

Compositional Quality of Service Semantics

Presented at SAVCBS 2004

November 4, 2004

The following slides were generated from Squeak's save as postscript option, which is very buggy and fails to reproduce fonts accurately. My apologies for this poor rendering and for the loss of the animations.

Richard Staehli

Compositional Quality of Service Semantics

Richard Staehli and Frank Eliassen

Simula Research Laboratory
P.O. Box 134
N-1325 Lysaker, Norway
email {richard,frank}@simula.no



Overview

Reasoning about component QoS.

A QoS meta model.

What's the value of this approach?

Related work.

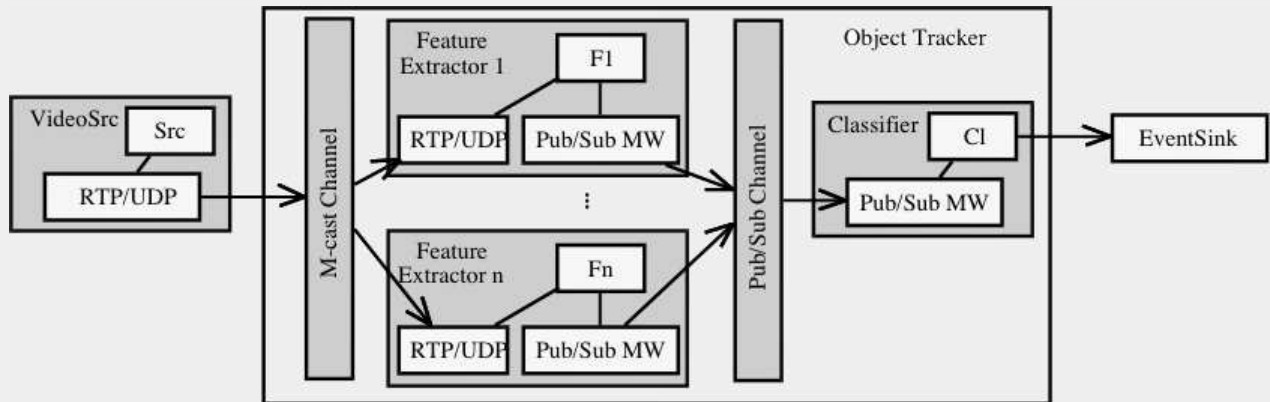
Conclusions.

The QuA Project

An architecture that enables component-based software engineering of real-time and multimedia applications (both a priori guarantee and dynamic adaptation approaches).

Need a theory of what QoS is, how to specify requirements, how to reason about for component selection.

The QuA QoS Meta Model provides a theory of how to specify QoS for components and how to reason about relation between component and composition QoS.

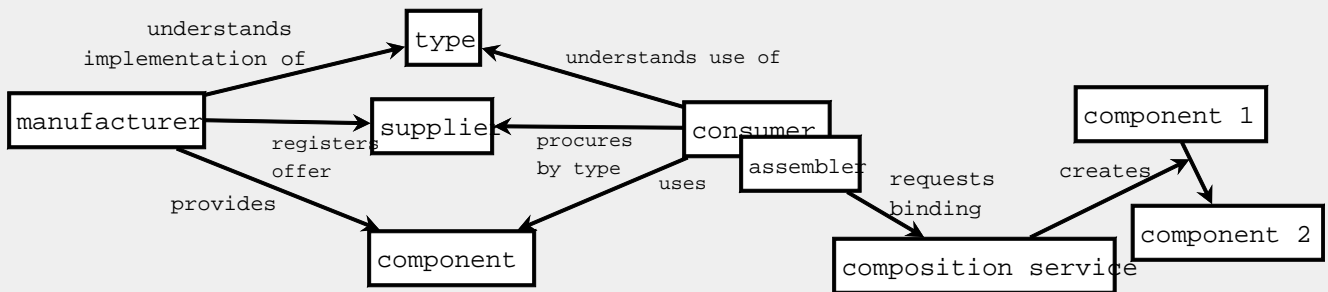


Reasoning About Component QoS

What is QoS?

delay, resolution, reliability...
scheduled event time, cost, bandwidth...
extra functional properties

What's different about components?



