model

- Based on SysML concept of allocation from an application to a platform
- Allocation and refinement relations can be annotated by TimeBaseConstraints
  - allocations provide links between independent models,
  - refinement/abstraction change the focus on an underlying similar structure.

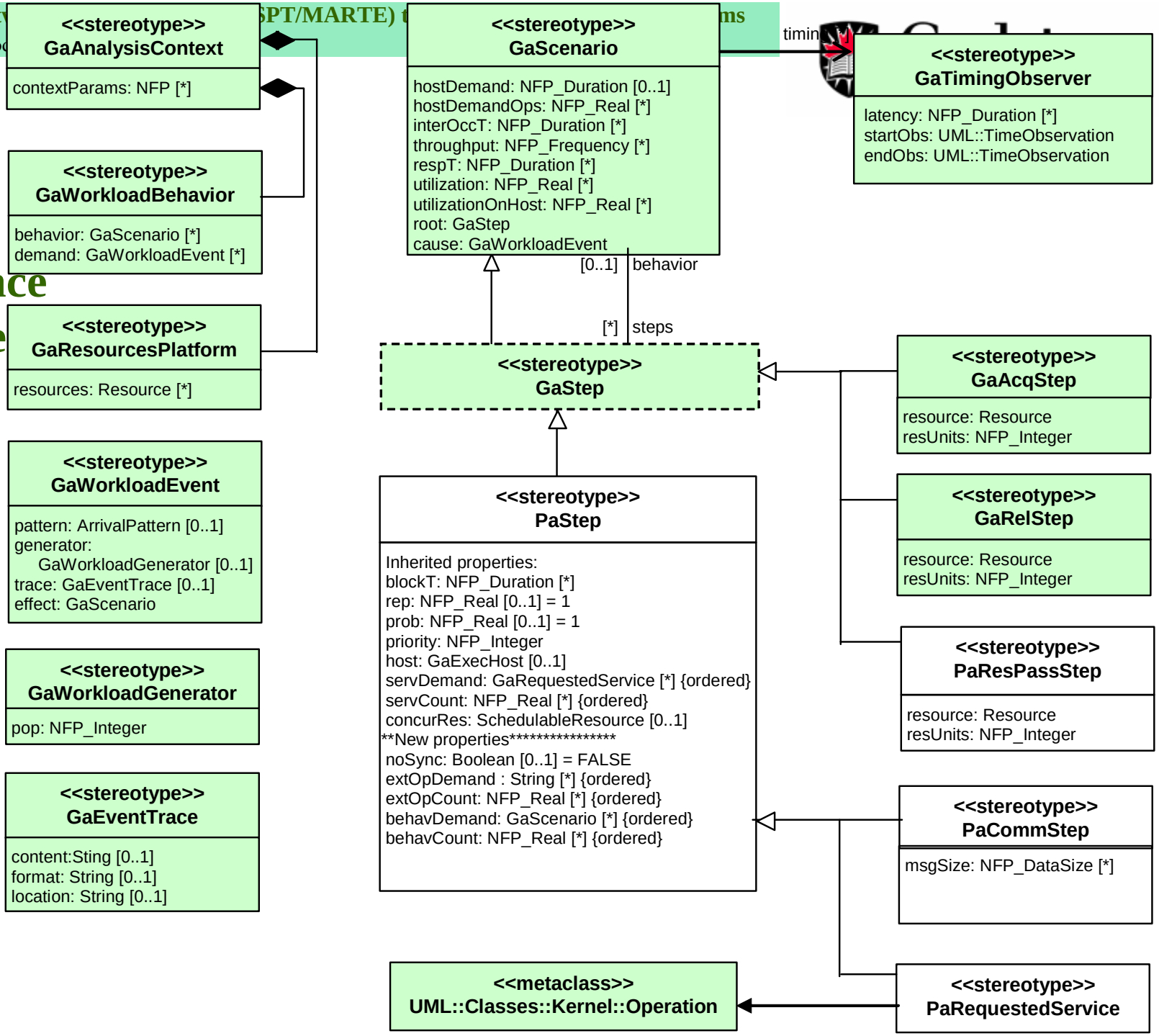


Carleton UNIVERSITY

Analysis Modeling

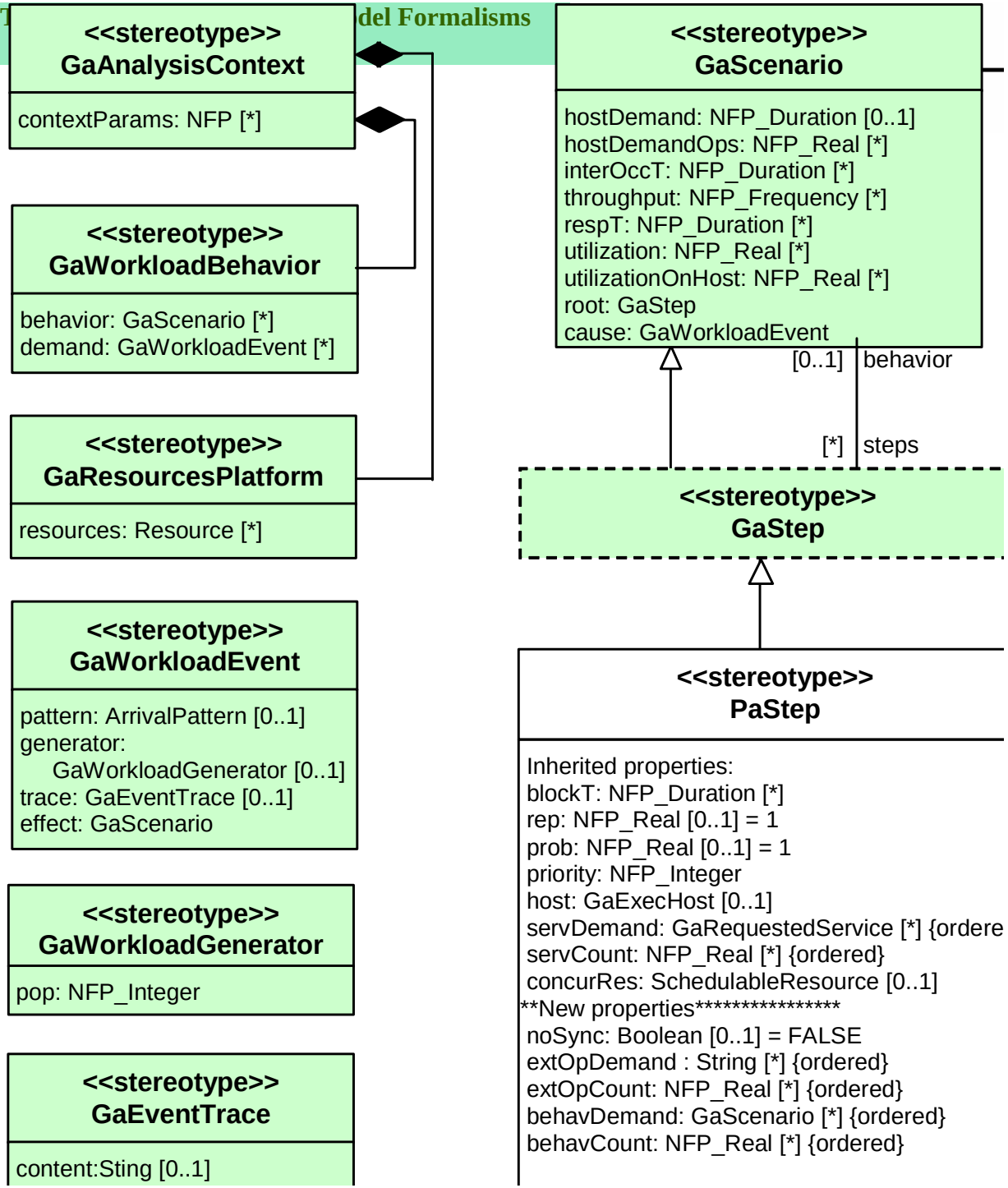
- a Generic analysis (Ga) subprofile for common concepts
- AnalysisContext with parameters scoped to the diagram
- Scenario, Step
- AcqStep, RelStep to acquire and release resources explicitly
- ExecHost, CommHost (for links)
- WorkloadEvent for workload (as an event stream)
- Resources from the General Resource model
  - SchedulableResource is a process
  - ExecHost, CommHost
  - CommChannel is a kind of SchedulableResource for communications
- Performance and Schedulability sub-profiles extend this

Performance sub-profile for behaviour

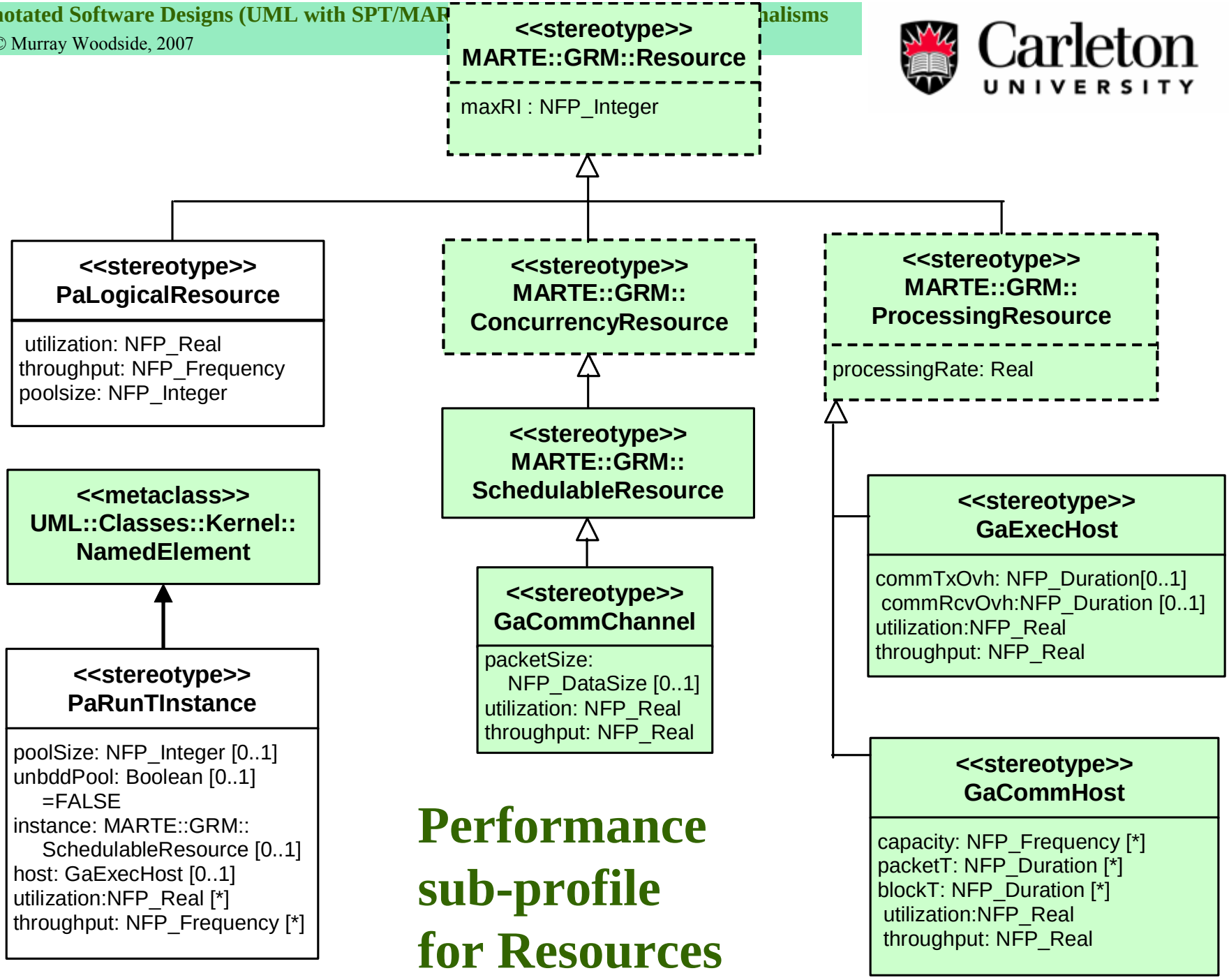


Detail...

Workload: define just one of (pattern | trace | generator). Closed and open workloads are pattern properties:
closed(population=NFP_Integer, extDelay = NFP_Duration)
open(interArrT = NFP_Duration)
generator = StateMachine
trace = String for filename,etc



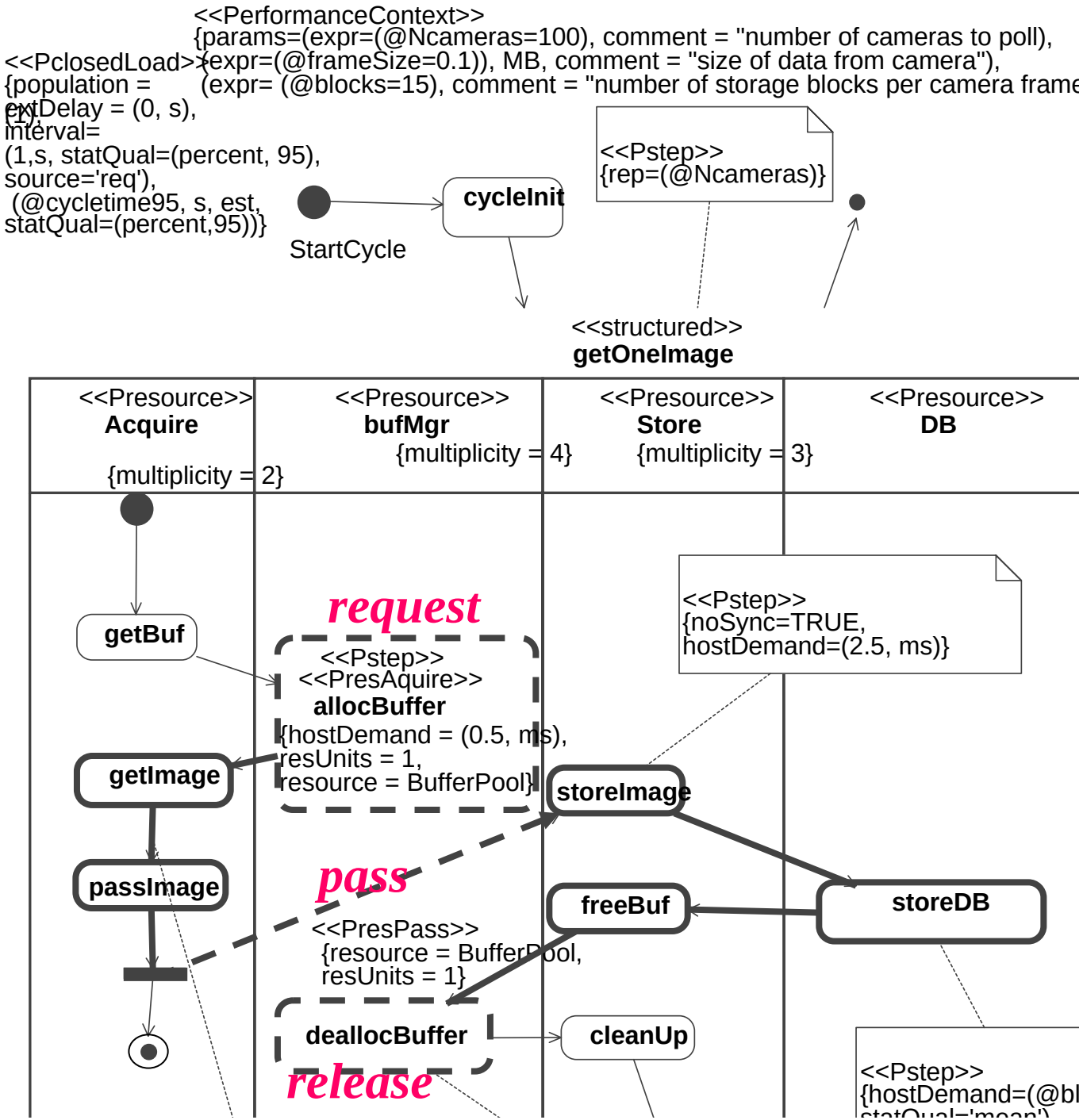
Carleton UNIVERSITY



Performance sub-profile for Resources

New Feature: Operations on a Logical Resource

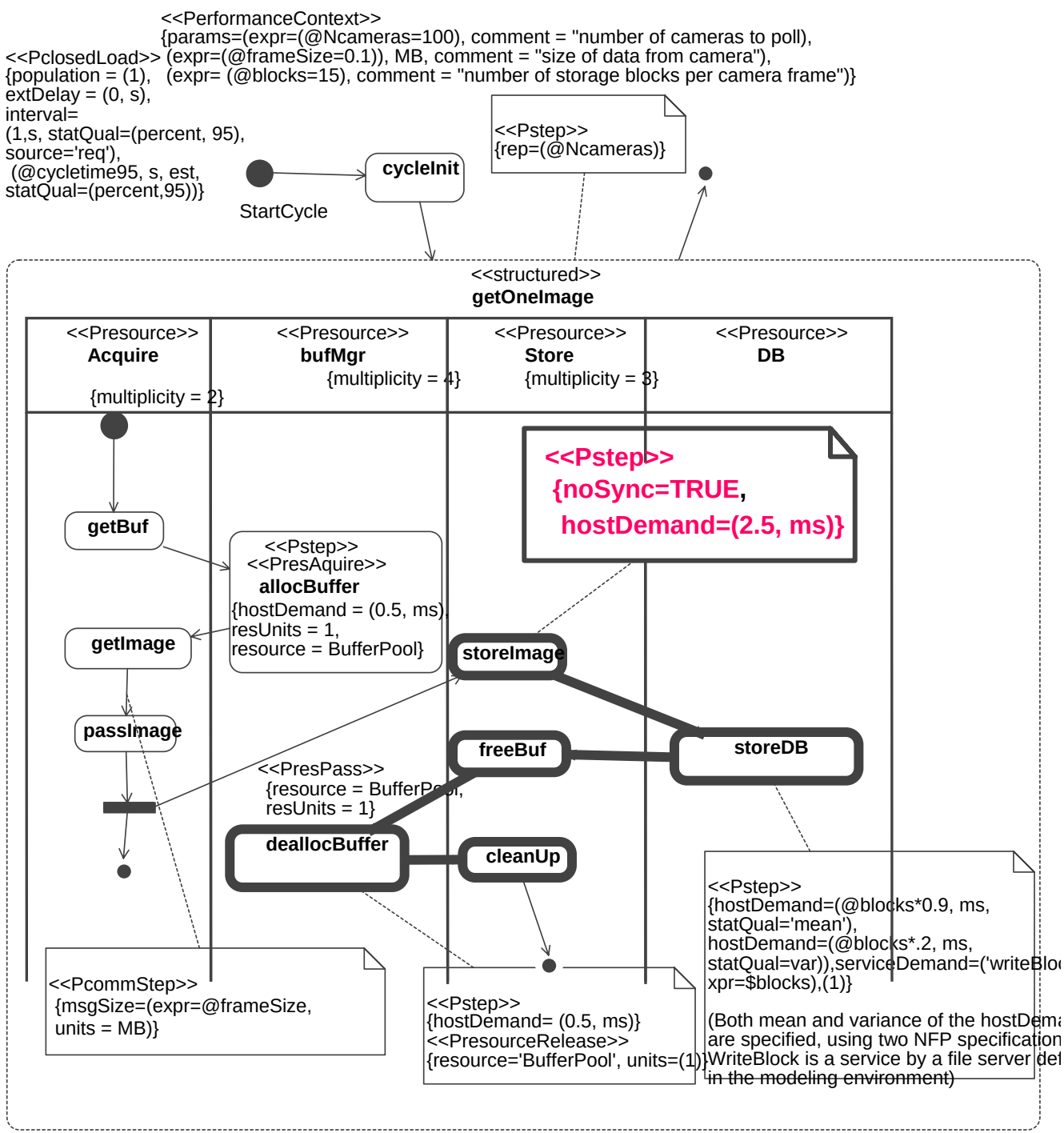
the heavy line shows what happens during one buffer occupancy



