SEAMS 2007

Implementing Adaptive Performance Management in Server Applications

Yan Liu, National ICT Australia
Ian Gorton, Pacific Northwest National Laboratory, U.S.A

May 26th, 2007

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members
NICTA Partners

Why *Adaptive*?

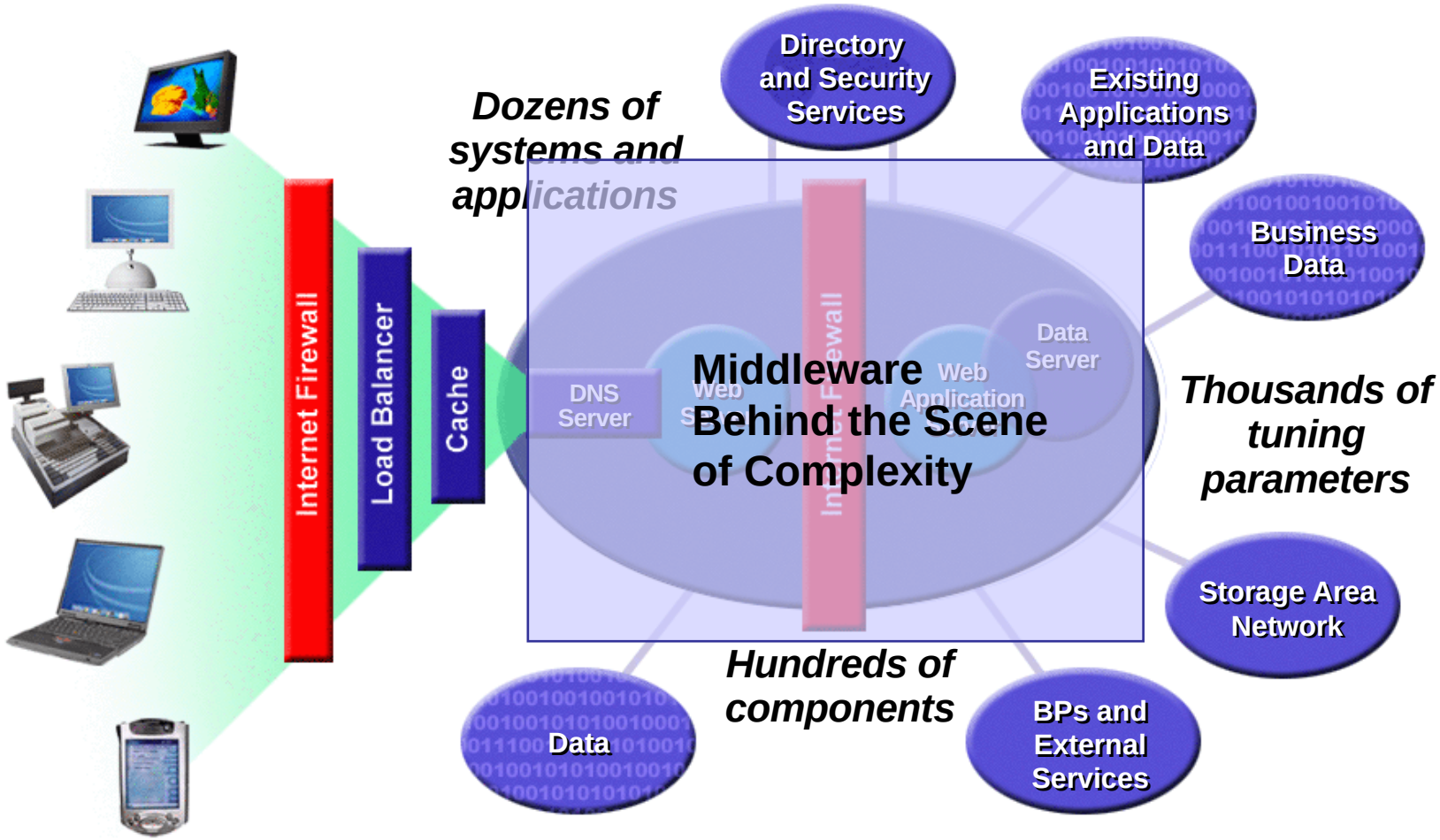
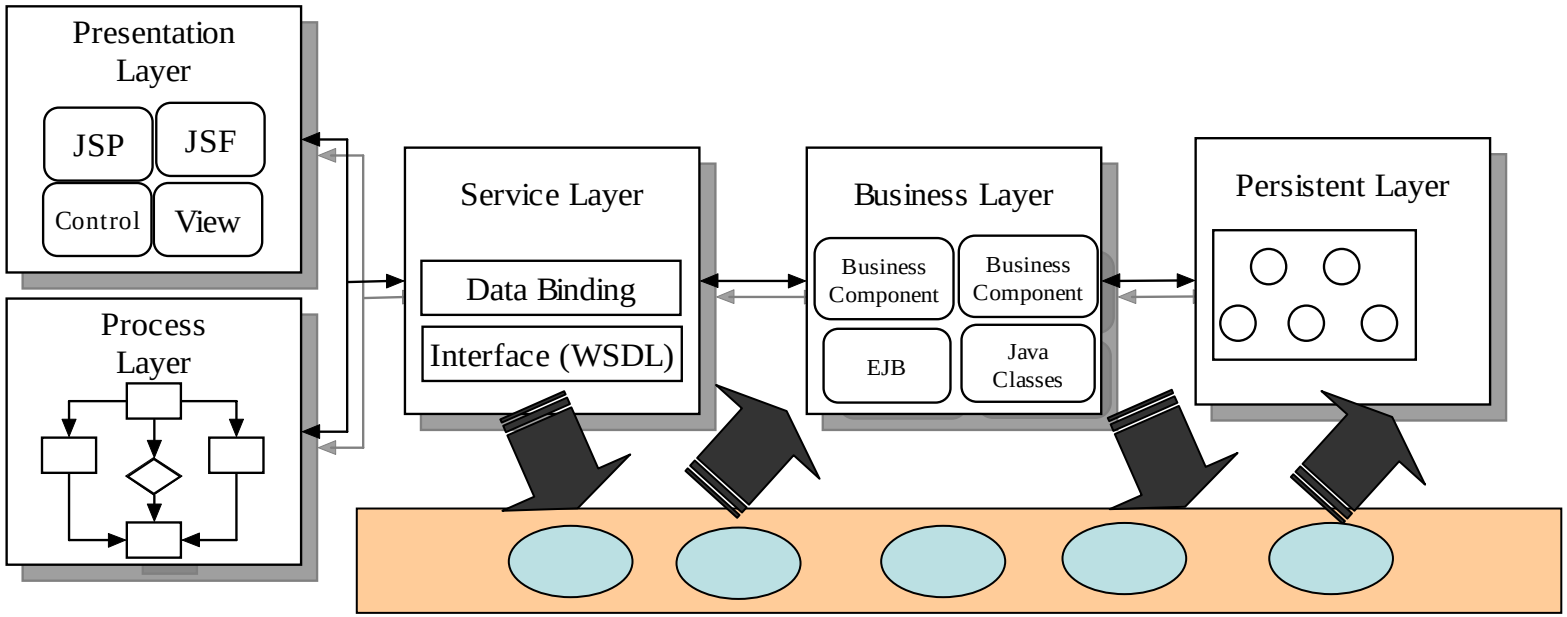