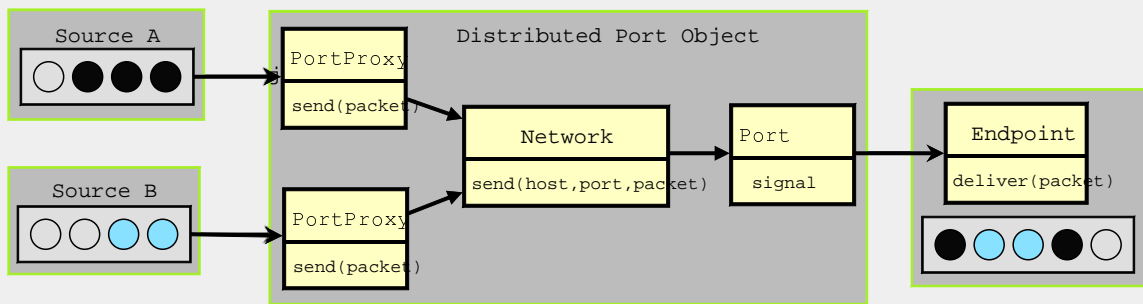
QoS?

QoS = Q1 + Q2?

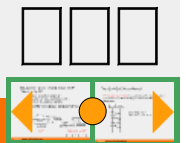


The QuA QoS Meta Model

Definition 1: A **service** is a subset of output messages and causally related inputs to some composition of object.

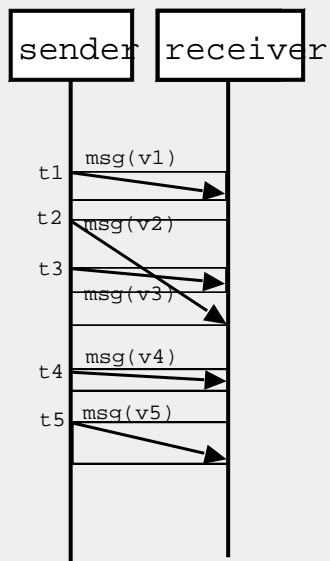


A Packet Service Example



The QuA QoS Meta Model

A **message event trace** is a sequence of message values associated with the sending interface and the time it was sent.



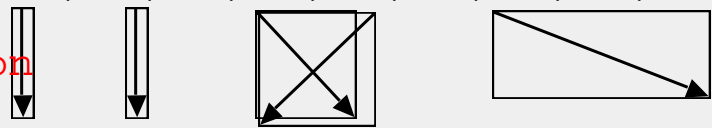
sender output trace =
(t1, v1, t2, v2, t3, v3, t4, v4, t5, v5)

What's the best QoS possible?

For a given input trace, the **ideal output trace** is generated when the service executes completely and correctly on an infinitely fast platform with unlimited resources.

The **actual output trace** is what is observed.

ideal = (10, v1, 20, v2, 30, v3, 40, v4, 50, v5)

interpretation 

actual = (12, v1, 21, v6, 44, v4, 45, v3)

actual = (12, v1, 21, v6, 45, v3, 44, v4, infinity, v5)
- ideal = (10, v1, 20, v2, 30, v3, 40, v4, 50, v5)
difference = (2, 0, 1, 4, 15, 0, 4, 0, infinity, 0)



How to model only interesting differences

An *error model* $\vec{\epsilon} = (\epsilon_1(\vec{\delta}), \dots, \epsilon_n(\vec{\delta}))$ is a vector of n functions that each map from a difference vector $\vec{\delta}$ to a real number such that $\|\vec{\epsilon}(\vec{\delta})\| = 0$ when $\|\vec{\delta}\| = 0$.

Packet service example error model:

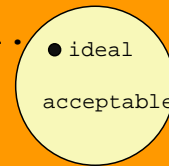
- *delay*(t) is the mean of time differences in $d(10, t, \vec{\delta})$.
- *jitter*(t) is the variance of time differences in $d(10, t, \vec{\delta})$.
- *loss*(t) is the ratio of the number of packets lost in $i(10, t, \vec{\delta})$ to the number sent, or zero if none were sent.

Error limits



What makes a good error model?

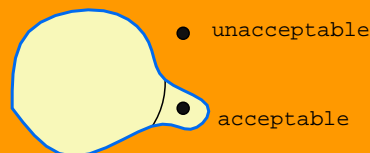
Sound: any error limits define acceptable output traces as a neighborhood of the ideal.



Complete: any trace different from the ideal can be excluded by some non-zero error limits.