New Feature: Non-Synchronized Branch

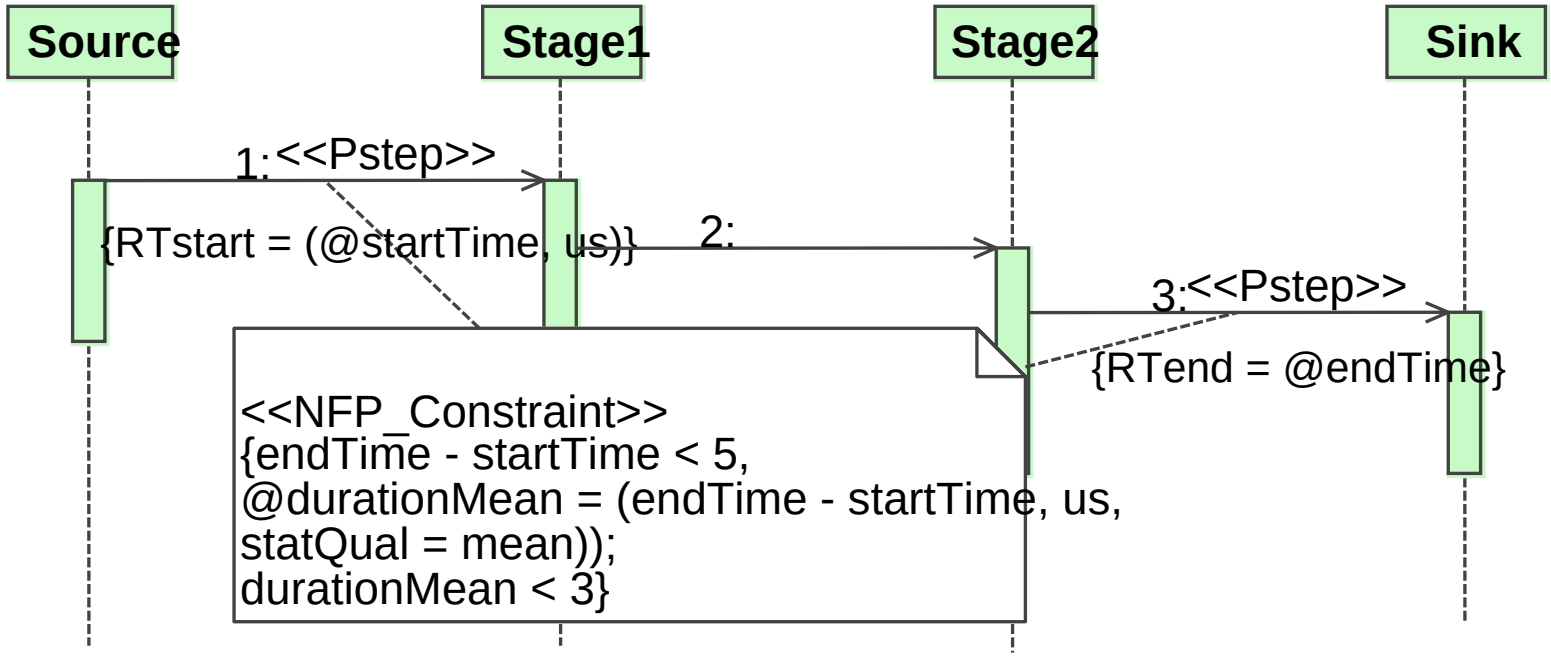
...after a fork, indicating there is no join

... more necessary in par blocks of SD



Carleton UNIVERSITY

New Feature: Interval Durations



- attributes for start and end events of a Step
- derived functions defined in a constraint block, or globally
- here: used in functions in a constraint block that includes NFPs

Carleton UNIVERSITY

Services and Service Requests

- «PaService» stereotypes an operation that forms part of a Step
  - Step has a demand attribute which is a tuple
    - servDemand:Operation, (may be a tuple)
    - servCount:NFP_Real (corresponding order)
  - the demand is the mean number of requests made by the Step.
- Services couple scenarios to calls and call hierarchies
- two additional forms:
  - an *external operation* is defined outside the UML model
    - extOpDemand:string
    - extOpCount:NFP_Real
  - *direct service definition* by a scenario not necessarily associated with an object
    - behavDemand:Scenario,
    - behavCount:NFP_Real

Carleton UNIVERSITY

Services: options

- standalone delay

DataBase
readDB «PaService» {hostDemand= (13.2, us, statisticalQ=mean)}

«PaStep» {servDemand = (readDB, 3.7), servCount = (1.7, mean)}

- service with demands

DataBase
readDB «PaService» {..... demands}

- ref to external operation definition

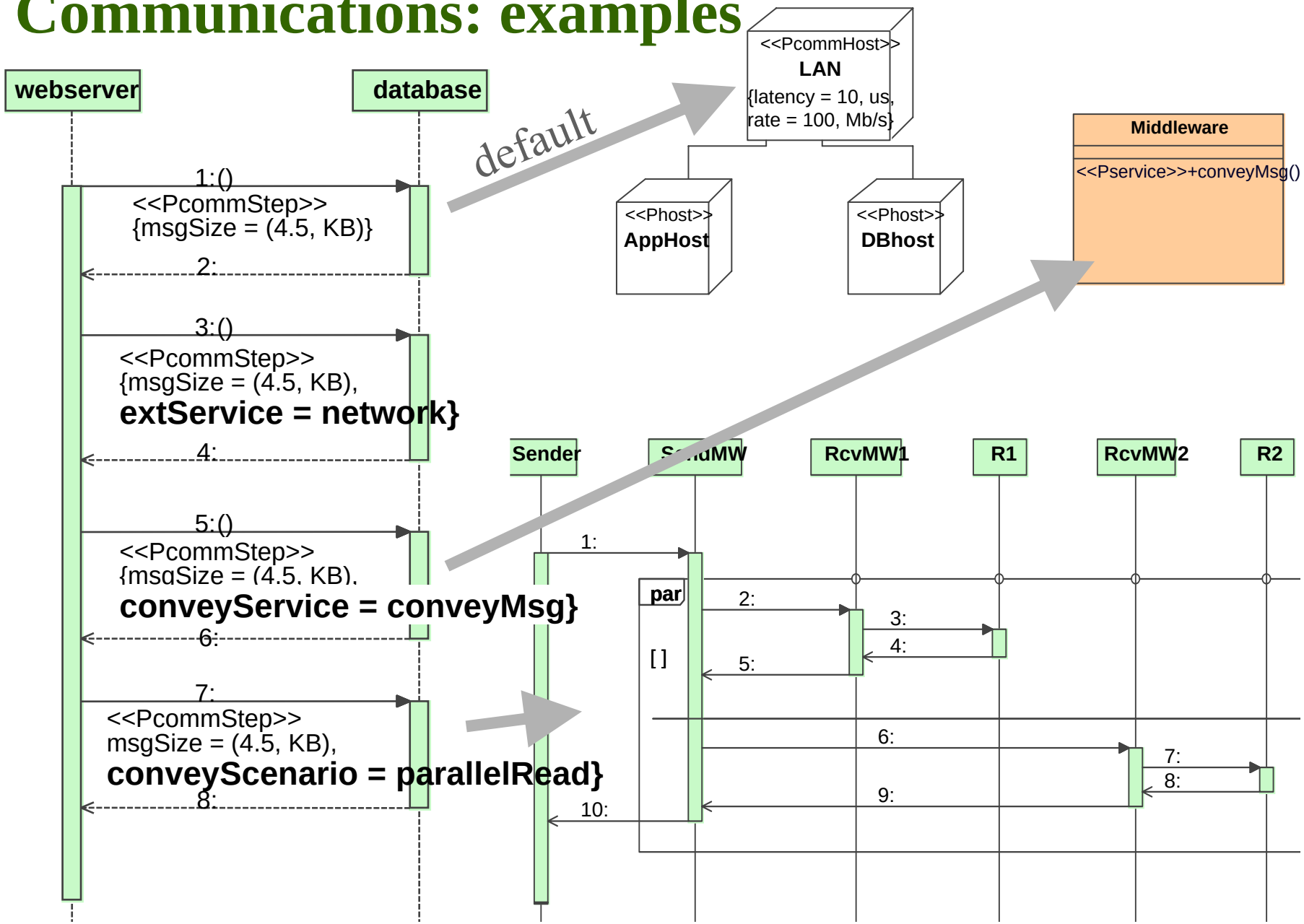
Carleton UNIVERSITY

State Machine Annotations

- a state or a transition can be a Step, with workload and service-demand attributes
  - service-demands can represent the functional action semantics
    - local method calls
    - remote service requests
    - can include a Scenario for complex actions on a transition, including communications actions
- SM is only defined for a *class*, so
  - all instances must have the same parameter values, OR
  - instances must be defined separately at the class level, OR
  - *variables* can be used to identify parameters, and a table can set values for different instances
- inter-object messages are represented by signals only
  - either broadcast or with a specified target object
  - communication host is inferred from hosting of the two SMs.

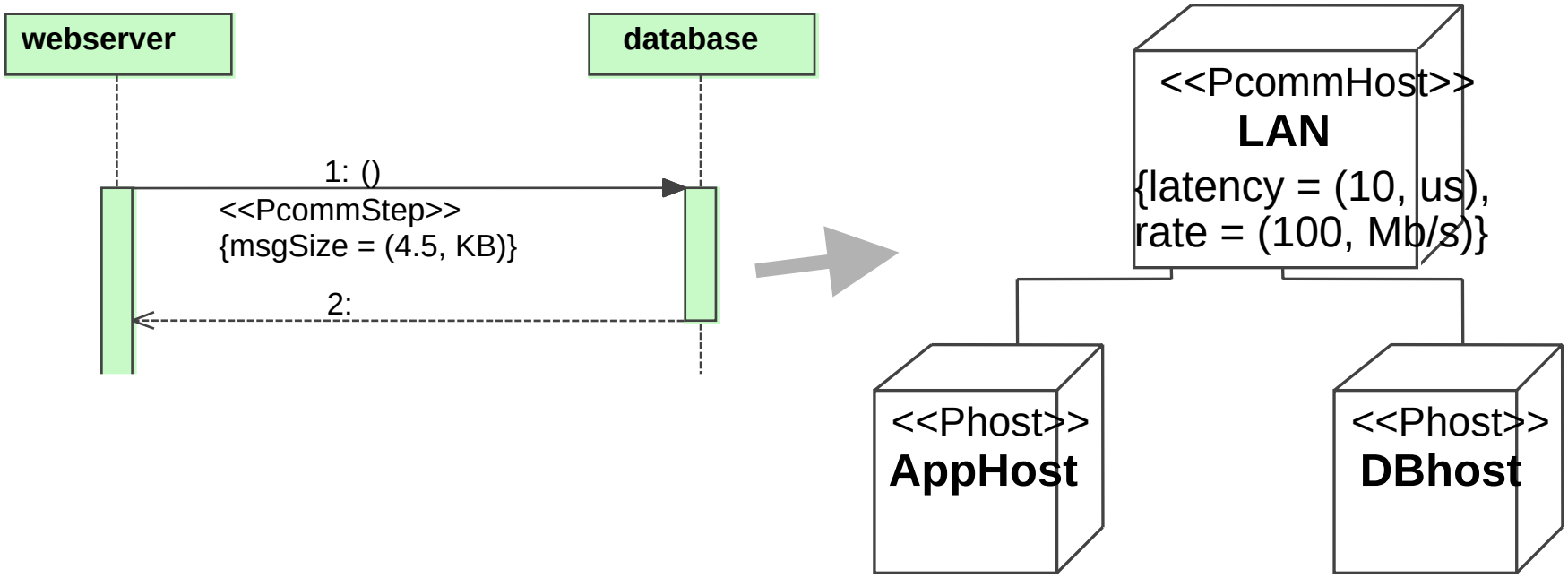
Carleton UNIVERSITY

Communications: examples



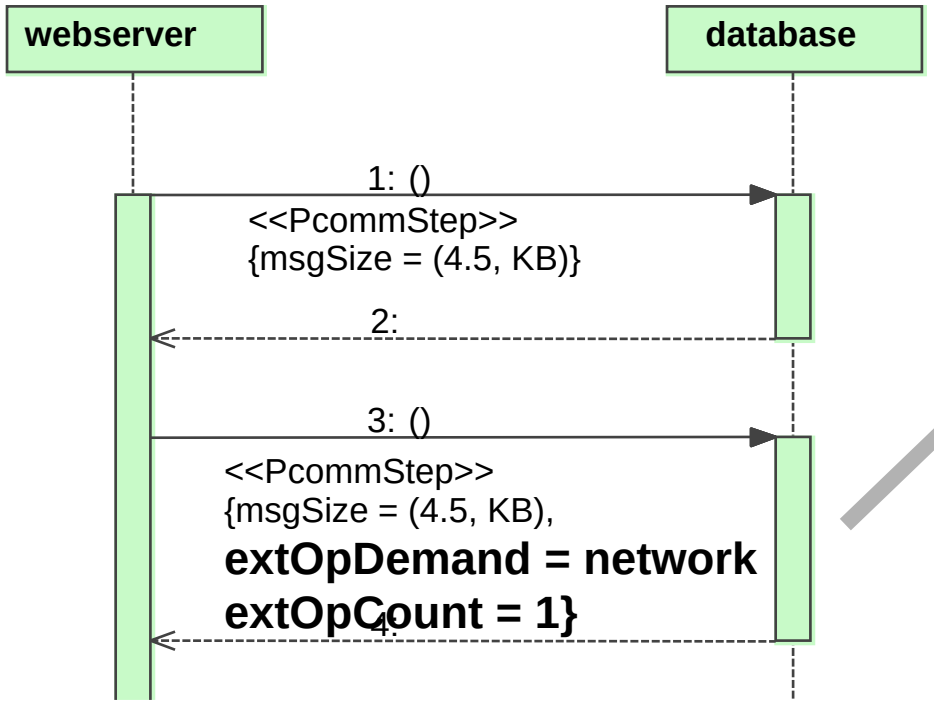
Carleton UNIVERSITY

Communications example: default



- the performance model uses the LAN characteristics to create a server submodel,
- and the Hosts have overhead CPU demand parameters (not shown) which are added to the workloads at sender and receiver

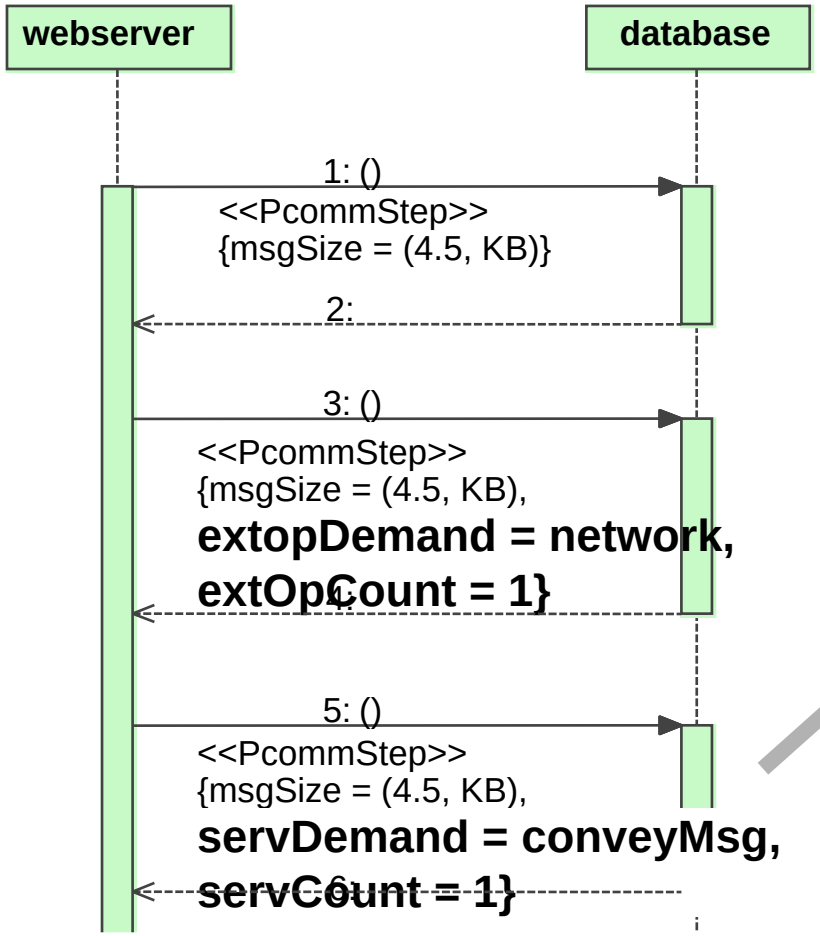
Communications example: External submodel