diagram from Jeff Kaphart's talk on Software Agents and the Information Economy

Adding Adaptation to the Architecture



- Separation of Concerns
- Loosely-coupled/independent
- Easy to reconfigure and incorporate new adaptation
- Policy-driven behavior dynamically adapts application to environmental changes
- Adaptive middleware provides programming models and services

Scientific Challenges

- The creation of adaptation capabilities presents the following scientific challenges:
  - the invention of new extensible software architectures
    - to provide flexible application server monitoring and feedback mechanisms, with minimal intrusion.
  - the invention of mechanisms to enable dynamic monitoring and adaption policies
    - reflection mechanisms
    - policies can be specified declaratively by the application designer.
  - the invention of predictive models
    - to be solved to produce optimal parameter settings for controlling various aspects of application configuration and behavior.

The Role of Models in Adaptation

- Applying models at design, implementation, and deployment stages of development
- Model-based and model-driven techniques for validating and monitoring run-time behavior
- Modeling is well supported by established theories

A key benefit is that models can be used to provide a richer semantic base for runtime decision-making related to system adaptation and other runtime concerns.

The Approach

Construction of models

- Convert performance goals into metrics
- Determine the impact of adaptive behaviors
- Calibrate application specific factors

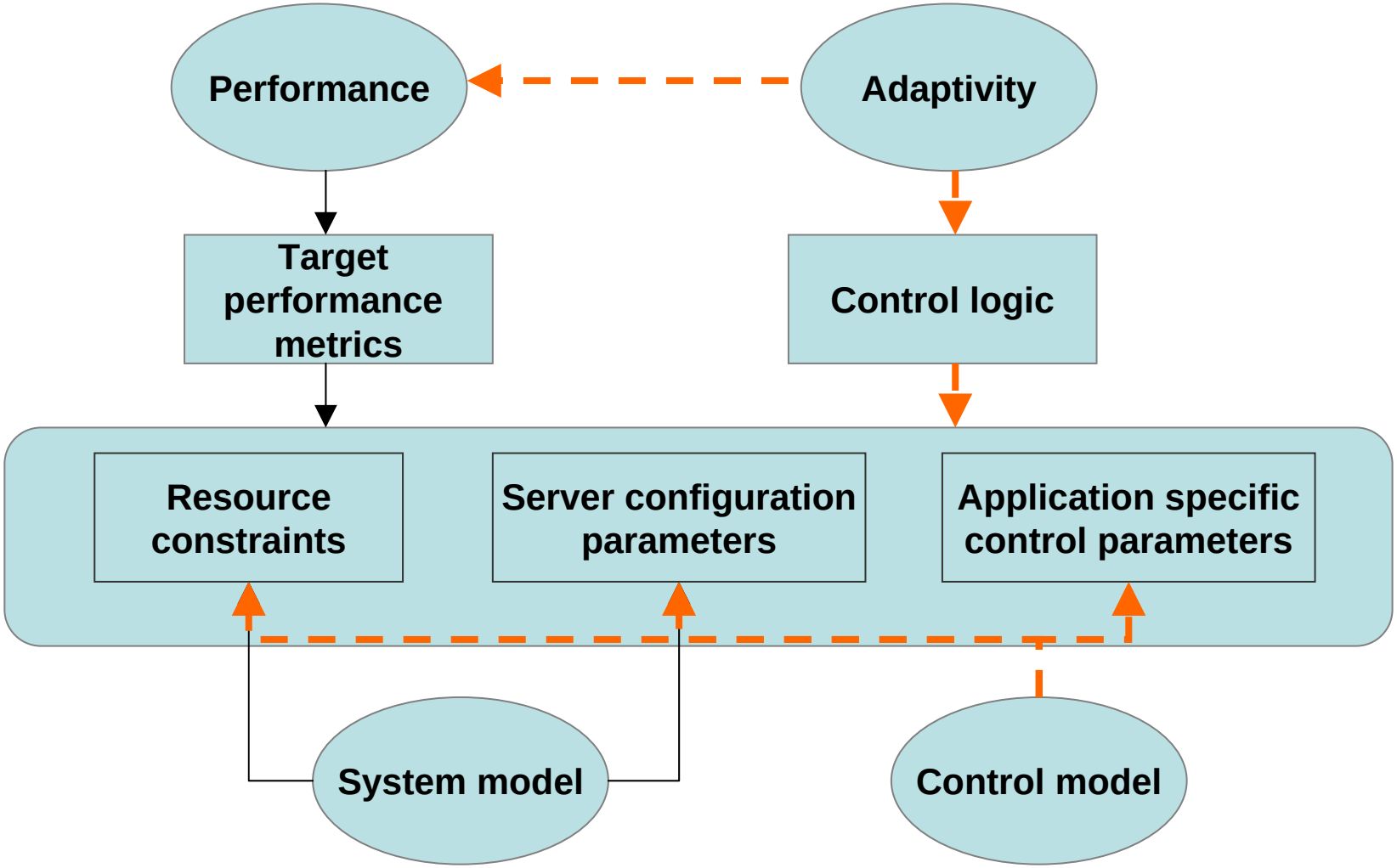
Construction of control loops

- Monitor the execution of the application
- Analyze the data collected
- Plan the necessary responses
- Execute actions to enforce the adaptive behavior