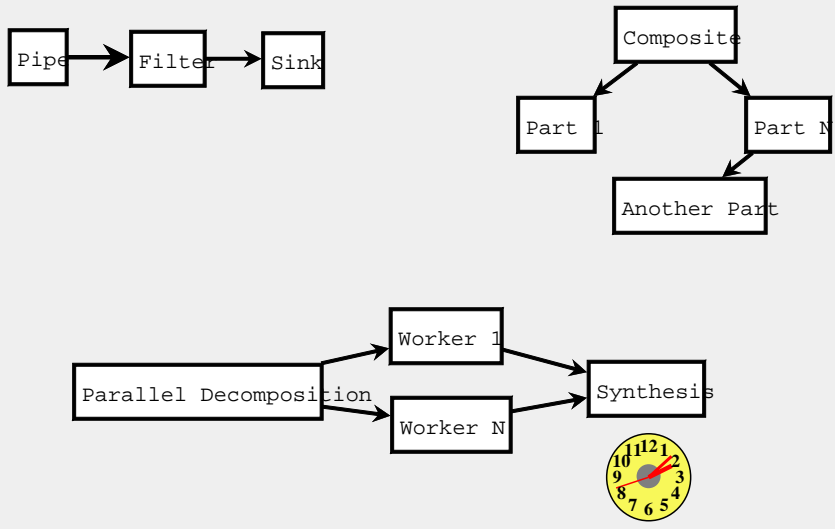
Expressive: one error model is more expressive than another if it can specify more sets of traces.



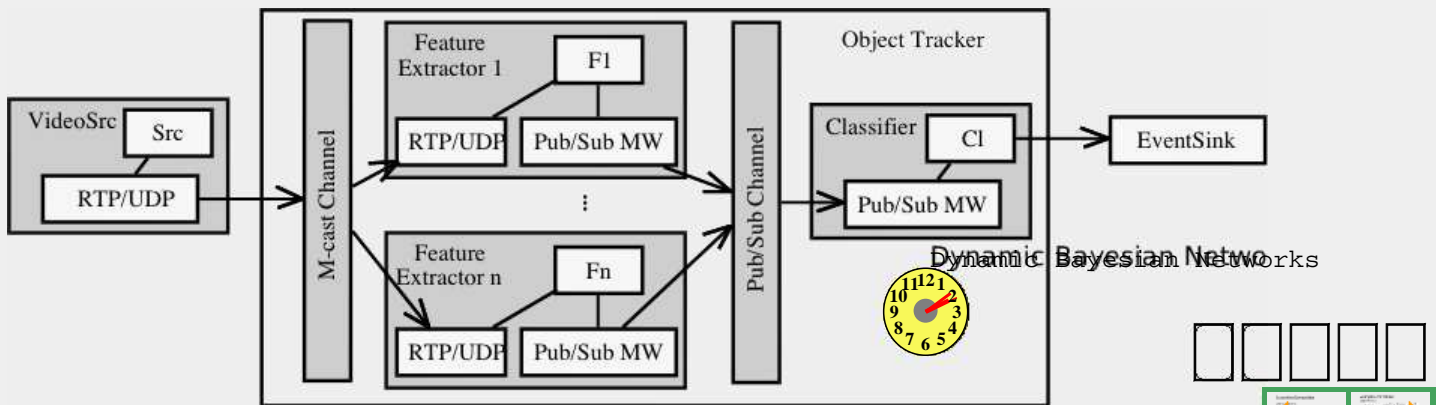
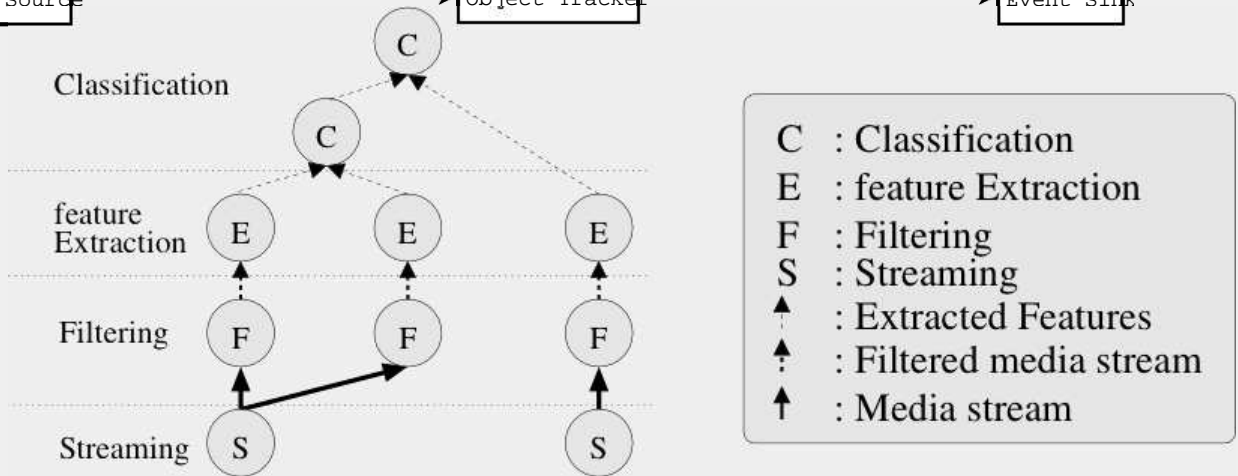
Supporting Composition