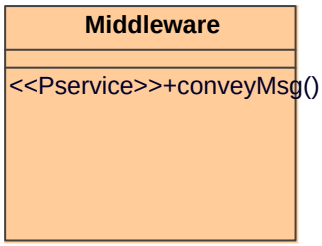


- "network" identifies an operation by a network submodel to the modeling tool, which inserts it into the final performance model, using the message size
- Host overhead parameters are still used, governed by message size

Carleton UNIVERSITY

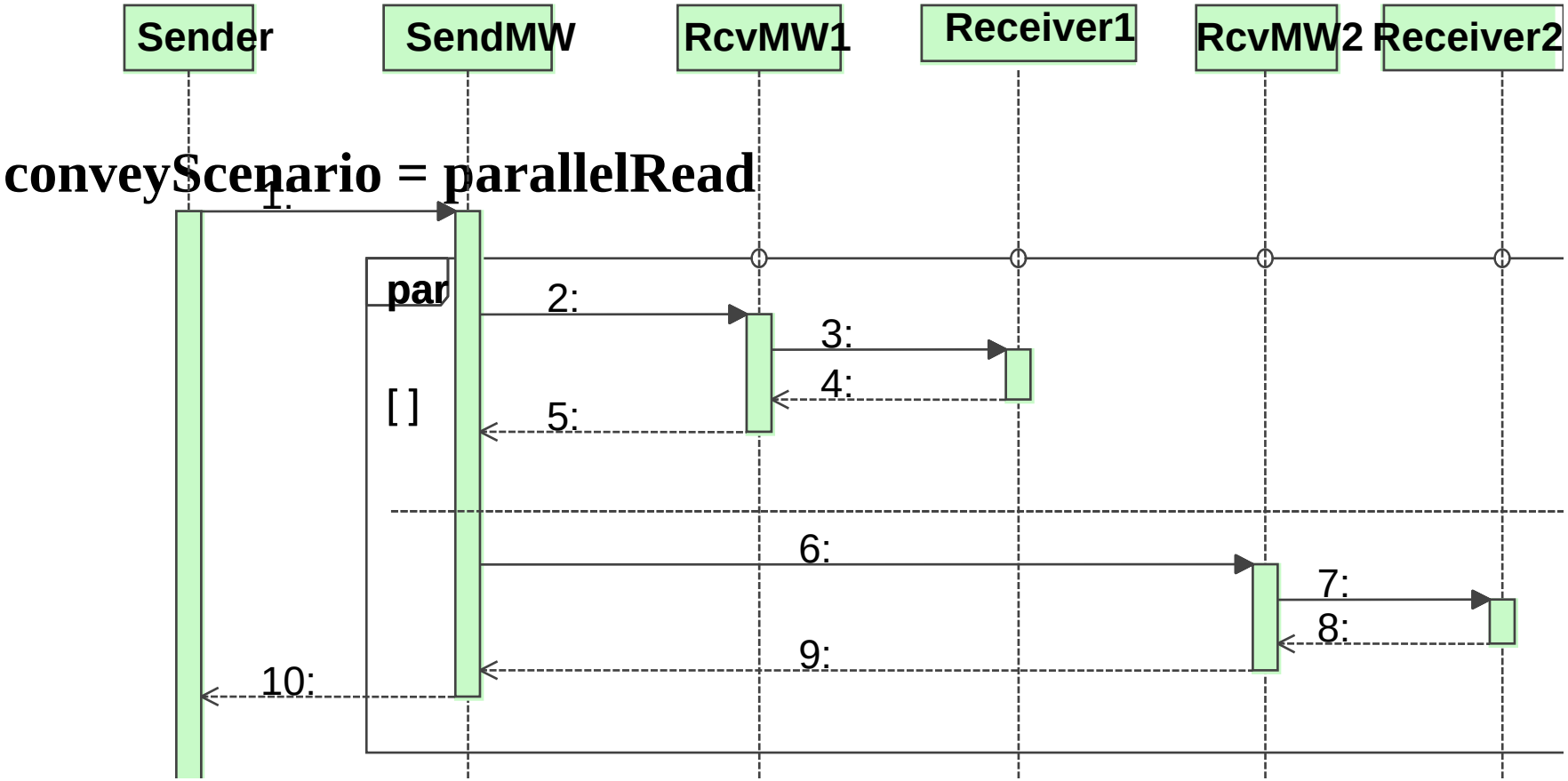
Communications example: Middleware Service





- "Middleware" can be a structured class defining a subsystem
- conveyService is an *operation* of that class
- it defines its own overheads, so default Host values of overhead are not used

Communications example: Scenario



- arbitrarily complex sequences of behaviour
  - good for protocol detail executed at both ends and at servers
- multiple deliveries in this case must be interpreted:
  - receiver behaviour is nested into the Receiver step(s)

Carleton UNIVERSITY

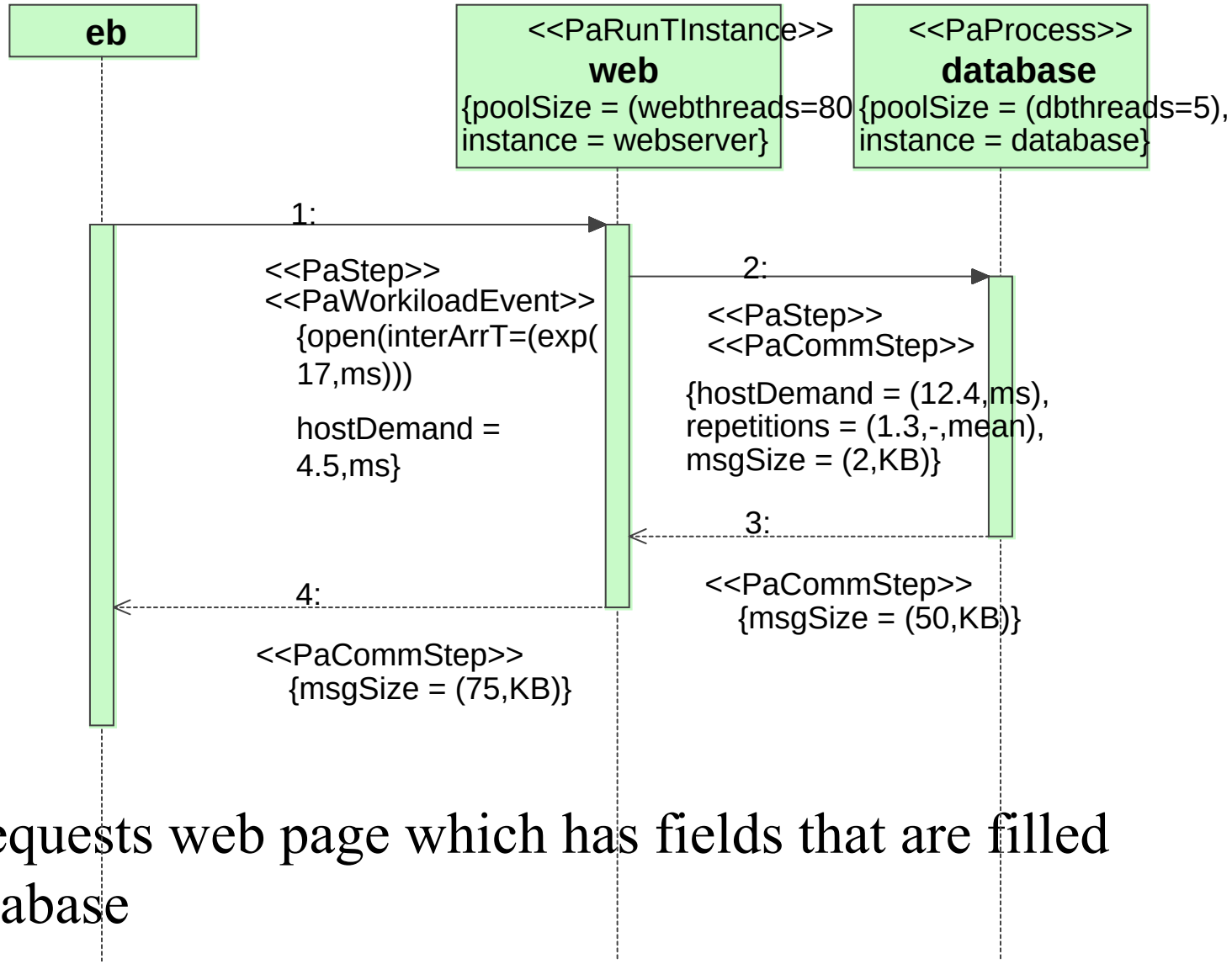
5. How MARTE can be applied

Now, a series of examples to nail down how the annotations work in practice, and how they relate to performance models

1. simple web interaction... review
2. TPC-W electronic bookstore
3. Embedded: building security system
4. Communications parameters
5. Communications layer
6. Component-based calling hierarchy
7. State machine to define a workload

Carleton UNIVERSITY

Example 1. Simple web interaction

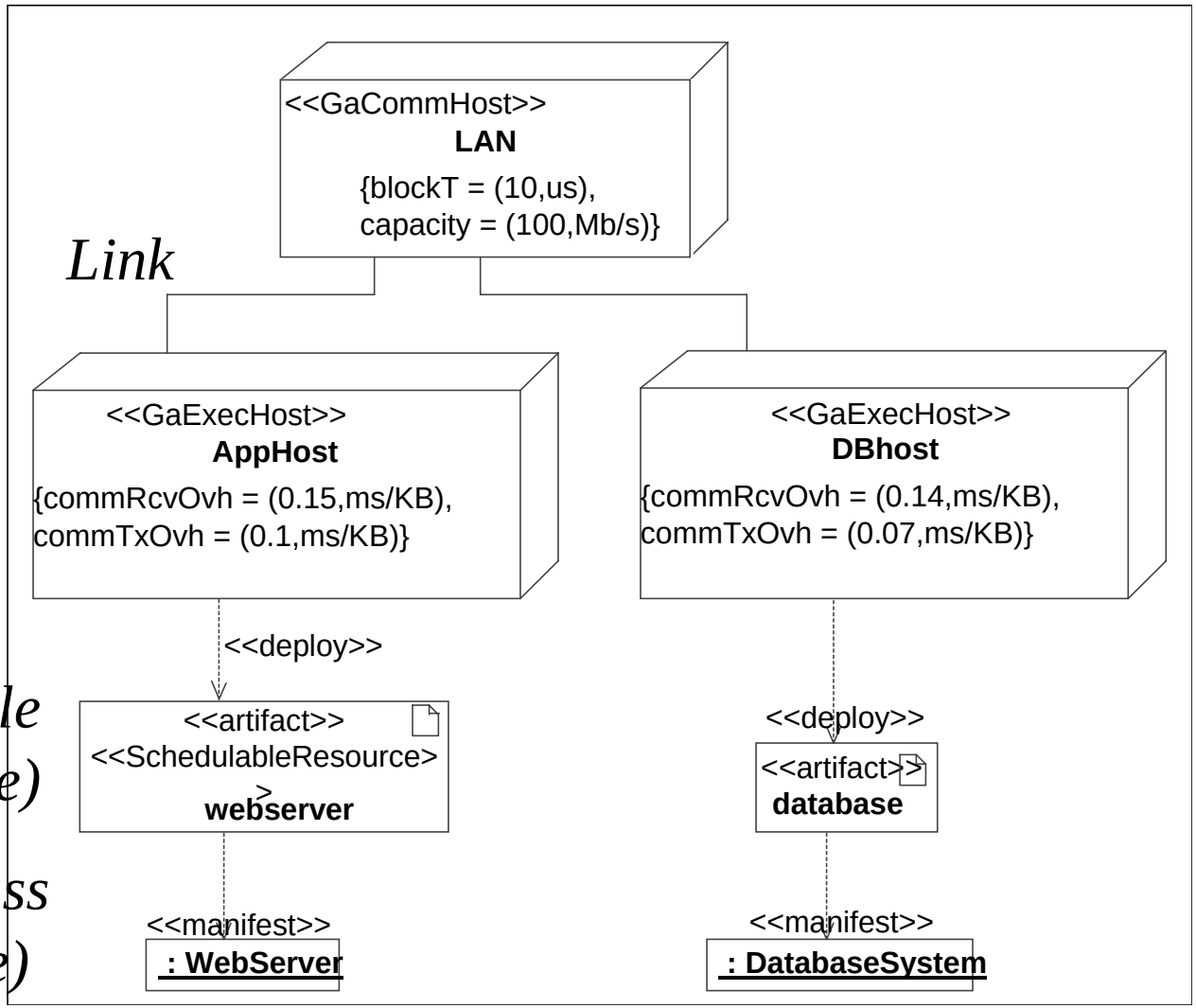


- browser requests web page which has fields that are filled from a database

Carleton UNIVERSITY

Deployment for web interaction

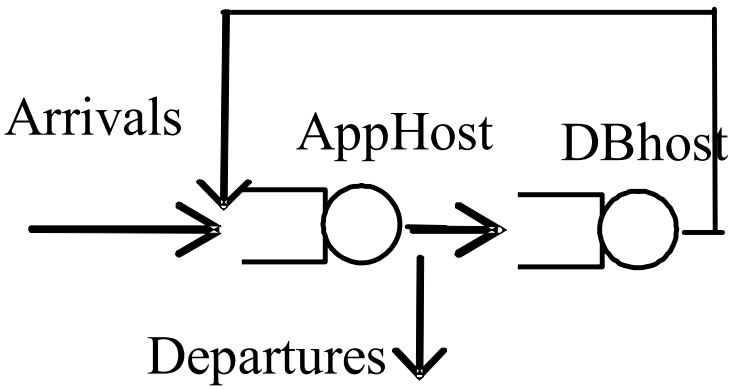
- showing just the webserver and database



Carleton UNIVERSITY

Queueing network model of web interaction

Solution follows from demand values:



AppHost demand/response:

4.5	hostDemand
0.1*(2 + 75)	TxOvh
0.15*50	RcvOvh
19.7 ms/response	

DBHost demand/response

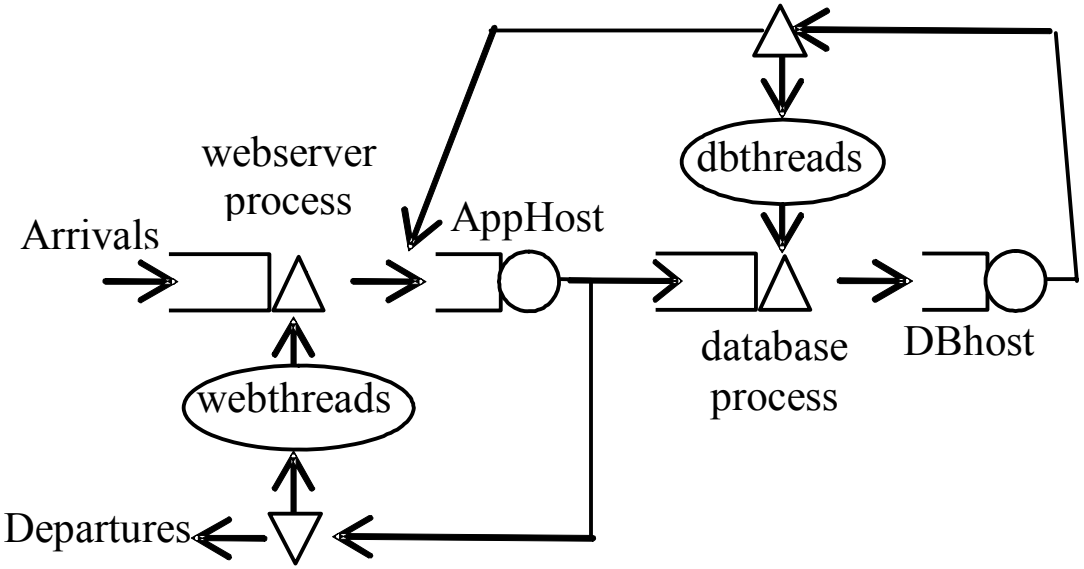
1.3 repetitions times:

12.4	hostDemand
1.4*2	RcvOvh
0.07*50	TxOvh
16.18 ms/visit	

21.03 ms/response

Carleton UNIVERSITY

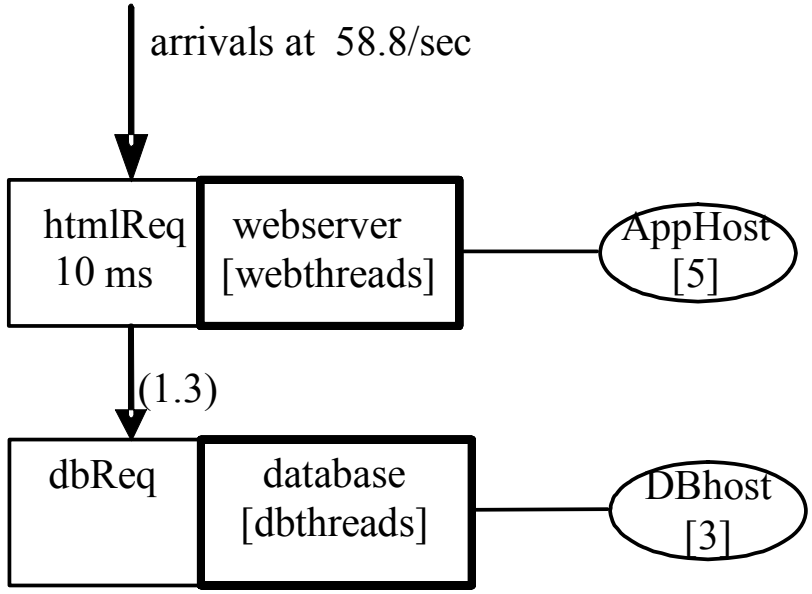
Extended queueing network (EQN)