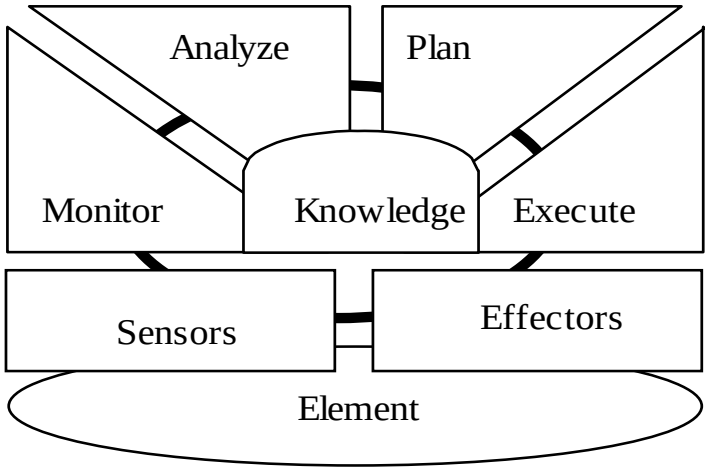
Standard-based architecture design

- identify the mechanisms to implement sensors and effectors
- design communication patterns to compose these components
- achieve non-intrusive performance adaptation

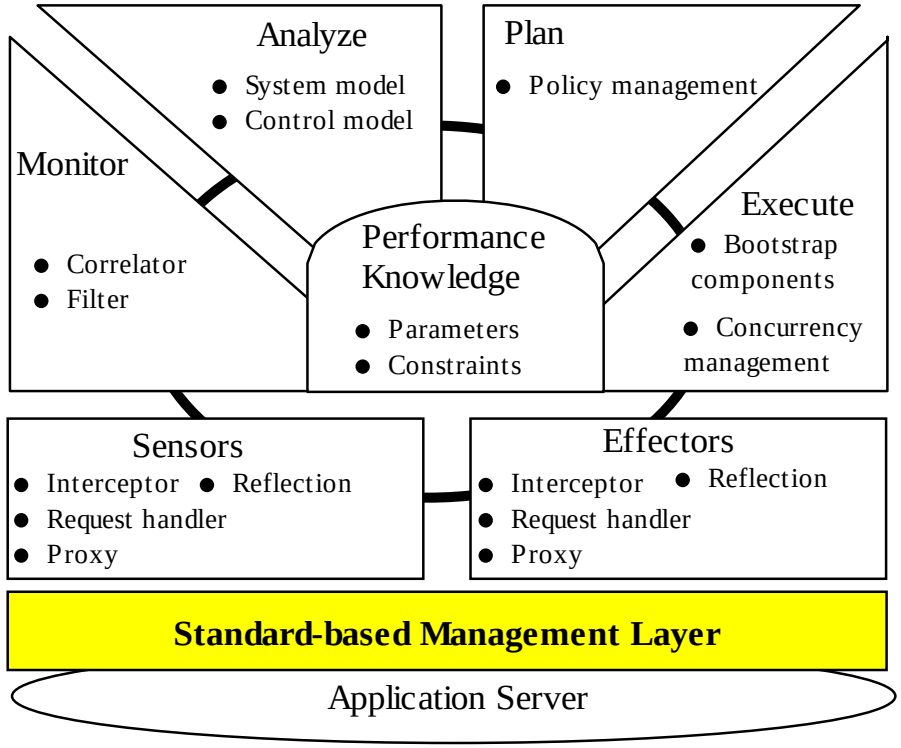
Construction of Models



