



Formal Method Support for UML- Based Systems Design

Colin J Neill¹ & Jon D Holt²

cjn6@psu.edu

¹Penn State University, Malvern, PA 19355

²Brass Bullet Ltd, Swansea, UK



The UML

- ◆ A family of languages intended to describe software intensive system in a form readable by humans and machines.
- ◆ Applicable to systems design because OO supports representation of real-world artifacts as objects.
- ◆ Falls short when considering the integration of disparate systems, particularly when time is an issue:
 - UML has only naïve support for the complex, time-dependent interactions that characterize system boundaries.

UML & Time

- ◆ The UML has been suggested as a suitable language for specification and design of real-time systems.
- ◆ BUT, the only representation of time is as a label.
 - Temporal properties are not described in detail or defined appropriately for analysis.
 - State-based representations apply to only a subset of real-time systems.
- ◆ Alternatives exist (temporal logics, Petri nets, etc) but are considered esoteric and non-scalable.
- ◆ A potential solution: transform UML models into an appropriate formalism as mechanically as possible.

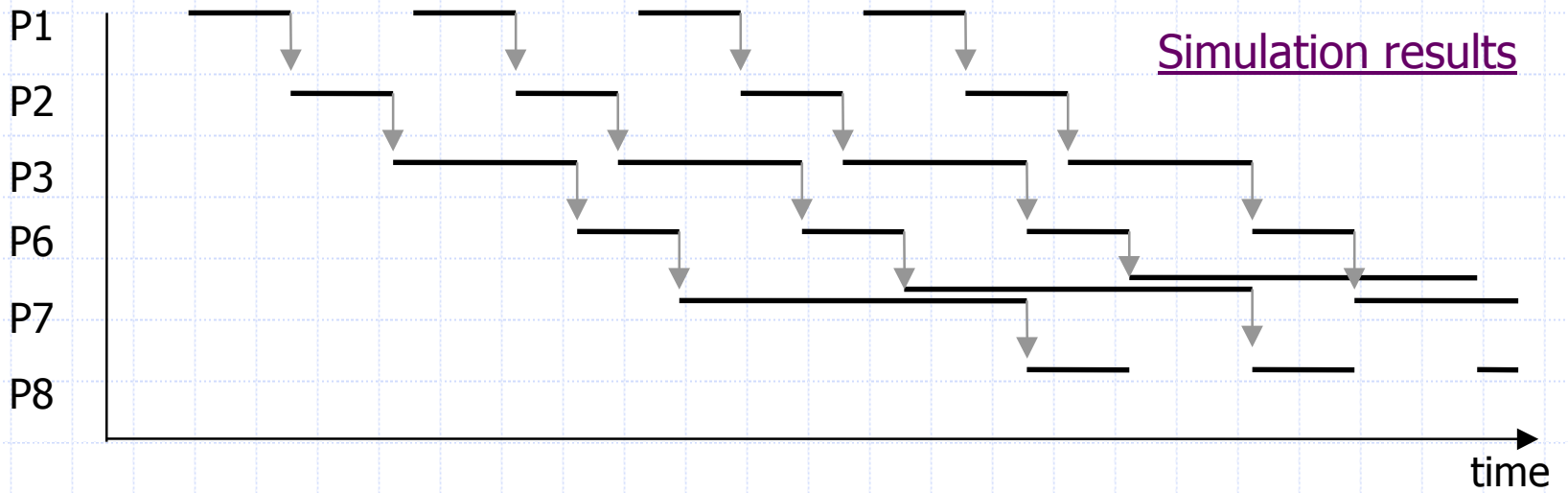
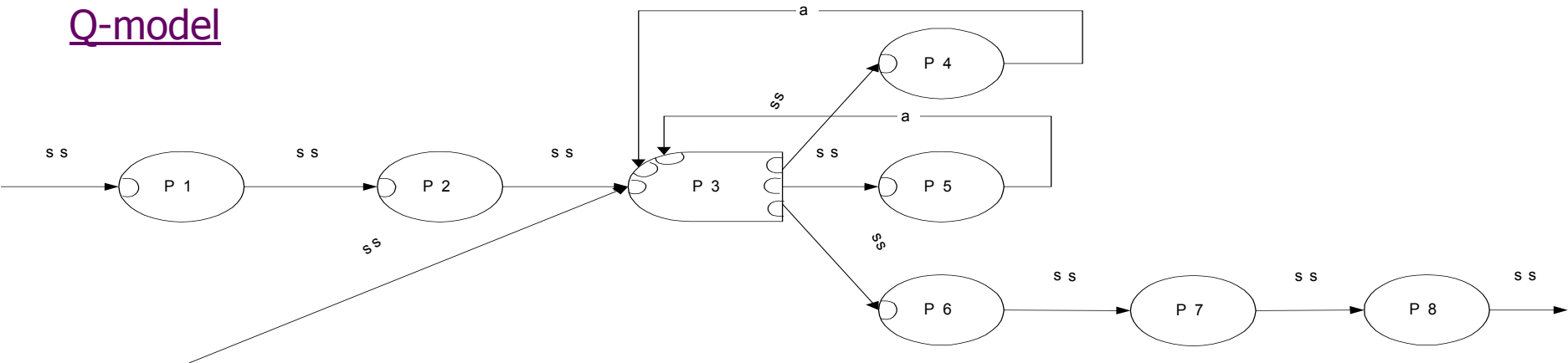
The Q-Model

- ◆ First developed by Quirk & Gilbert in the 1970's
- ◆ Mathematically-based computational model for describing distributed, time-constrained, concurrent programs.
- ◆ Relatively simple model construction; strong verifiable mathematical foundation
 - Can simulate and formally analyze temporal aspects of specification.
- ◆ Considers a system as a set of abstract processes loosely coupled by a set of channels.
 - Temporal parameters can be defined for each process, where a process can represent behavior at any abstraction.
 - Channels define inter-process communication and synchronization.