- able to model the webserver and database thread resources as token pools
- needs service times
- create submodels for EQN solution techniques

Carleton UNIVERSITY

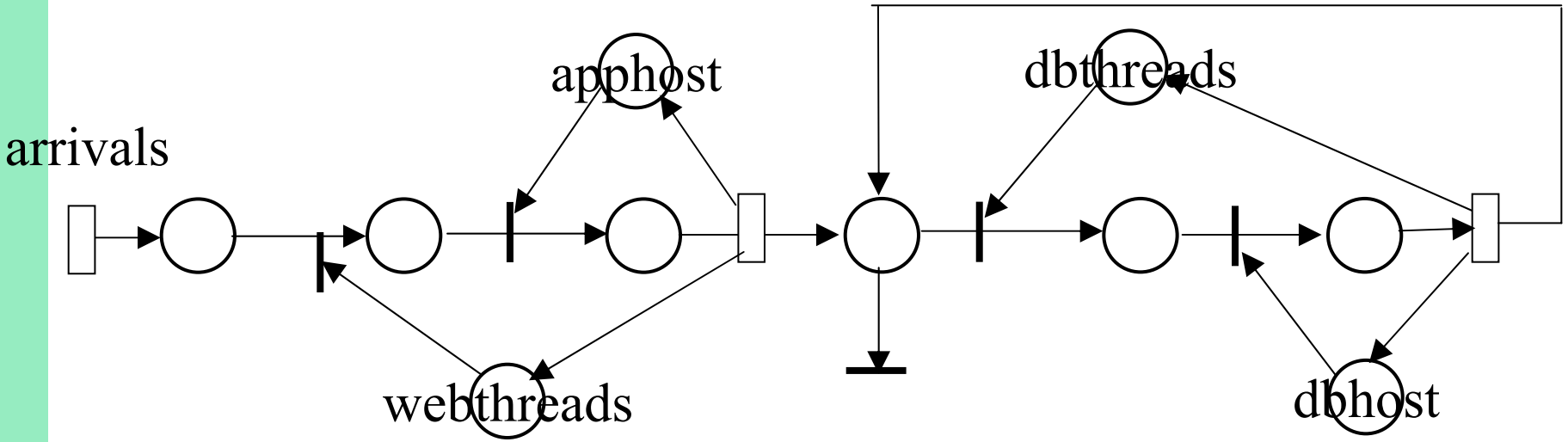
Layered queueing version of the web interaction EQN



- same model semantics
- LQN solver creates the sub-models automatically and solves by surrogate delays

Carleton UNIVERSITY

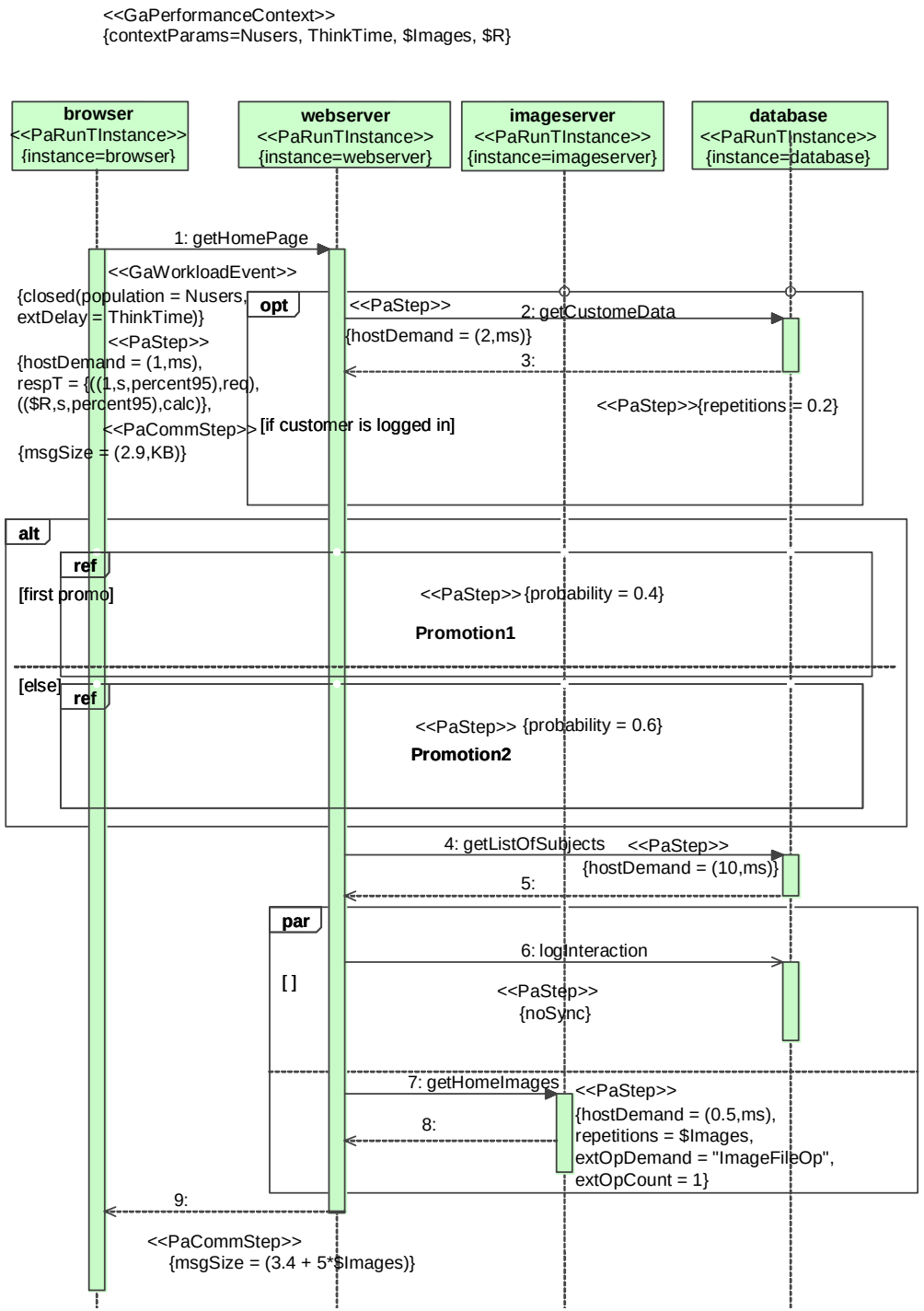
Petri net model of the web interaction



- arrival process must be made finite to get a finite state space
  - a pool of N customers who arrive and leave

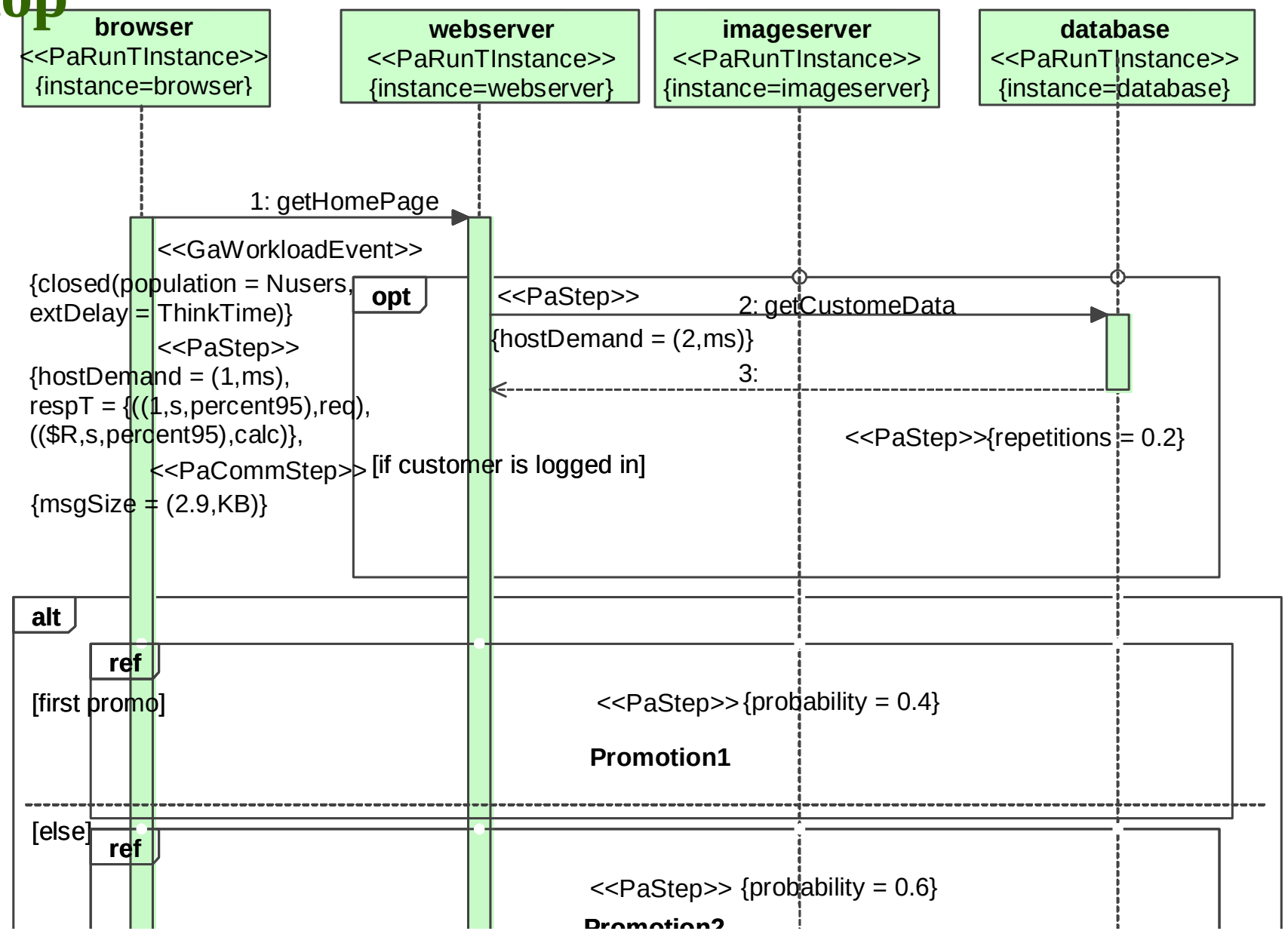
Example 2: TPC-W Bookstore

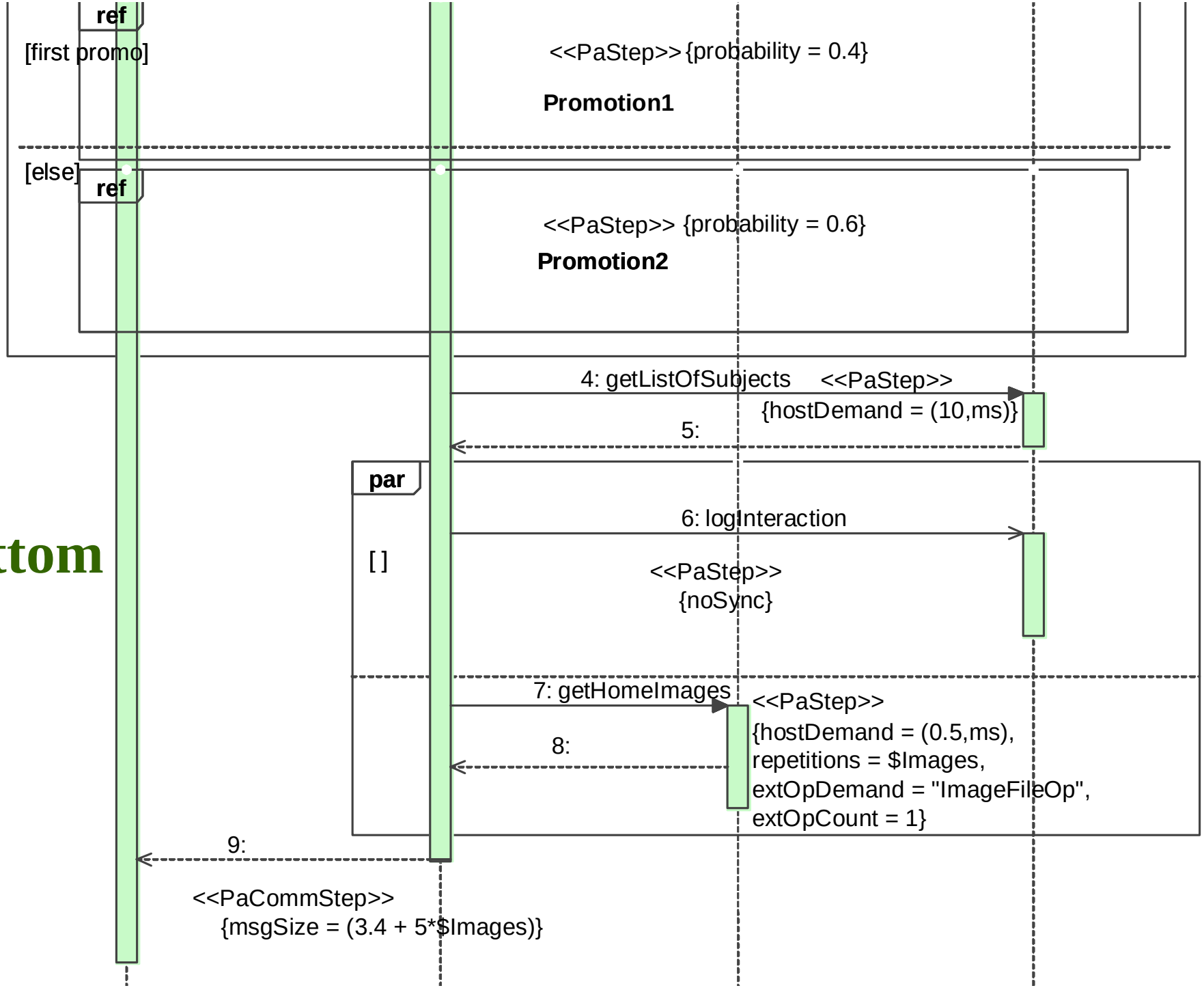
- this is the GetHomePage interaction
- one of 14 scenarios specified in the benchmark standard
- each page operation has a set of successor operations, which can be specified by a state machine
- close look at blow-up...



<<GaPerformanceContext>>
{contextParams=Nusers, ThinkTime, $Images, $R}

top

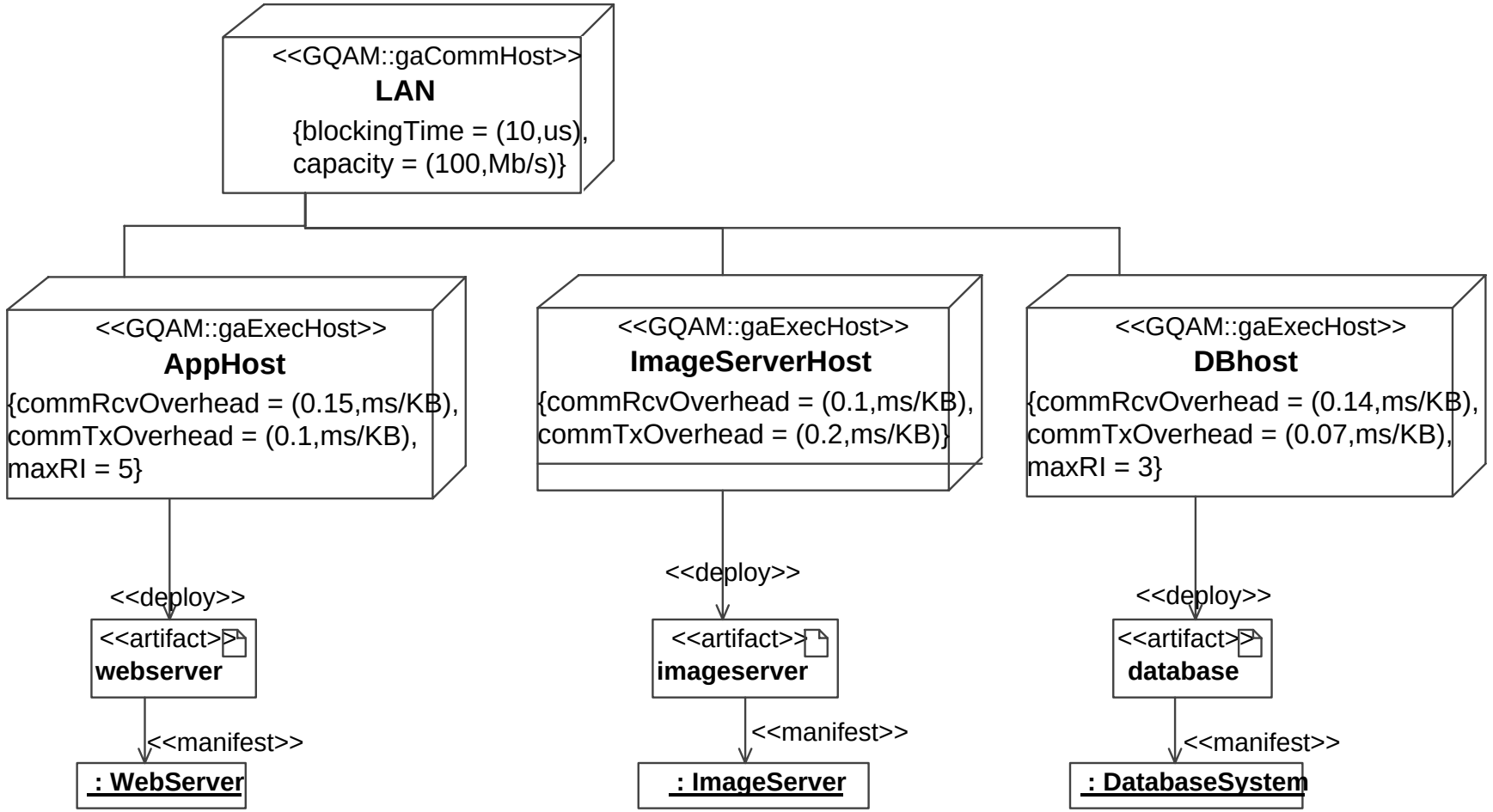



ref
[first promo]
<<PaStep>> {probability = 0.4}
Promotion1
[else]
ref
<<PaStep>> {probability = 0.6}
Promotion2
4: getListOfSubjects
<<PaStep>>
{hostDemand = (10,ms)}
5:
par
[ ]
6: logInteraction
<<PaStep>>
{noSync}
7: getHomeImages
<<PaStep>>
{hostDemand = (0.5,ms),
repetitions = $Images,
extOpDemand = "ImageFileOp",
extOpCount = 1}
8:
9:
<<PaCommStep>>
{msgSize = (3.4 + 5*$Images)}

bottom

Carleton UNIVERSITY

Bookstore Deployment



Carleton UNIVERSITY

Points about the annotation of TPC-W

- context parameters
  - Nusers in the closed workload
  - thinkTime for users,
  - $images per web page,
  - $R = 95th percentile of response time (defined in the Workload)