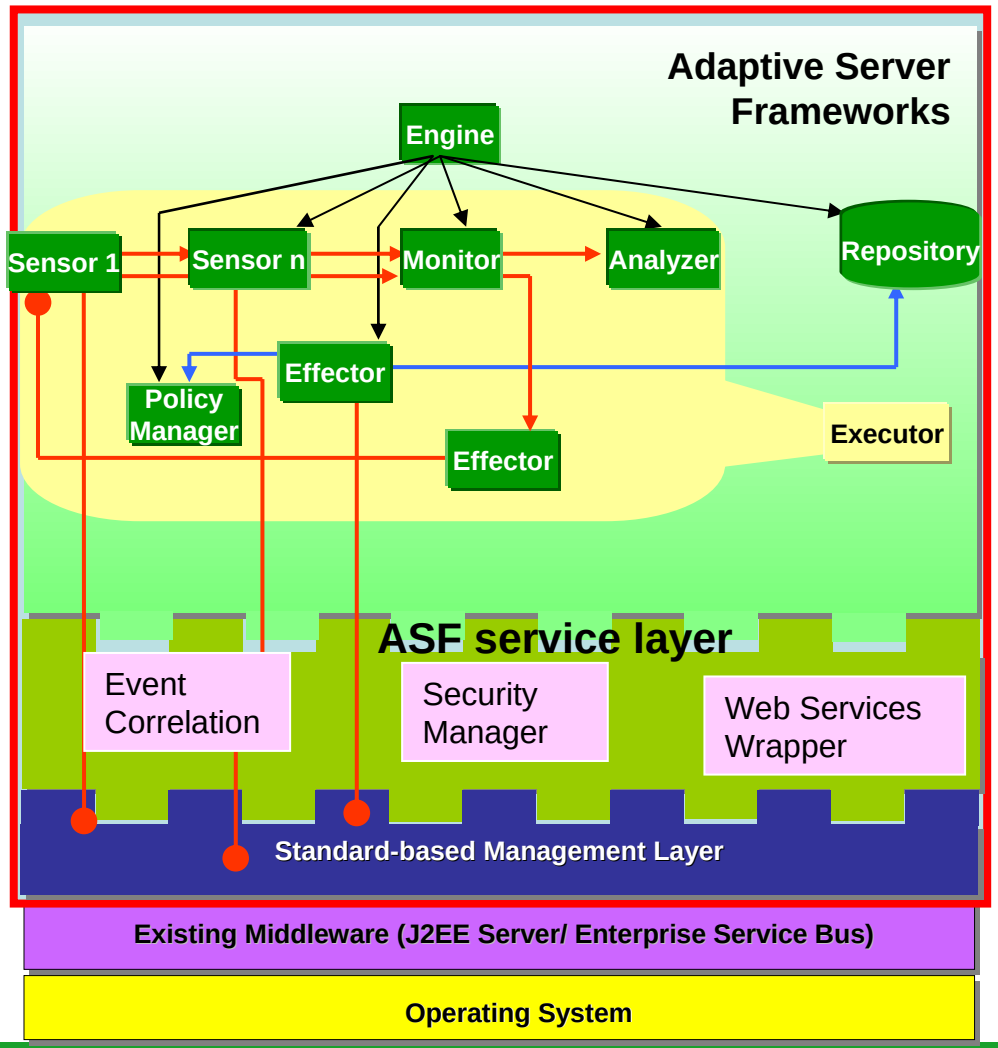
Construction of Control Loops



An Autonomic Element



Standard based Architecture Design



NICTA Adaptive Middleware Platform

Quality Requirement

Policy Definition Tools

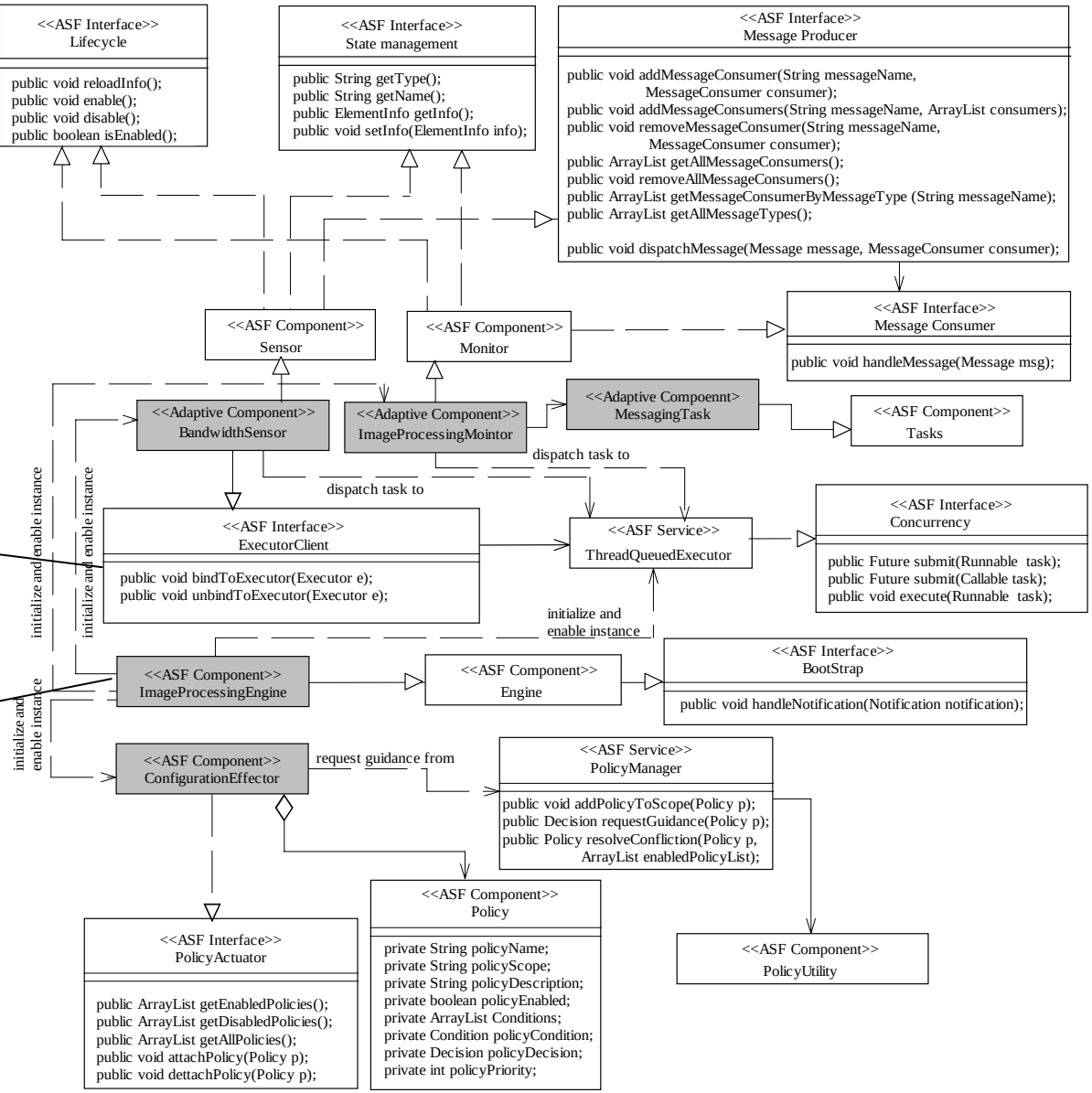
Framework and programming models

Components and services

Adaptivity