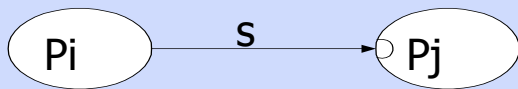
Processes & channels

- ◆ Two process types:
 - Standard process
 - Selector process – represents selection of input, output or both.
- ◆ Five channel types:
 - Synchronous – data producer and consumer processes activated concurrently
 - Synchronous null – a non-data carrying synchronous channel
 - Sequential – data consumer is activated after producer process completes
 - Sequential null – a non-data carrying sequential channel
 - Asynchronous – data is passed, but consumer process is not triggered.
- ◆ Temporal characteristics of each process is defined by its timeset
- ◆ Time-selective communication supported by defining channel functions

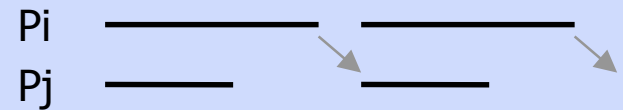
Q-model



Note:



results in →

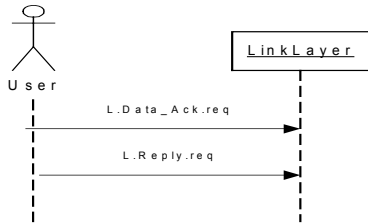


An Integrated Approach

Use Case

USE CASE:
AckConnectionless
 Typical Course Of Events
 1. This use case begins ...

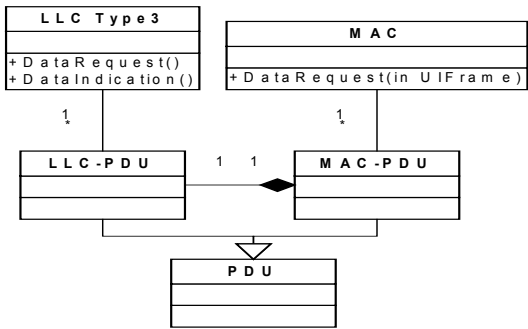
System sequence diagram



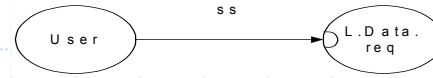
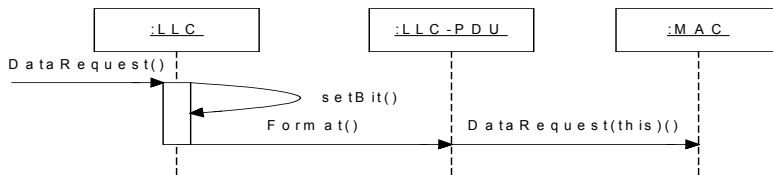
Contracts

Operation:
 L.Data_Ack.req
Preconditions:
Postconditions:
Time Constraints:

Class diagrams

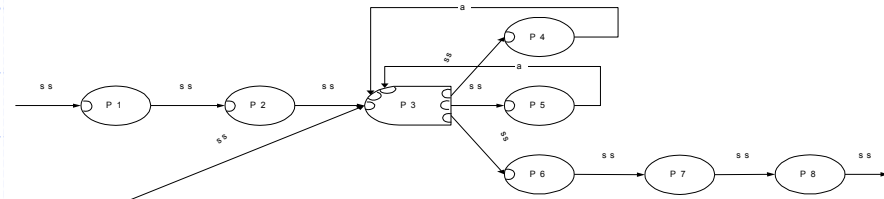


Sequence diagrams



Process Timeset:
L.DataRequest
 Start instants = $[t_0 + 0.02]$
 Execution time = $[2, 3]$
 Consumption time = $[0.5, 0.5]$
 Equivalence interval = n/a
 ...

Channel functions:
 P1!P2 = $[0, 0]$
 P2!P3 = $[1, 0]$
 P3!P4 = $[1, 0]$
 ...



Tools

- ◆ Clearly the process described requires tool support.
 - Q-model generation
 - ◆ Encapsulates transformation rules
 - ◆ Requests additional data in meaningful terms
 - Simulation
 - ◆ “Execute” Q-models and feedback results to users (again, in meaningful terms)
 - Analysis
 - ◆ Formally test Q-models for liveness, deadlock, livelock, etc..
- ◆ LIMITS tool¹ supports B and C, but requires manual generation of Q-models.

1: L.Motus, T. Naks, “Formal timing analysis of OMTdesigns using LIMITS.” Proceeding of the 3rd International Workshop on Object-Oriented Real-time Dependable Systems, 5-7 Feb, 1997. IEEE Computer Press, USA, pp.137 – 144.

Future work

- ◆ Formal definition of transformation rules.
 - Currently transformation requires thorough understanding of both modeling domains
- ◆ Investigation of appropriate feedback mechanisms of analysis results
 - Again, to understand the analysis results, and any necessary changes, the user must understand the Q-model formalism.
- ◆ Develop tools to implement transformation!