- Workload defines both required and provided 95th percentiles (required is 1 sec.)
  - multiple stereotypes on one message: Step, CommStep, Workload. Properties become properties of the message

Carleton UNIVERSITY

Points (2)

- parameters on optional Step (rep) and on alt (prob)
- noSync for logging operation means the par block does not wait for it (asynchronous parallel)
- the last message size is a function of the $images parameter

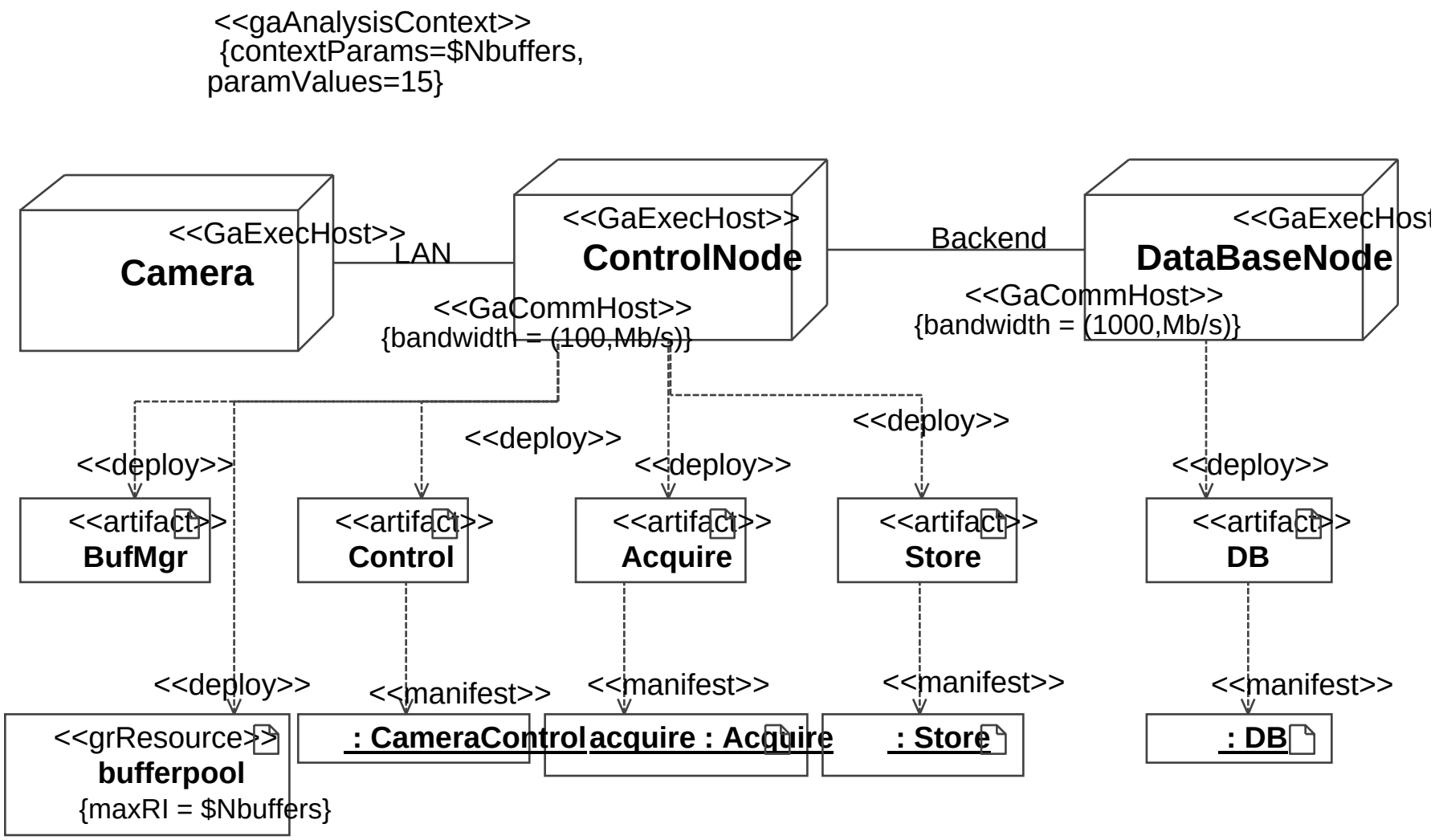
Carleton UNIVERSITY

Example 3: Building Security System (BSS)

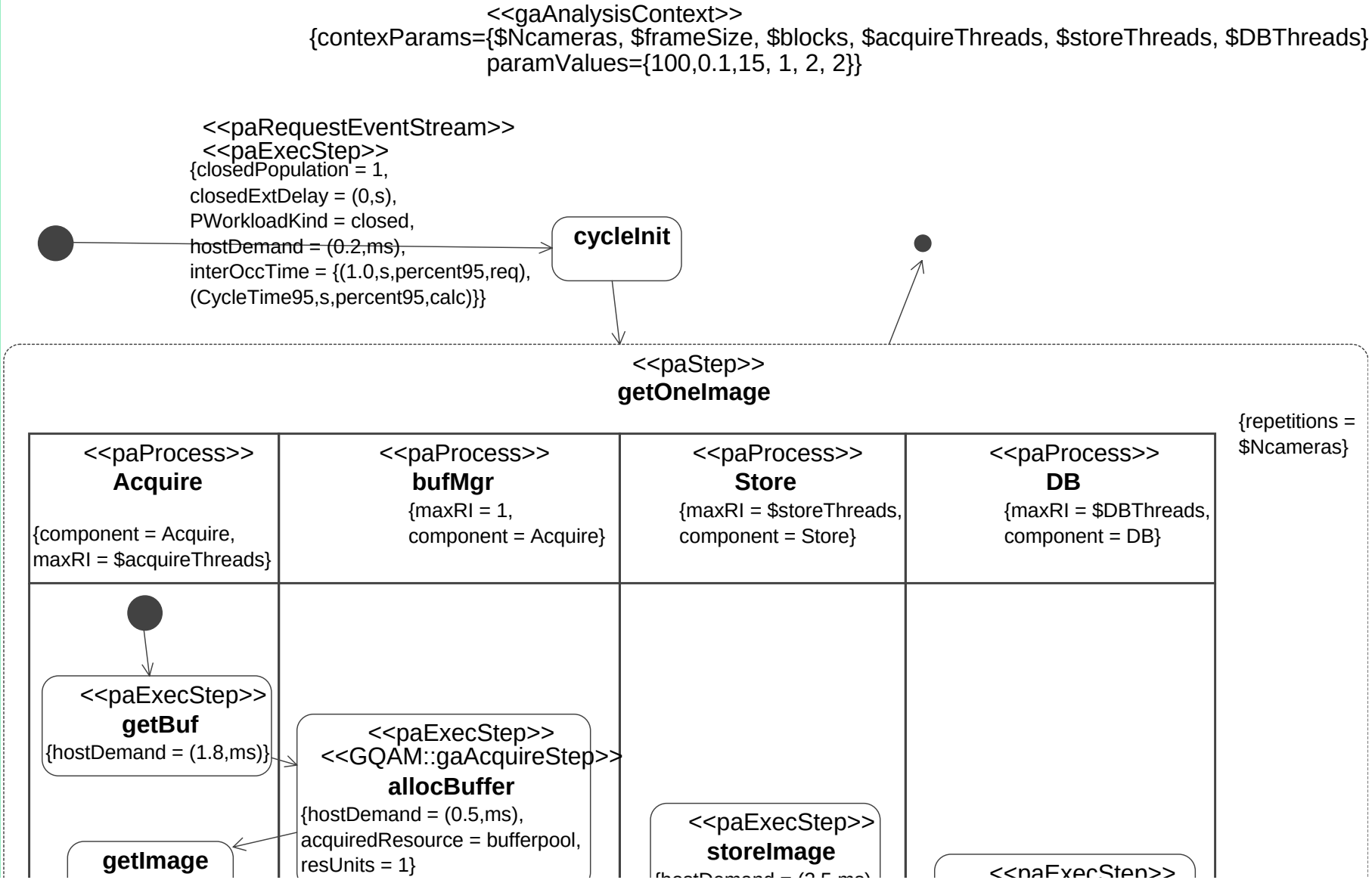
- an embedded system
  - video surveillance of a building
  - the system queries a set of cameras one at a time in a loop, stores a frame in a buffer, then stores the buffer in a database
- performance spec is that 95% of complete scans should finish in less than 1 sec, so each camera gives one stored frame per second.
- logical resources control performance seriously

Carleton UNIVERSITY

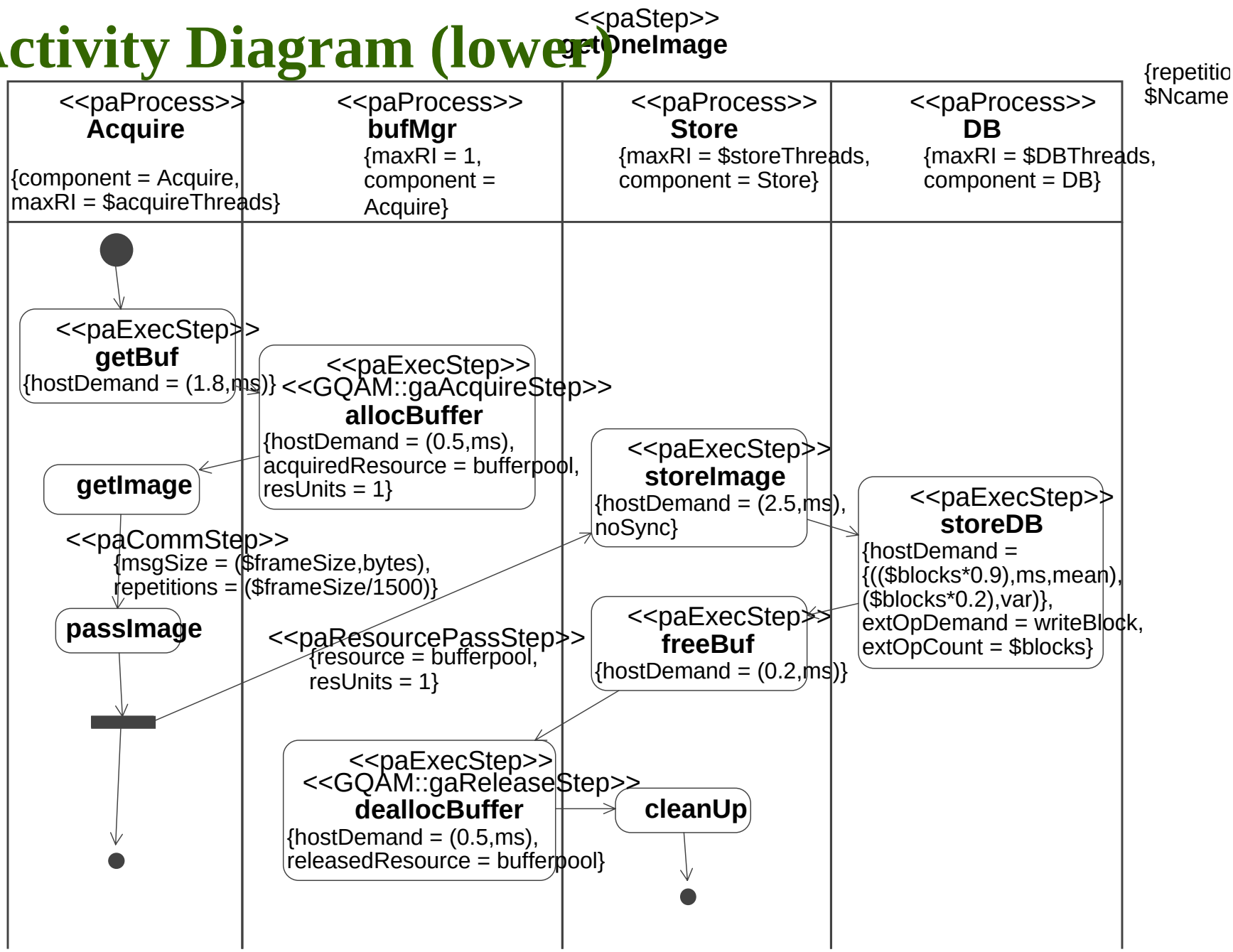
Deployment of BSS



Activity Diagram of BSS

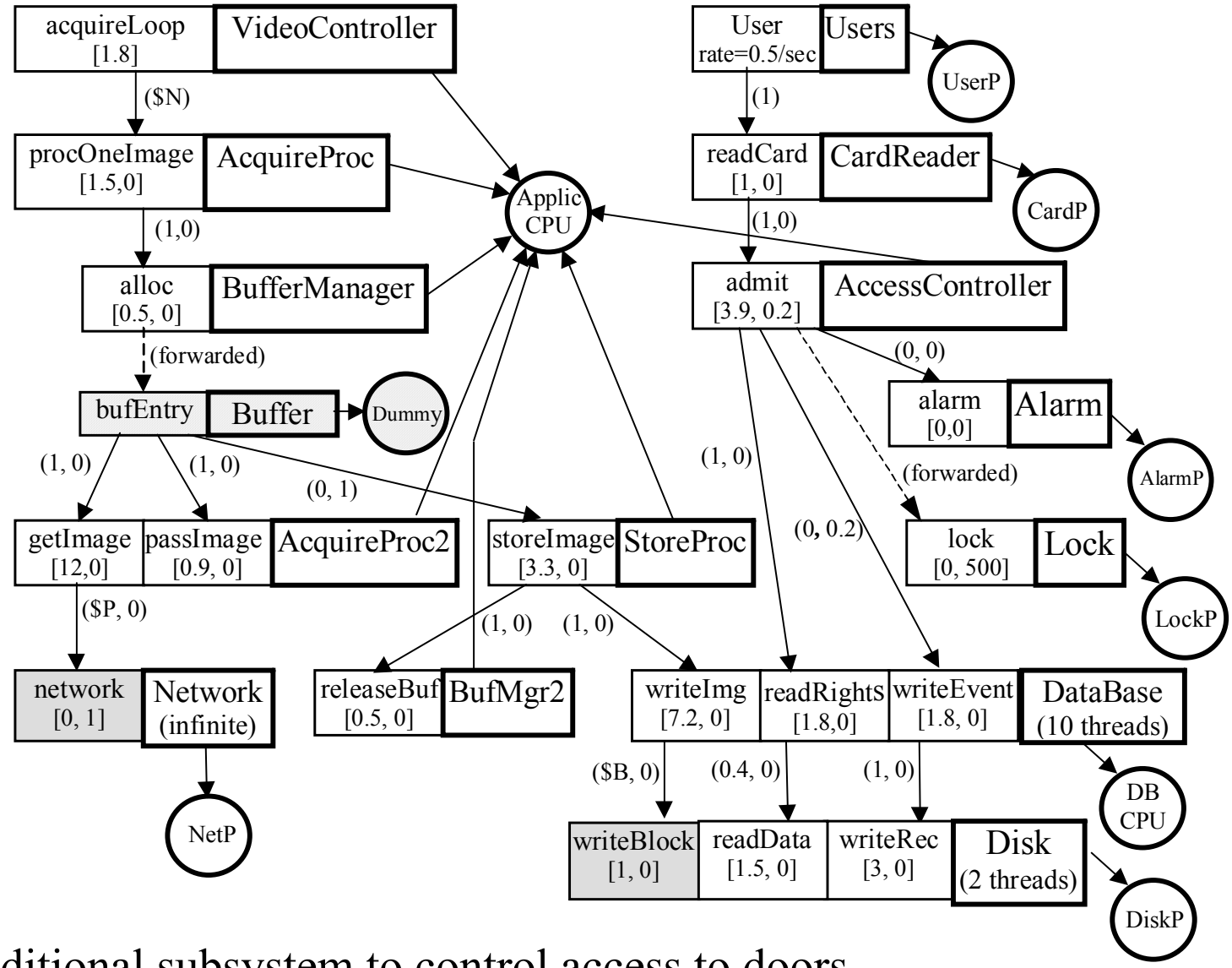


Activity Diagram (lower)