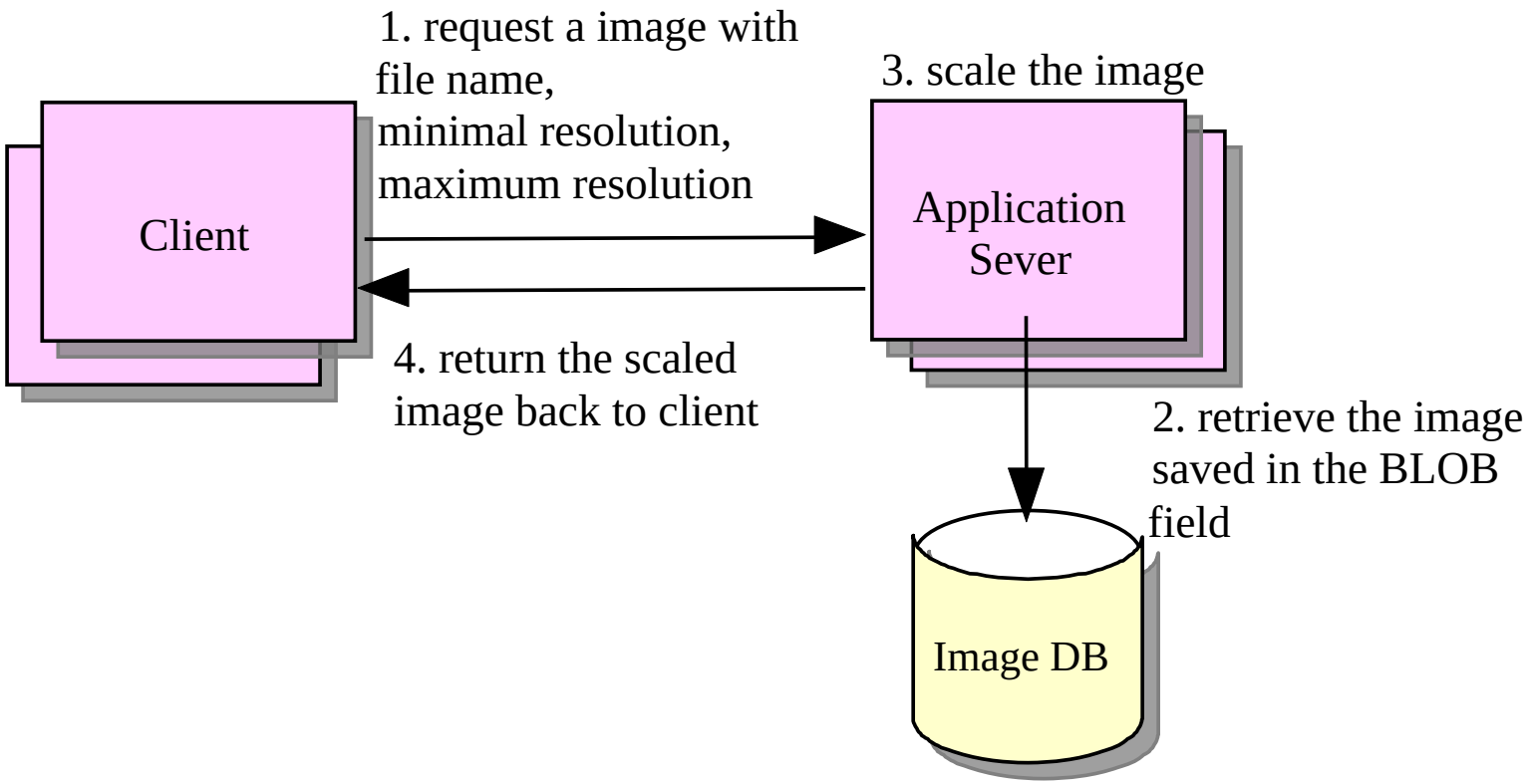
ASF pre-defined component with default implementation

Customized, Application specific Adaptive component

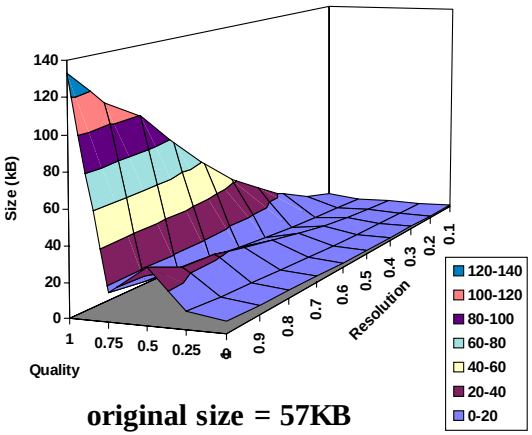


Case study - An Adaptive Image Server

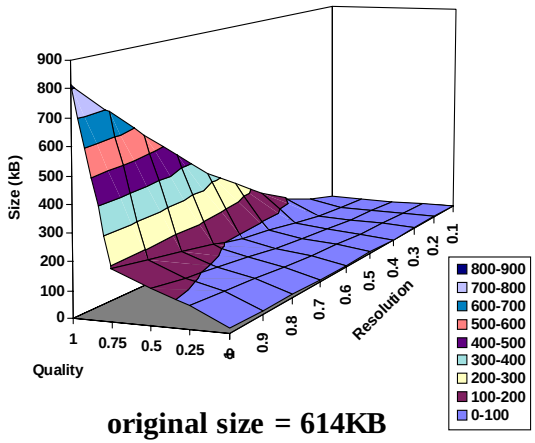
- Simulating image servers (a J2EE implementation)