Arbitrary patterns:

pipeline, framework, backup, parallel decomposition



A Video Object Tracking Service



Example Error Models

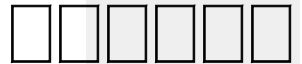
Object Tracker:

- *latency*(t) is the mean of time differences in $i(10, t, \vec{\delta})$.
- *errorRate*(t) is the ratio of non-zero location differences in $i(10, t, \vec{\delta})$ to the total number of location values.
- *period*(t) is the maximum q such that location difference is non-zero for all values in $i(q, t', \vec{\delta})$, where $t - 10 \leq t' \leq t$.

Classifier: **same error model**

Feature Extractor:

- *latency*(t) is the mean time difference in $i(10, t, \vec{\delta})$.
- *errorRate*(t) is the ratio of non-zero motion vector array differences in $i(10, t, \vec{\delta})$ to the total number of motion vector array values.

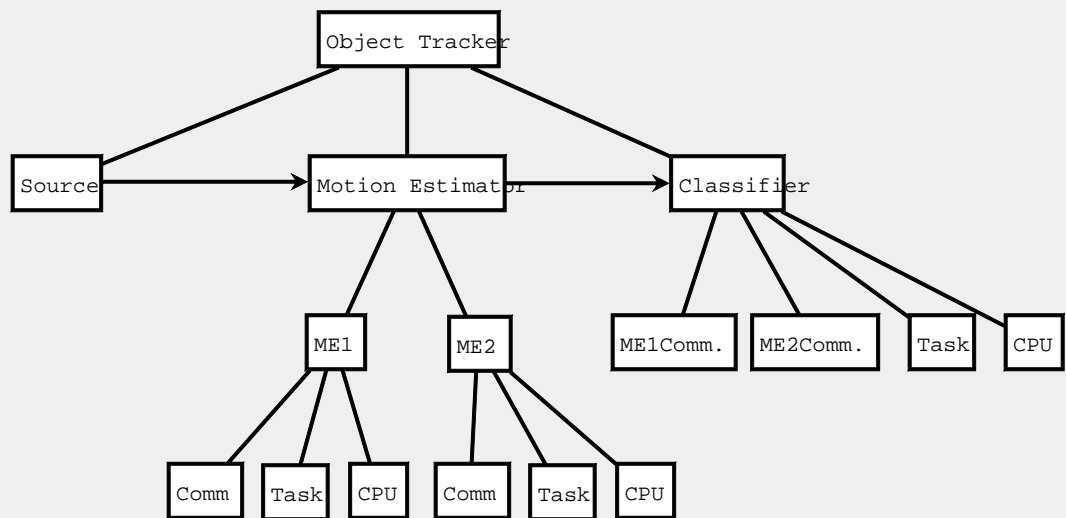


Error Prediction

```
latency(t) =  
    communication.delay +  
    avg( ME←1.latency, ...ME←n.latency) +  
    classifier.latency  
  
error←rate(t) = f( ME1.errorRate, ME1.latency(t),  
    MEn.errorRate(t), MEn.latency(t))  
  
period = max( classifier.period, source.period )  
  
ME.latency(t) = f(task, cpu.delay)  
ME.error←rate(t) = 0
```



Compositional Properties



Related Work

QuO (Schantz, et al., BBN)

Reusable adaptive QoS management.

Client-server contracts.

Informal semantics.

SLANG (Lamanna, et al., Tapas)

Very large grain components.

Acknowledged informal semantics problem.

QoS compiler (Nahrstedt, et al.)

Fixed set of QoS dimensions.

More research needed to allow uniform specification of QoS for different application domains.



Conclusions

QoS Meta Model

- basis for formal QoS specification semantics
- can model any component service
- prediction functions relate component and composition QoS

Because QoS error models do not depend on implementation, standard model may be used by both component consumers and third party developers.

Because interface location is well defined, component assemblers may provide accurate error prediction functions.