Carleton UNIVERSITY

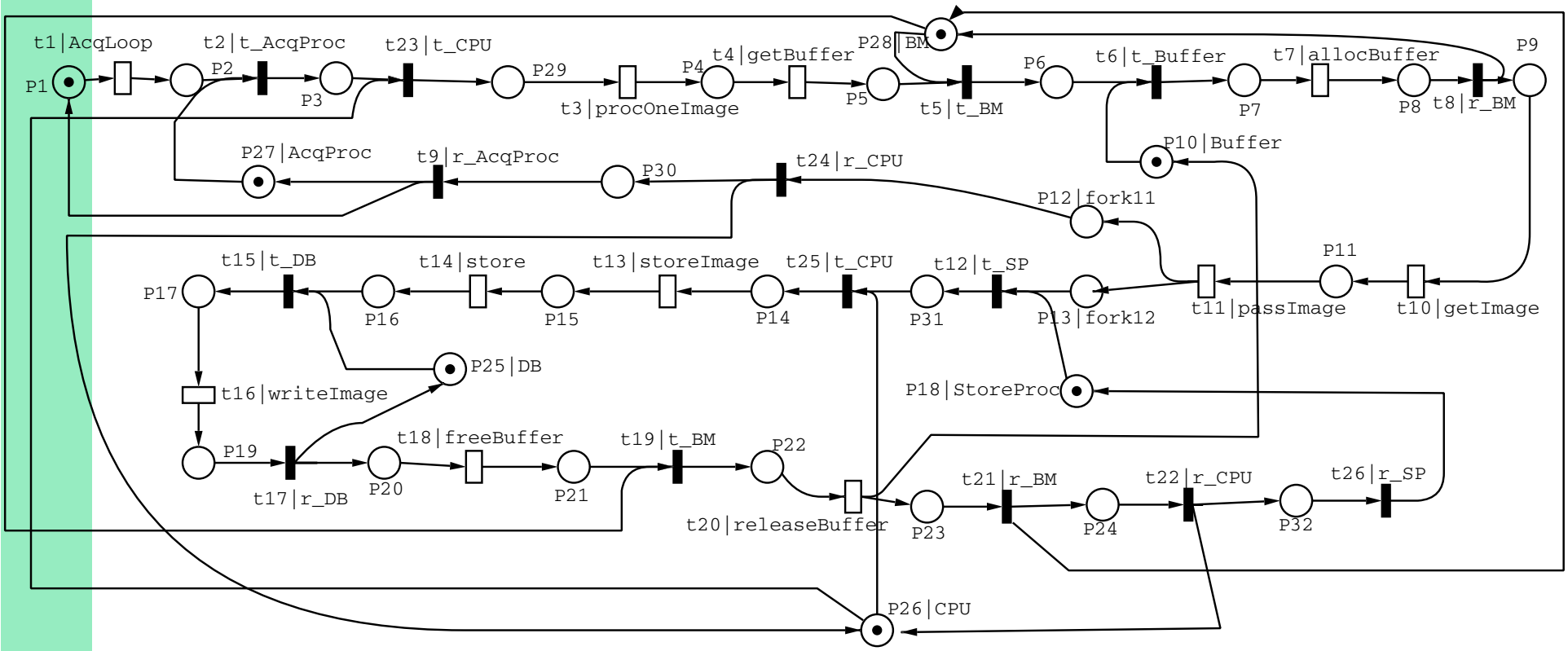
LQN for BSS



- with an additional subsystem to control access to doors

Carleton UNIVERSITY

Petri net model for the video-acquisition scenario



- by Jose Merseguer

Carleton UNIVERSITY

Performance Improvement in BSS

- **Base case system capacity: 20 cameras (for 95% of cycles < 1 sec)**

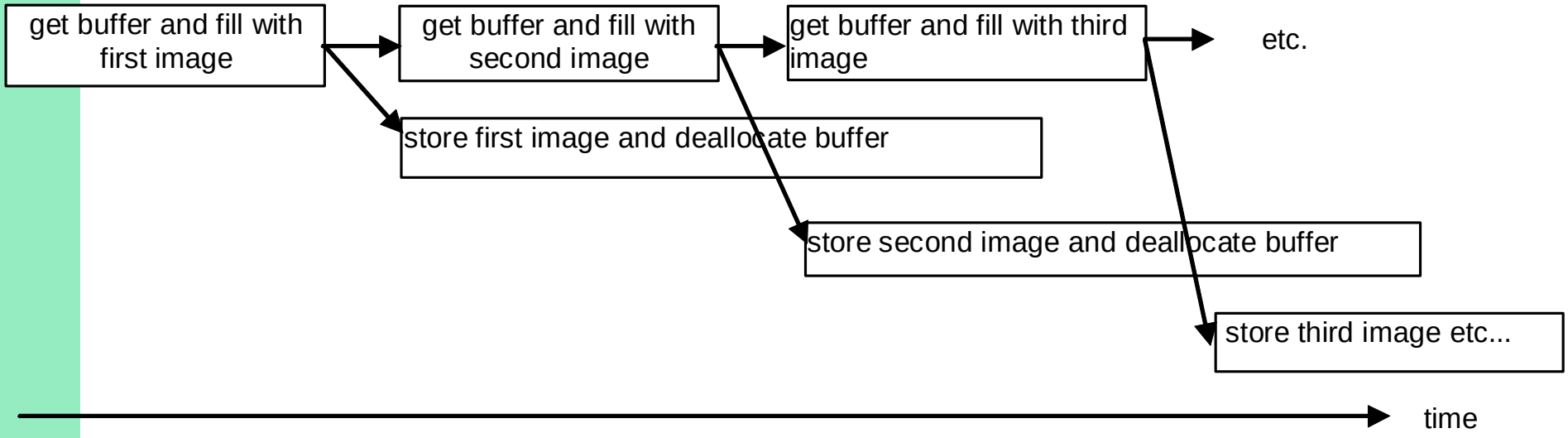
- **Cloning Software Bottleneck:**
  - 4 Buffers and 2 StoreProc threads
  - gives buffer overlap, storage in parallel with next read
  - 40 cameras, performance improvement 100%

- **Cloning Hardware Bottleneck:**
  - Dual Application CPU
  - 50 cameras, performance improvement 150%

- **Introduce more currency**
  - Moving first phase call to second phase
  - allows next retrieval to begin at once
  - 100 cameras, performance improvement 400%

Carleton UNIVERSITY

Buffer overlap



Carleton UNIVERSITY

Communications between objects

We can distinguish five cases for transferring a message of size $msgSize between two objects by a CommStep:

1. between objects in the same process: communications is by pointers and the workload is absorbed into the object hostDemands
2. between processes on the same host: there may be IPC overhead but it seems best to absorb it into the process workloads.
   - however, we could in principle compute the added hostDemand as a function of $msgSize

Carleton UNIVERSITY

Communications (2)

3. between processes on different nodes, using OS protocols, the node has messaging overhead properties.
    3. commRcvOvh and commTxOvh: CPU demands for messages
    4. they are constants for the node
    5. in principle they might be better to be linear functions
        3. Ovh = constant + slope * $msgSize
4. also, the link (which is a CommHost) has properties
    - latency, called "blockT" for consistency with other host resources
    - a bandwidth limit, which establishes a service time for the message: service time = $msgSize/bandwidth.
    - a contention model for the link uses this service time

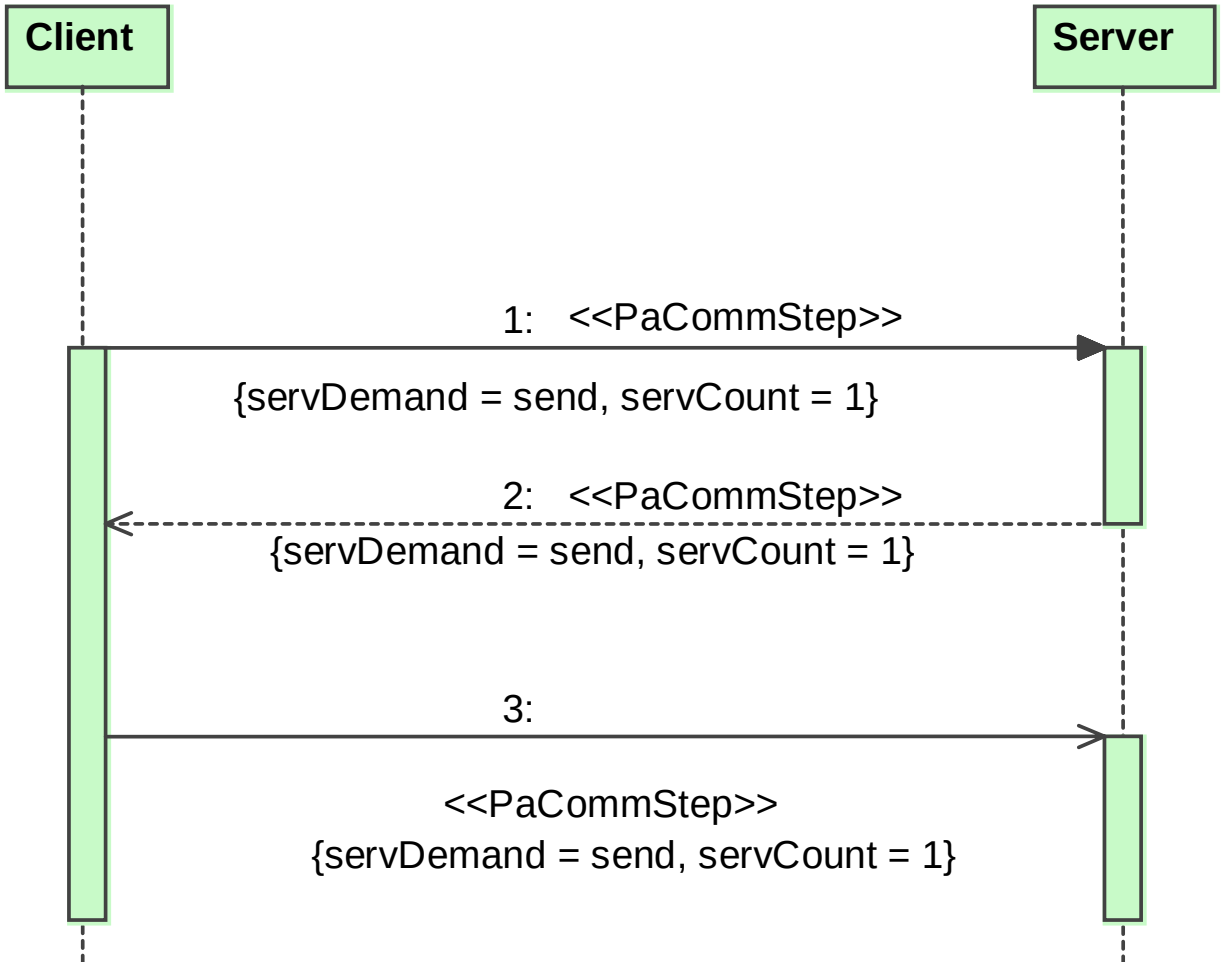
Carleton UNIVERSITY

Communications (3)

5. between objects that are linked by a communications infrastructure such as a CORBA or RMI layer
   - the CommStep identifies the layer operation as a RequestedService through a servDemand property
     - servDemand = operation name
     - servCount = number of requests to the operation
   - The operation is defined at the interface of the layer model, in a class diagram
     - class
     - structured class (composed of components) with an interface
   - OR the CommStep identifies the operation as an external service, modeled in the performance domain.
     - for example a separate model of TCP protocol

Carleton UNIVERSITY

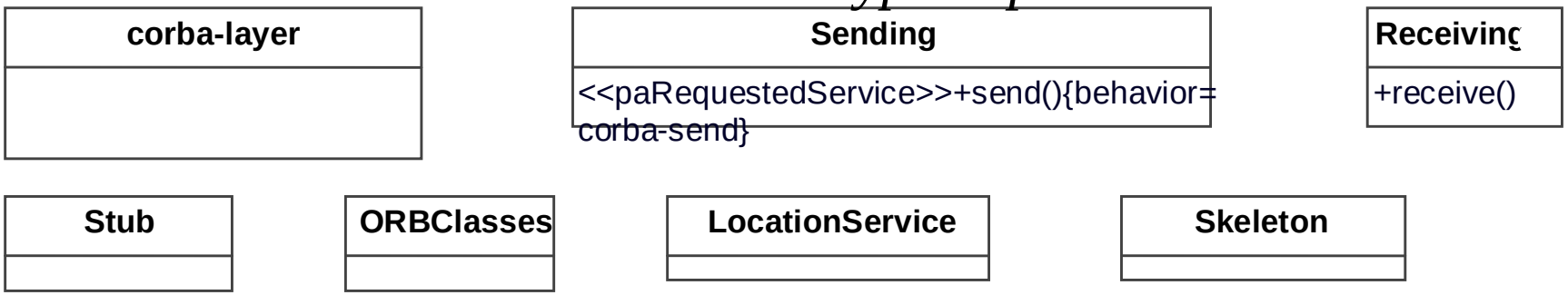
Example 4 : Communications layer with a send operation