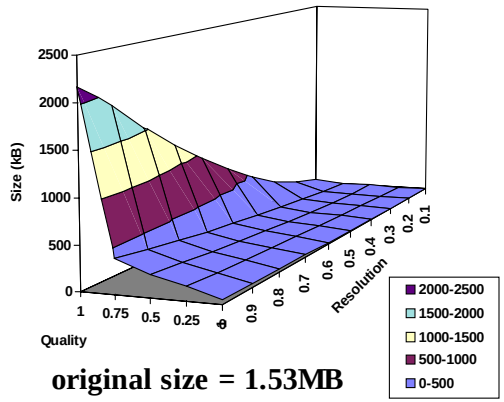
Observations



original size = 57KB

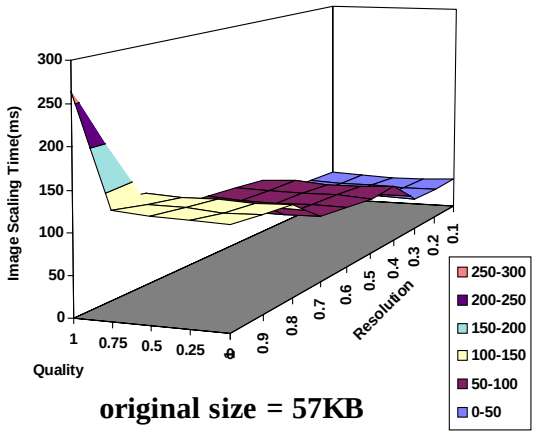


original size = 614KB

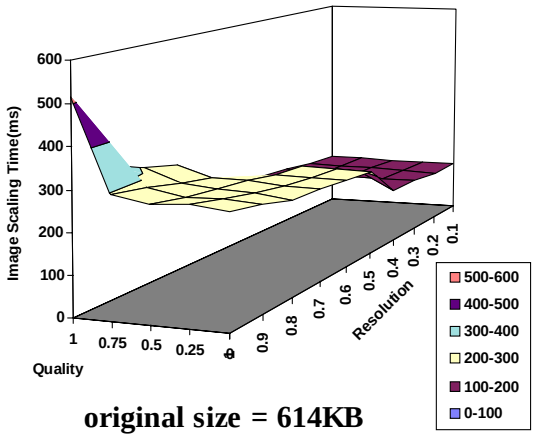


original size = 1.53MB

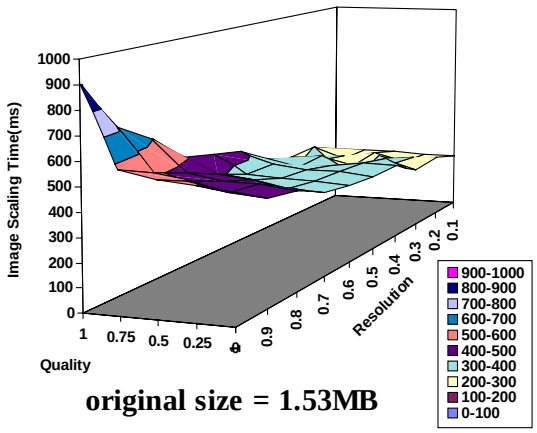
(a) Scaled Image Size



original size = 57KB



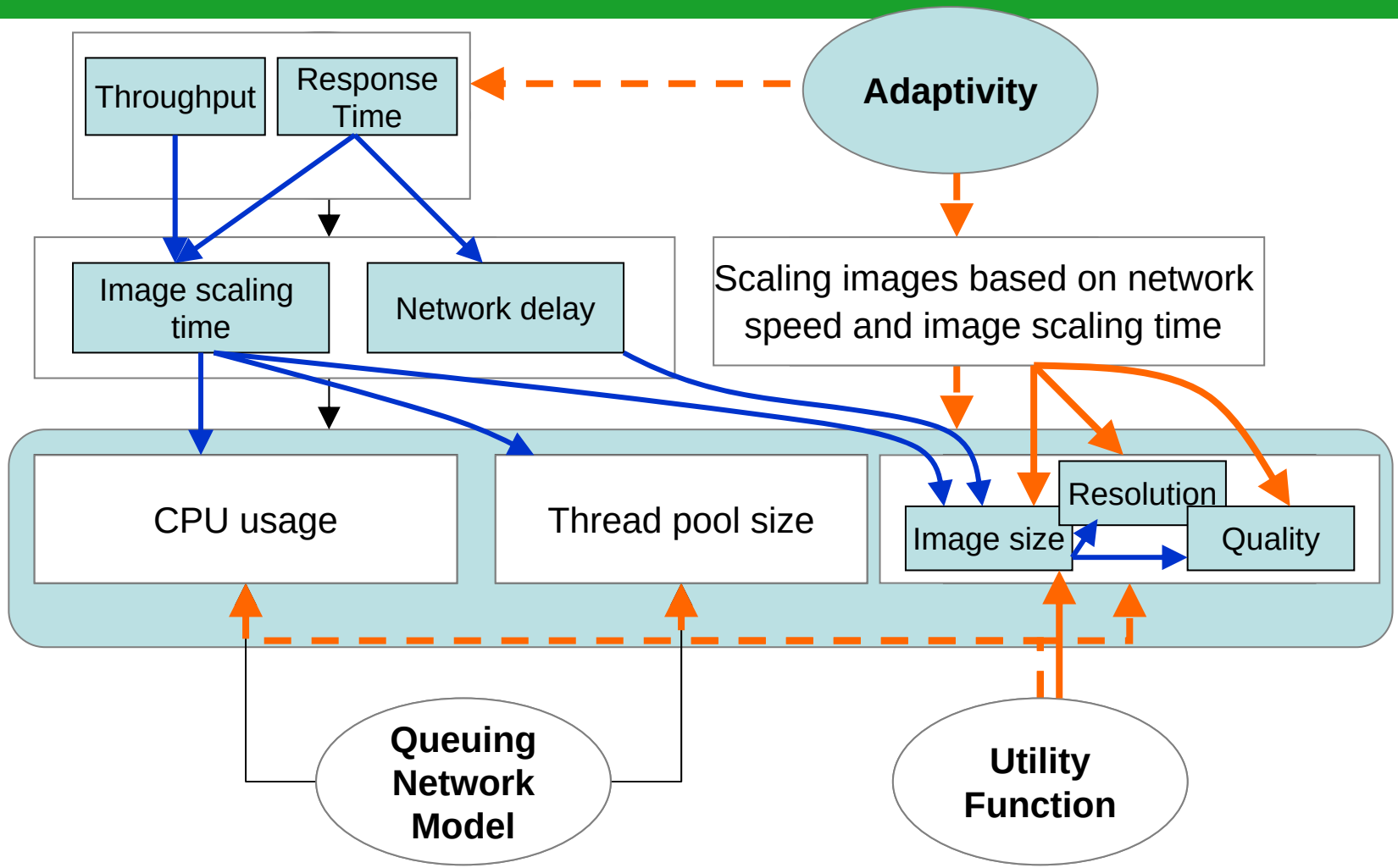
original size = 614KB



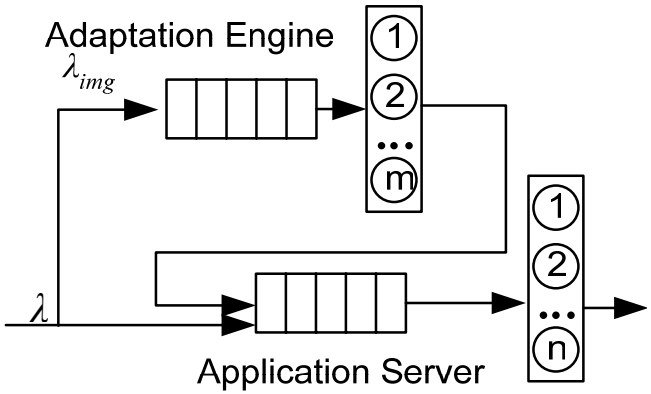
original size = 1.53MB

(b)Image Scaling Time

Derived Control Loops



Construction of Models

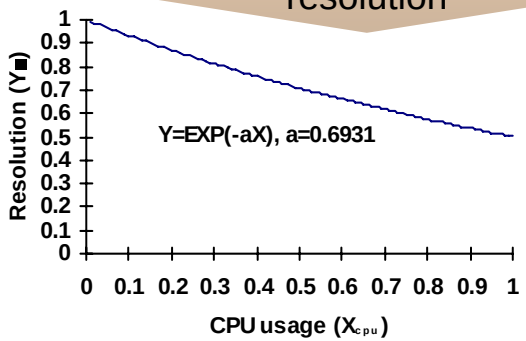


Construction of Queueing Network Models

Service demand, CPU usage and Response time

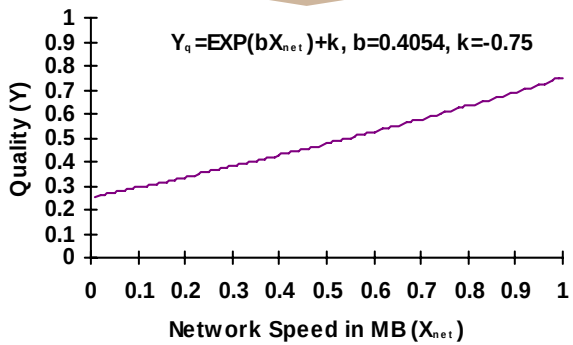
Construction of Utility Function

CPU usage and Image resolution

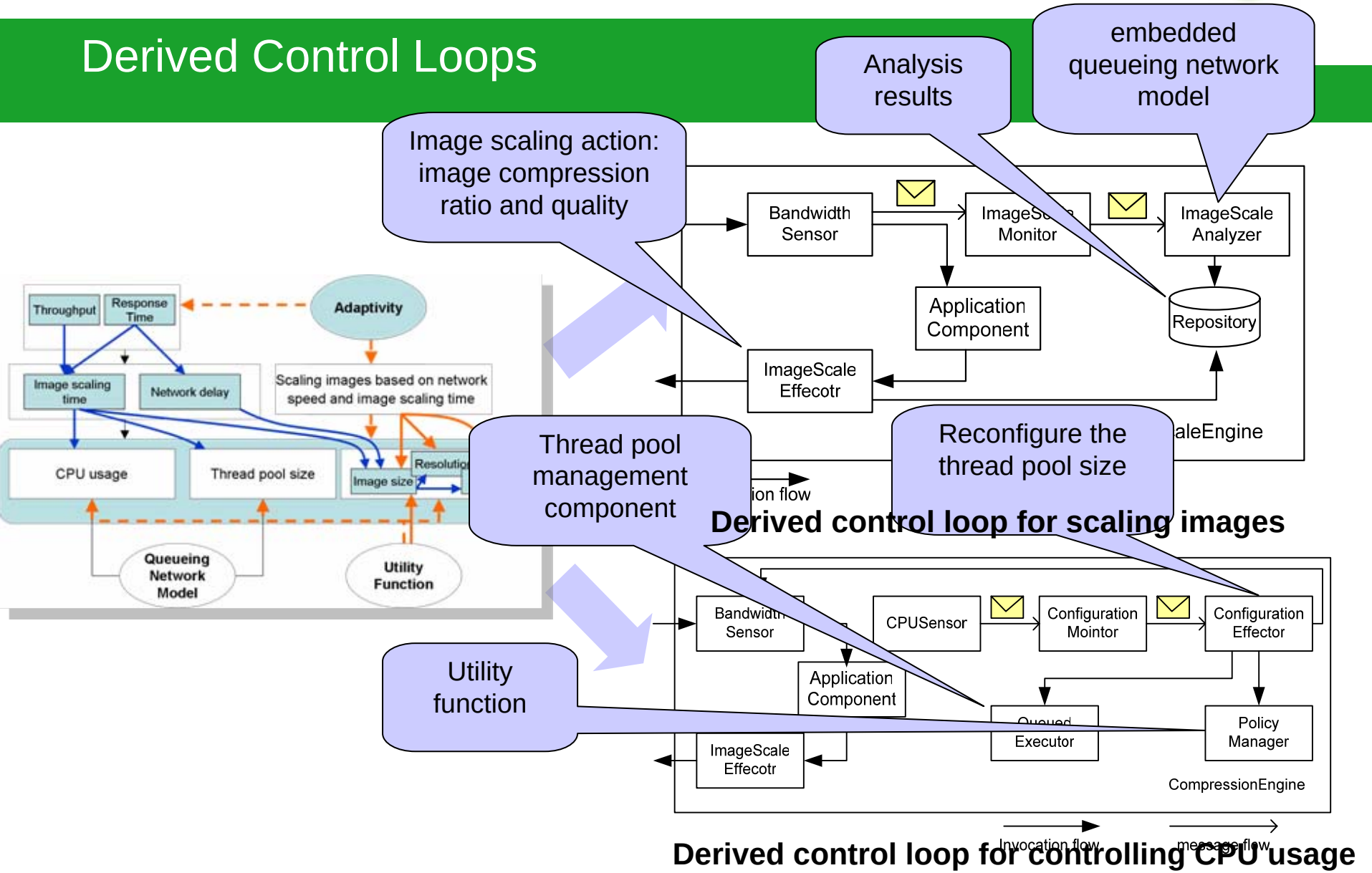


Construction of Utility Function

Network speed and Image resolution