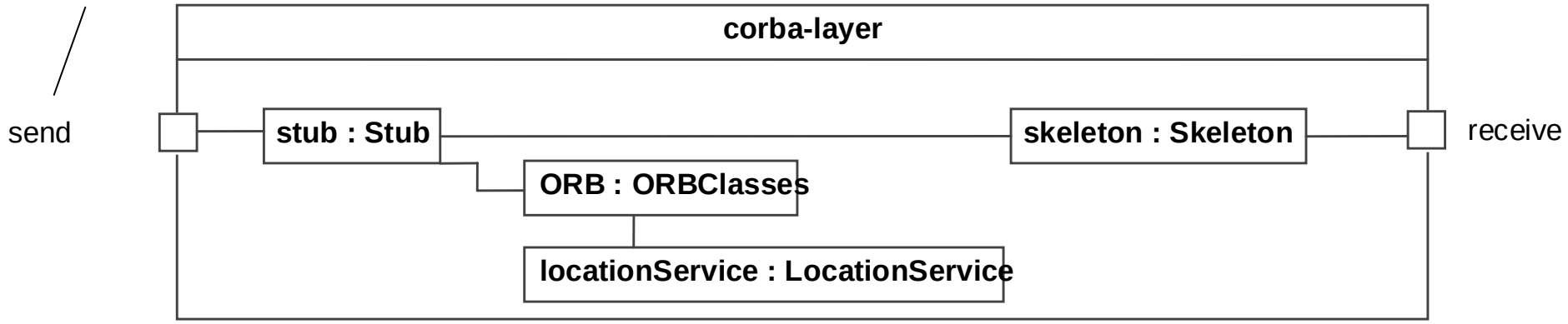
Carleton UNIVERSITY

CORBA component structure to provide a communications layer

interface class with stereotyped operation

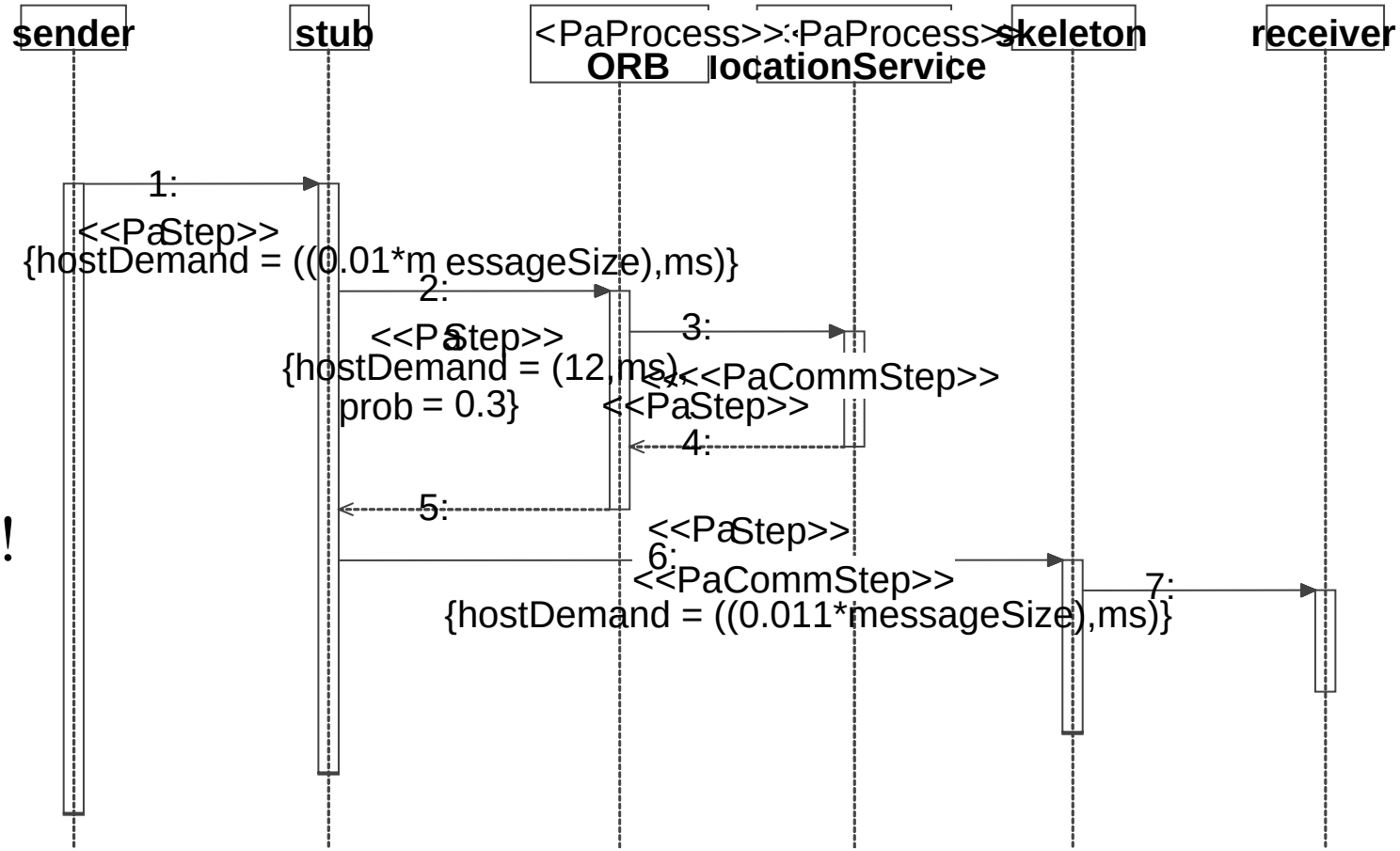


interface of class Sending



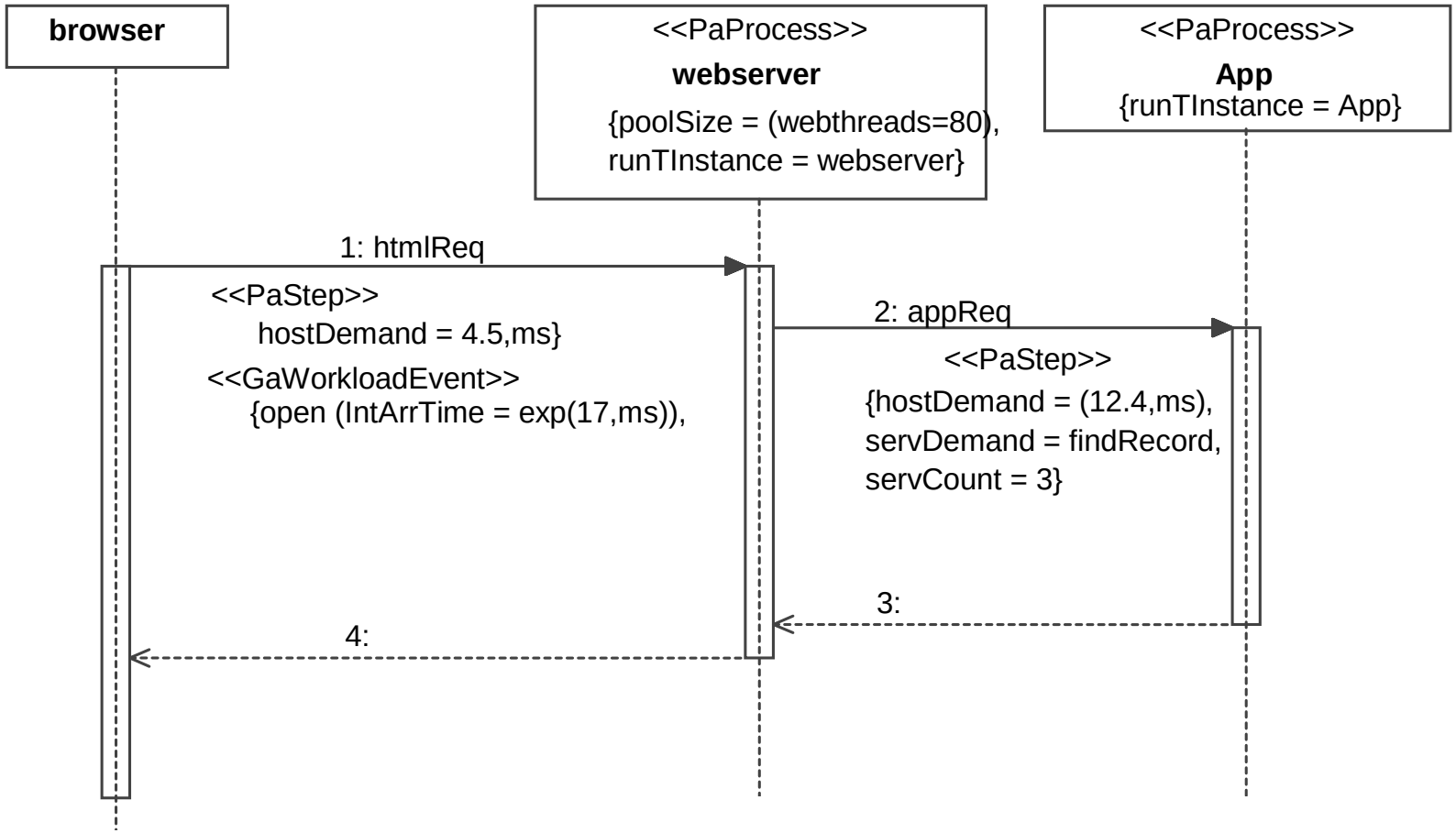
CORBA service is defined by a subscenario for the service (a property of the RequestedService stereotype)

- requires binding the sender and receiver roles to the context of the demand!

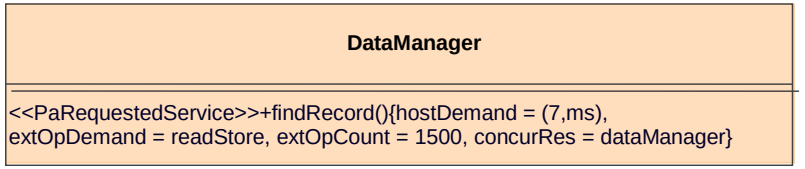


Carleton UNIVERSITY

Example 5: Component calling hierarchy

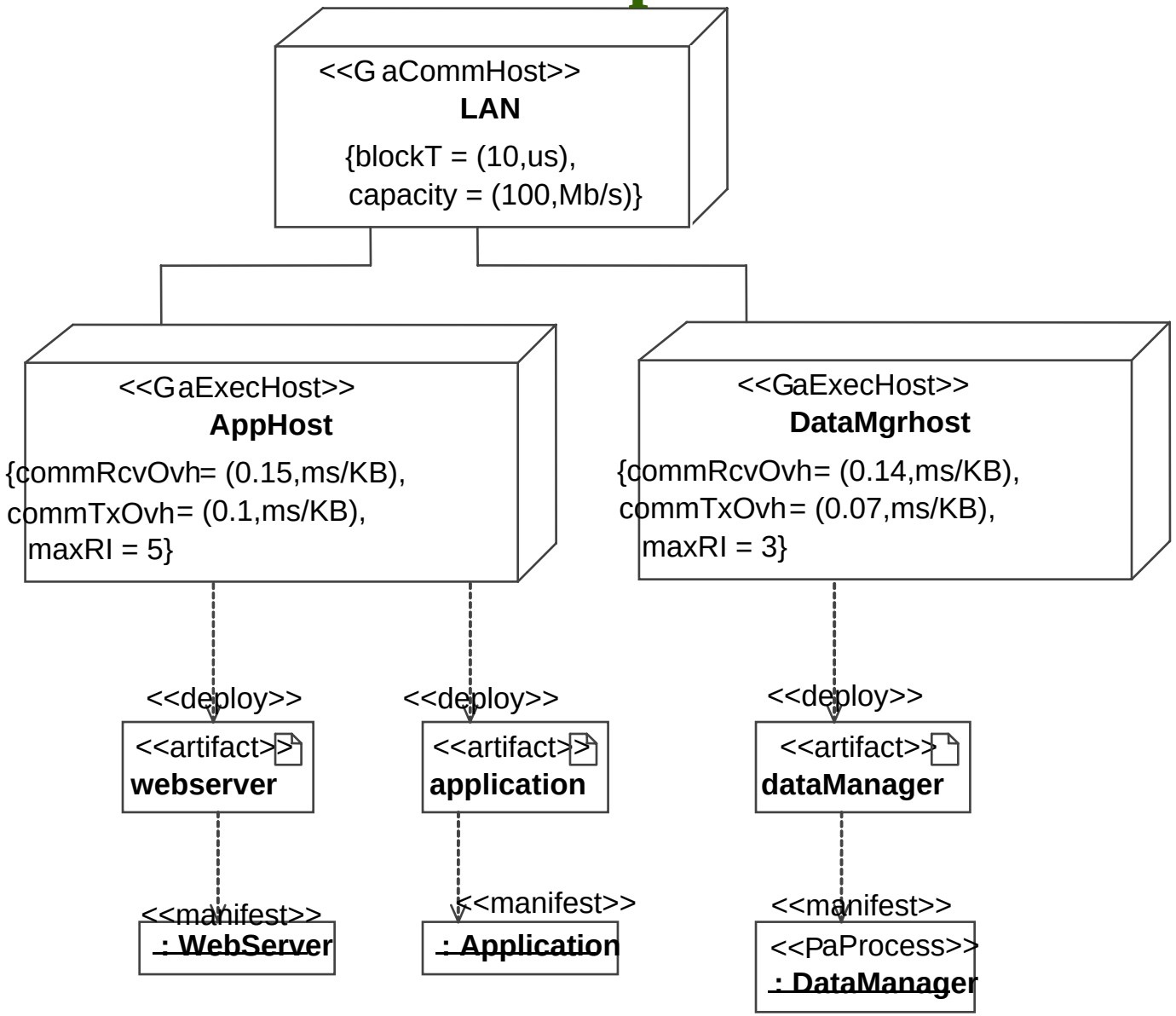


findRecord is provided by a class DataManager:



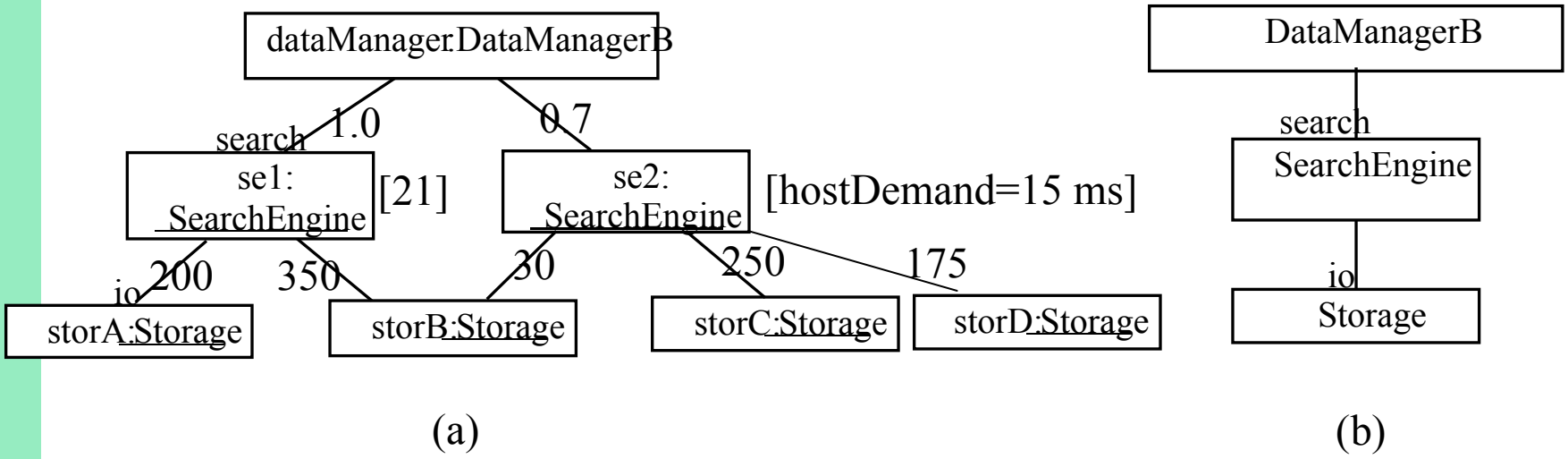
Carleton UNIVERSITY

Deployment for the called component



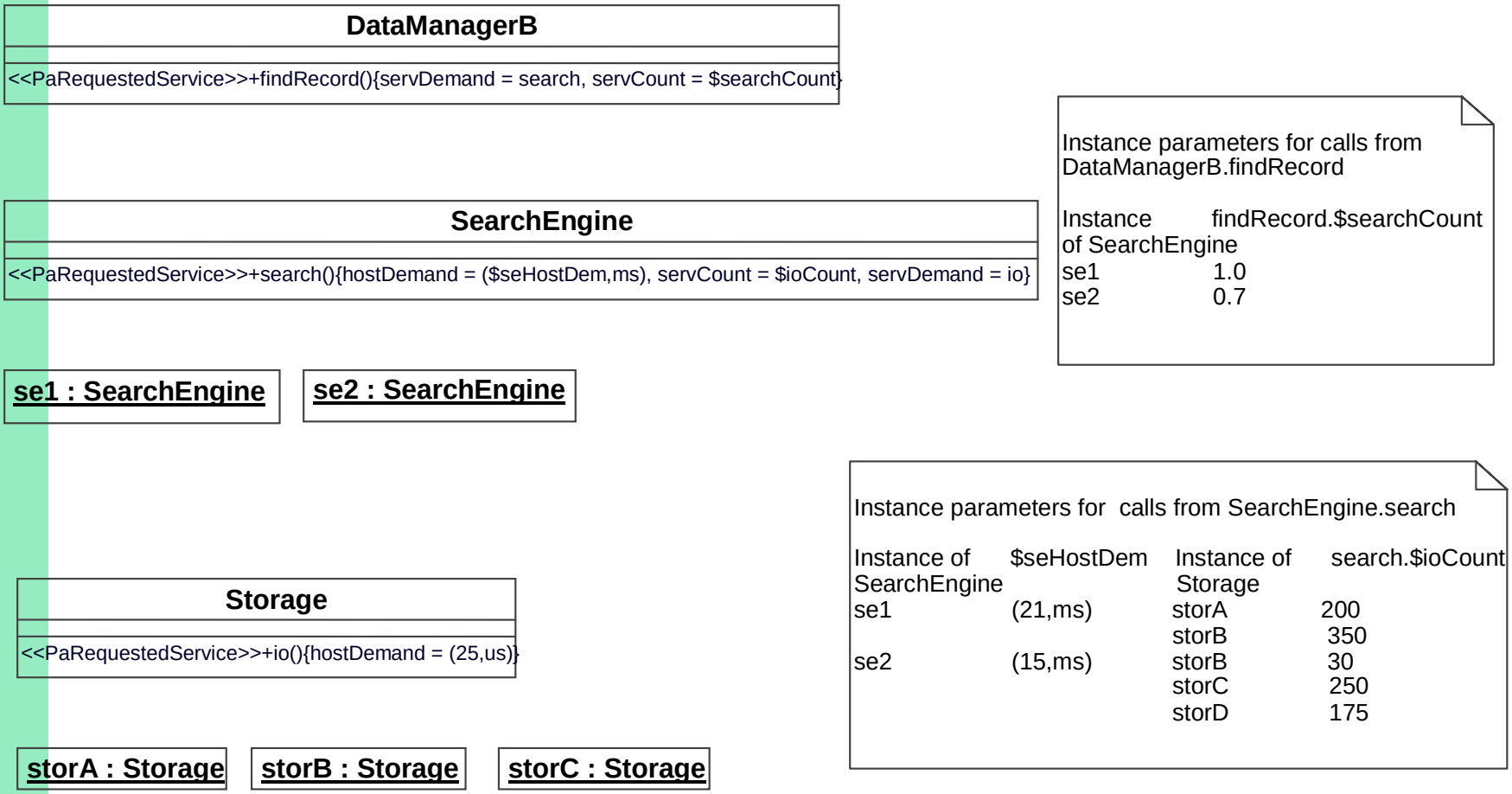
Carleton UNIVERSITY

Call hierarchy with heterogeneous instances



- Now suppose dataManager makes calls into a subsystem
  - Classes and relationships at right
  - instance-based call hierarchy at left
  - suppose the instances have different demand parameters
    - calls to sei
    - sei hostDemands
    - calls to storages

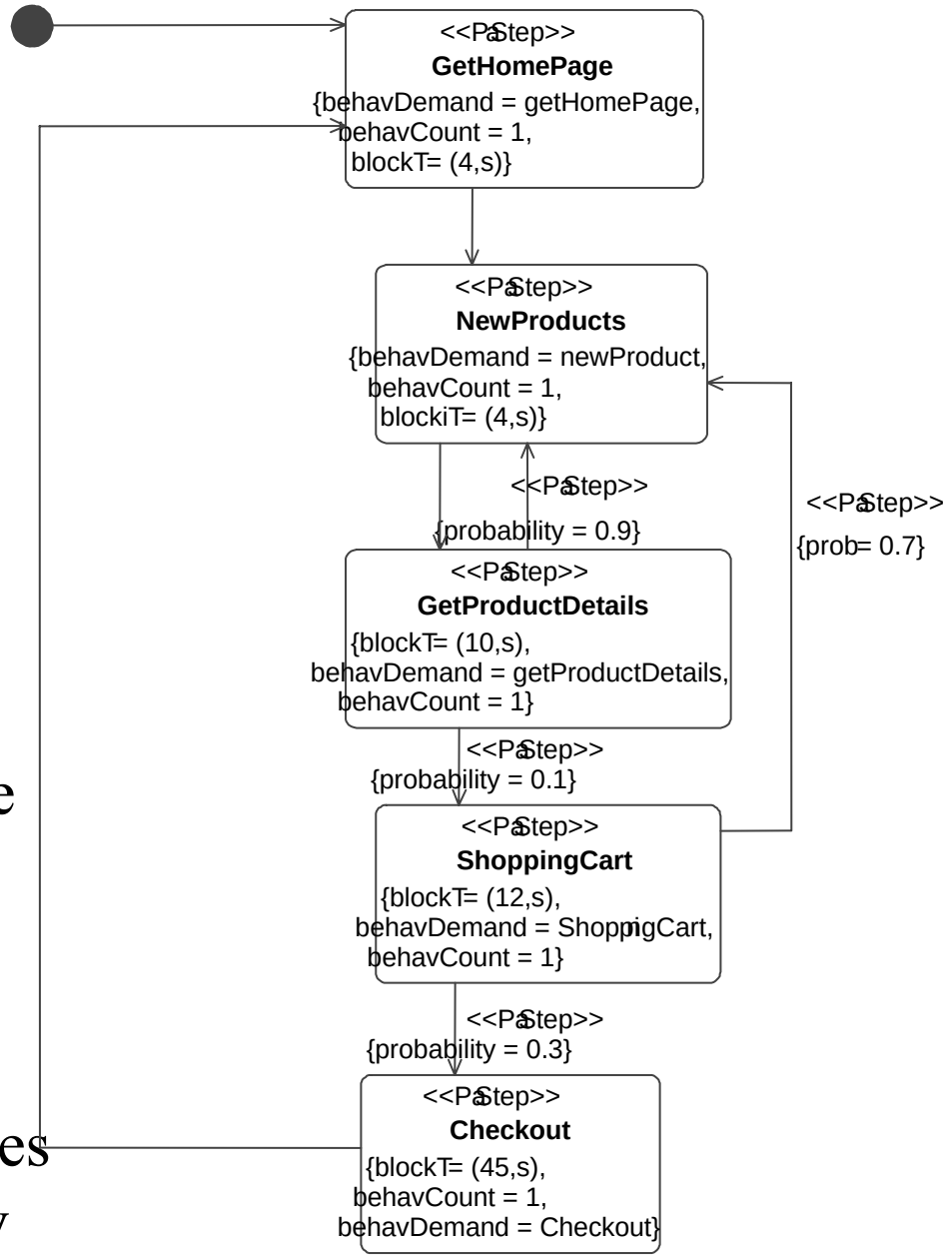
Call hierarchy with heterogeneous instances (2)



- use variables in the class stereotypes and tables that identify the instances by name, to give values
- if no variation (as in Storage), use a constant parameter.

Example 6: a Structured workload defined by a State Machine

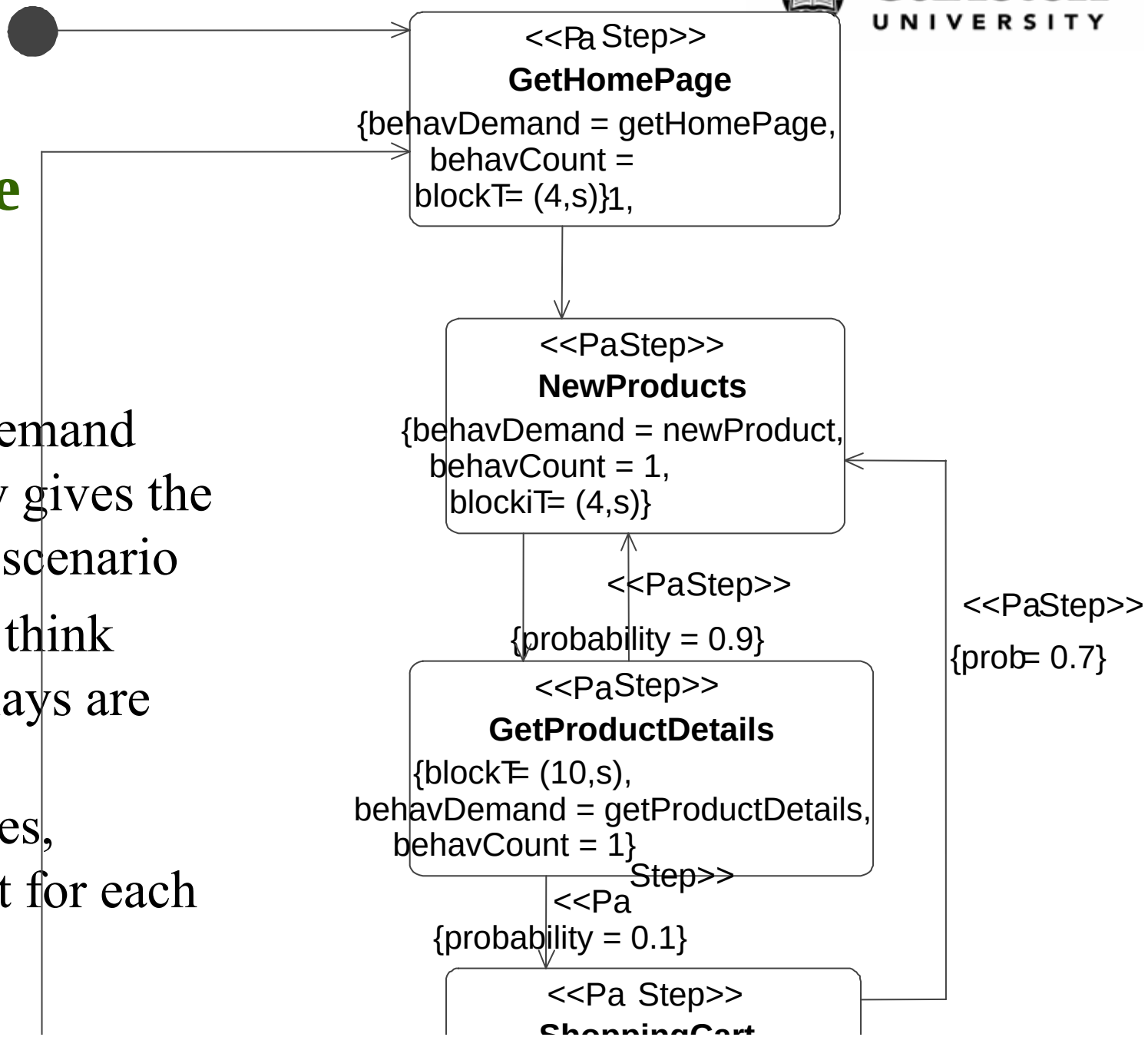
- the SM is a WorkloadGenerator property of the overall Workload
- each state is a PaStep
  - with an explicit subscenario for one page of TPC-W
- Some transitions are also PaSteps
  - with probability that gives the transition probability



Carleton UNIVERSITY

State Machine Detail

- behavDemand property gives the explicit scenario
- internal think time delays are blockT properties, different for each step



Carleton UNIVERSITY

Example 7: State machines in general

A scenario can be defined by a SM or set of SMs

- top level SM is driven by a workload
- SMs that are invoked by signals sent, are sub-scenarios
  - if they cause a return to the invoking SM
  - otherwise a global SM construction is necessary, with association of timing parameters to global transitions... not described in MARTE.
- concurrent regions of a composite state are forked on entry
  - join must be due to termination of all regions
  - exit from the entire composite state on a given event like an exception, is not accommodated by MARTE
    - possibly multiple endings of a scenario
- parameters are given for classes: for instances with different parameter values, an approach using variables and tables is needed, as for component instances.

Carleton UNIVERSITY

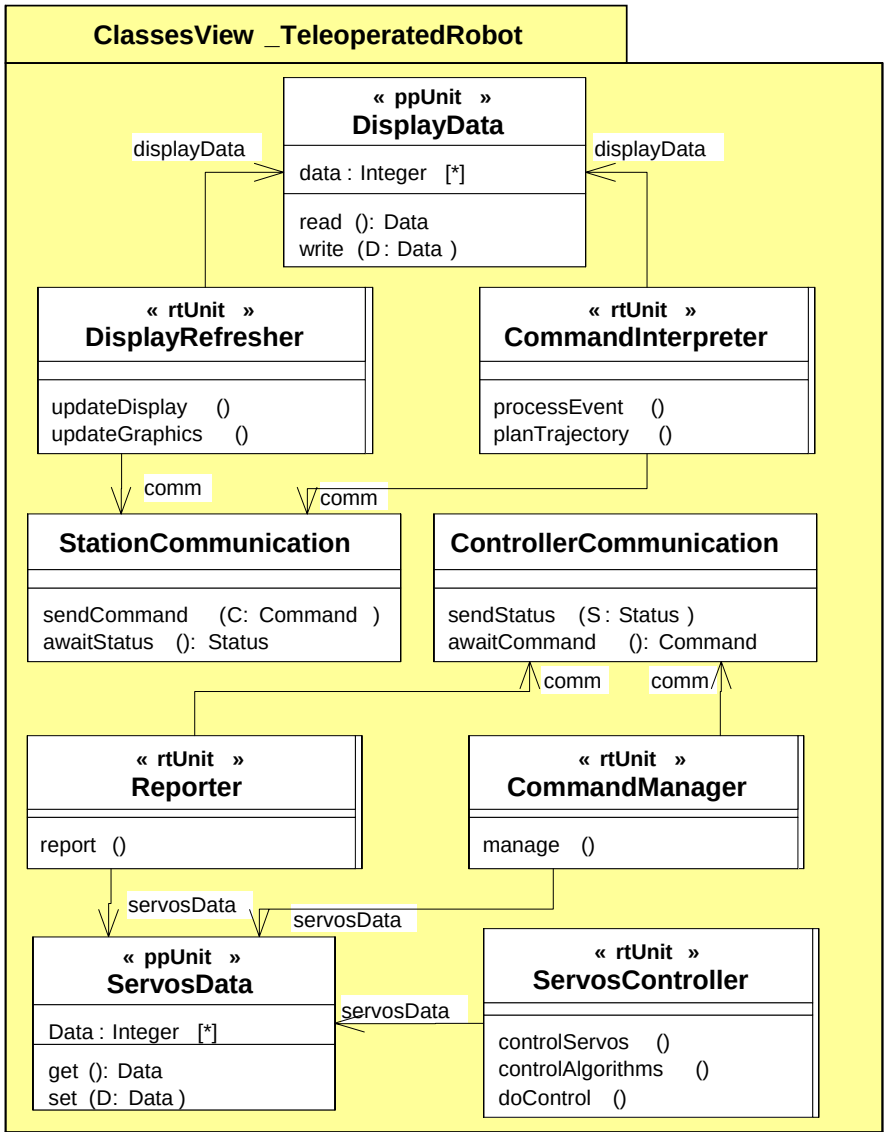
Schedulability

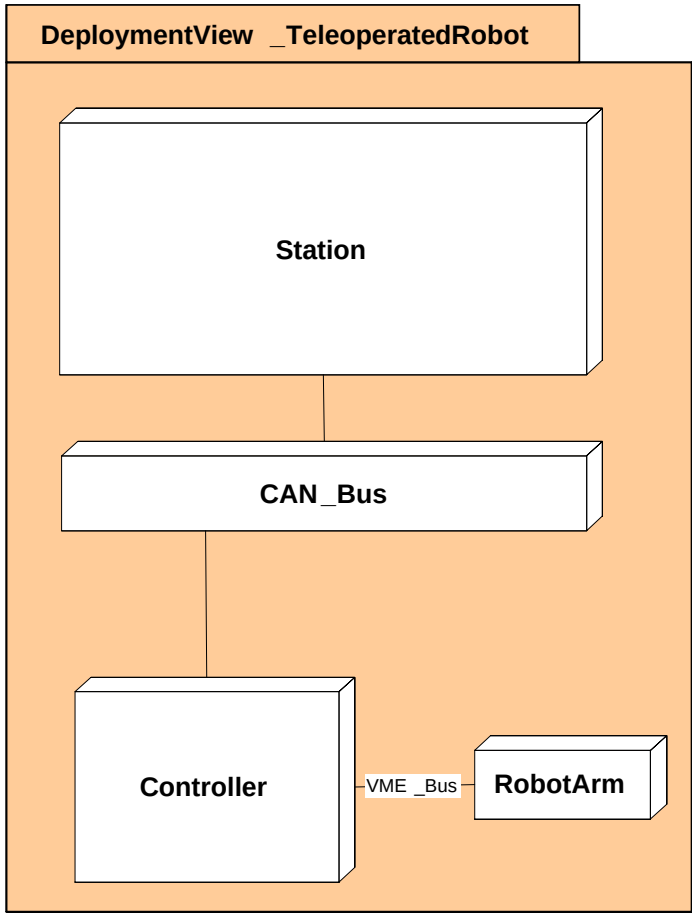
- this subprofile is somewhat less abstract
  - detailed types of shared logical resources, e.g. semaphores.
  - deterministic ordering of actions
- naming from the schedulability sub-culture
  - stereotype names, e.g. “shared resource” for logical resources.
  - many specific terms from the field have stereotype attributes,
    - e.g. WCET = worst case execution time
- emphasis on deterministic values and behaviour

Carleton UNIVERSITY

Schedulability Example

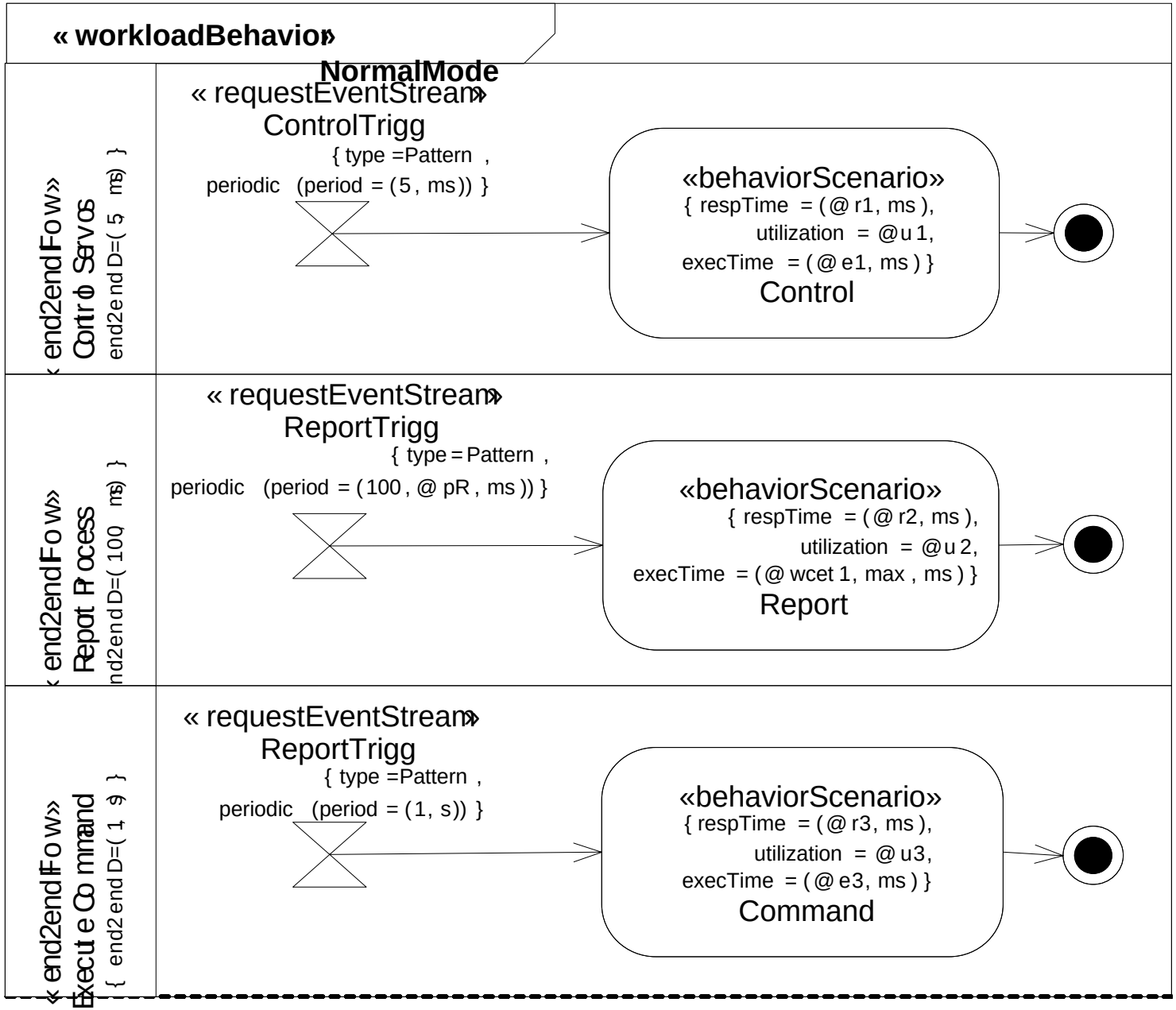
- remote control of a manufacturing cell





Behaviour

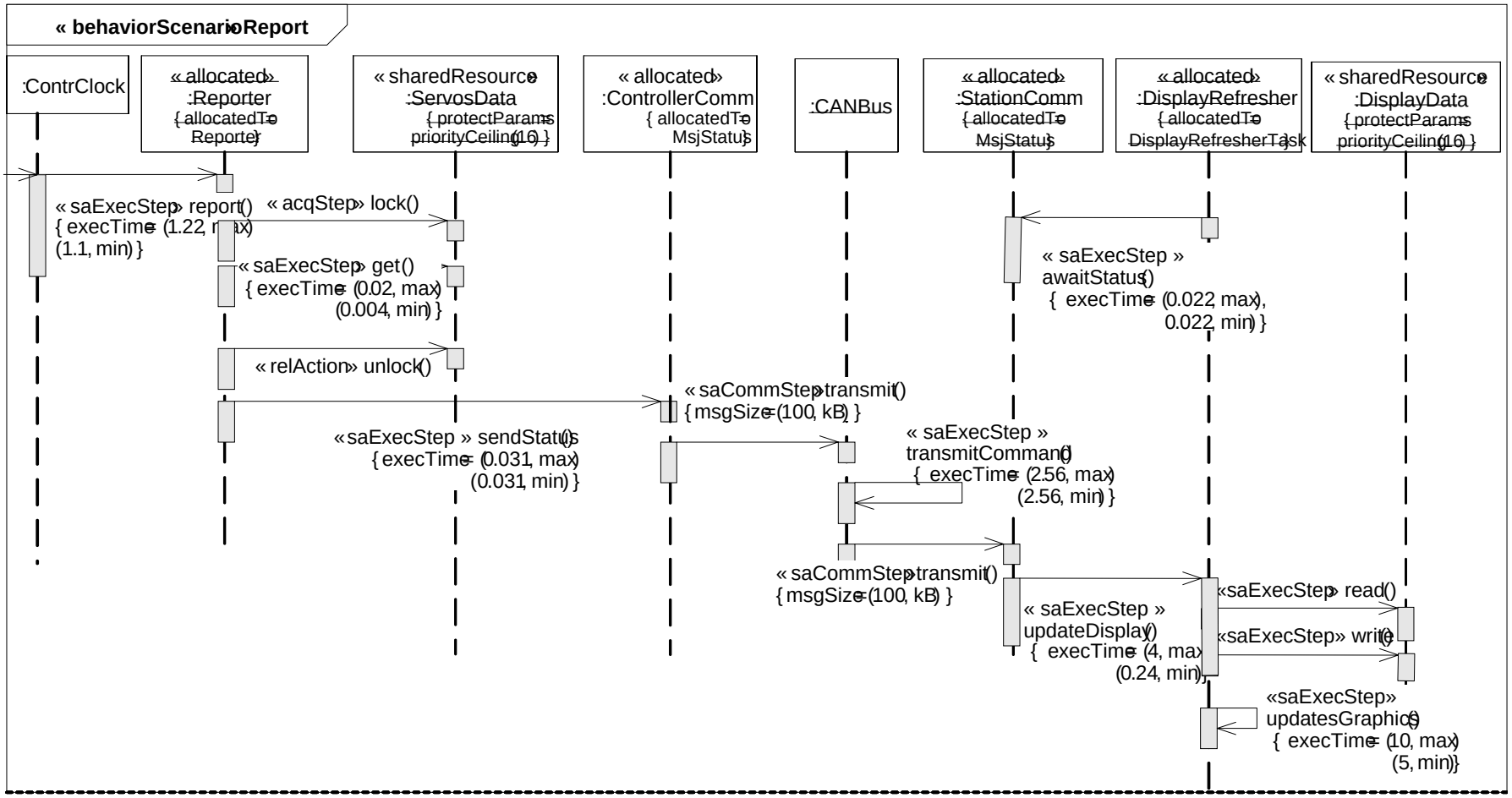
- three scenarios each with its own trigger



Carleton UNIVERSITY

Scenario

- one of the three behaviours of previous slide



Carleton UNIVERSITY

Conclusion

- MARTE is apparently about to be adopted (June 2007)
- tool support may be slow because of the size of it
  - subprofiles are defined, e.g. for performance, schedulability, and for real-time design.
- significant extensions for performance analysis
  - communications modeling
  - components and call hierarchies
  - point-to-point delays
  - VSL (which may also be adopted by SysML)
- a lot of interest at OMG in real time properties now.