Derived Control Loops

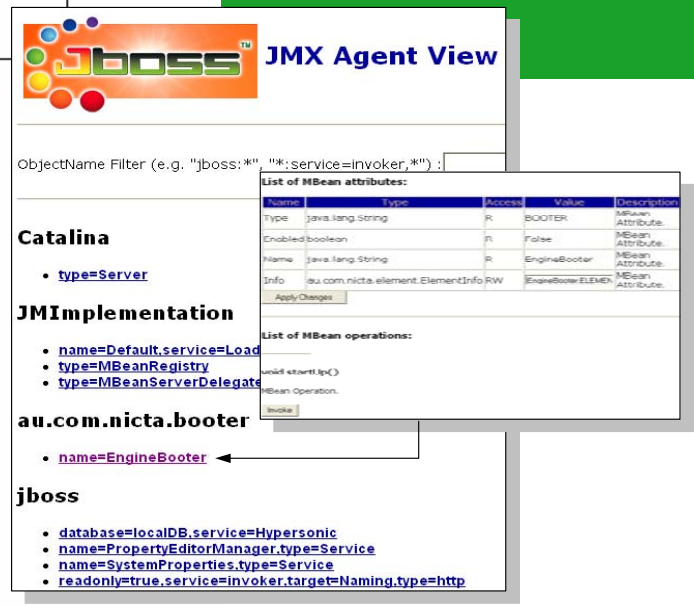


```
public class BandwidthSensor extends AbstractInterceptor
                        implements Sensor, Serializable{
  //....
  public Object invoke(Invocation mi)throws Exception
  {
     //code for collecting the network connection speed
  }
 //....
}
```

```
public class ImageScaleEffector extends AbstractInterceptor
   implements Effector, MessageProducer,RepositoryClient,
   AutonomicExecutorClient{
   //....
   public Object invoke(Invocation mi)throws Exception
   {
      //code for setting the invocation parameters
     // with calculated resolution
   }
  //....
}
```

```
public class ImageProcessBean implements SessionBean {
     public byte[] getImage(String imageName,float minResolution,float maxResolution)
      throws RemoteException{
          ImageMetrics im = imgPro.getImageInByteArray(imageName);
           // ImageProcessor imgPro = new ImageProcessor();
          if(maxResolution == 0)
               bi = im.ImageByteArray;
          else bi = imgPro.getScaledImageByGetInstance(im,maxResolution,defaultQ);

          return bi;
     }
}
```

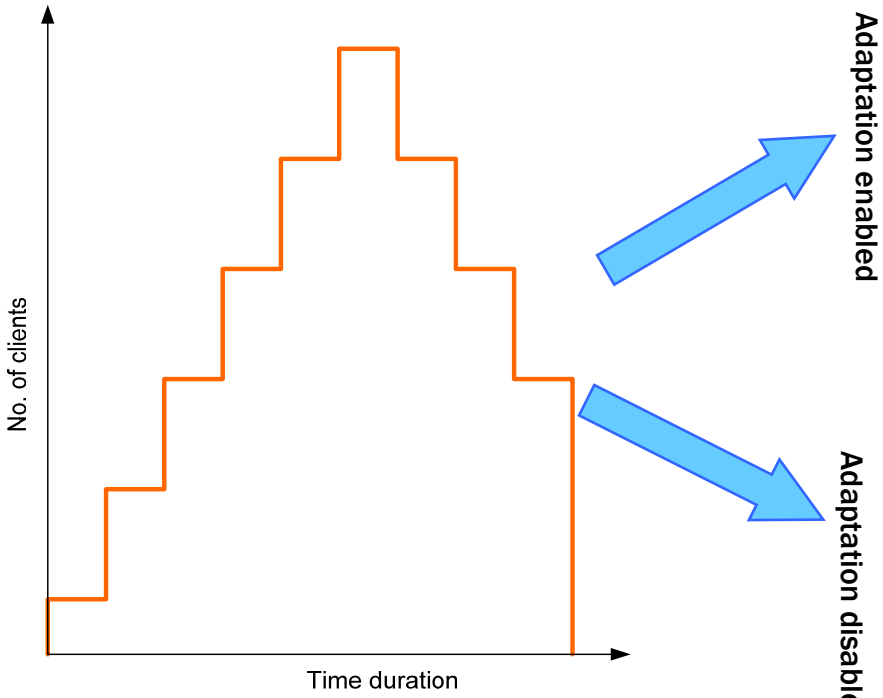



```
ner-configurations>
ntainer-configuration>
<container-name>Standard Stateless SessionBean</container-name>
<call-logging>true</call-logging>
<invoker-proxy-binding-name>stateless-rmi-invoker</invoker-proxy-binding-name>
<container-interceptors>
    <interceptor>org.jboss.ejb.plugins.ProxyFactoryFinderInterceptor</interceptor>
    <interceptor>org.jboss.ejb.plugins.LogInterceptor</interceptor>
    <interceptor>org.jboss.ejb.plugins.SecurityInterceptor</interceptor>
    <interceptor >au.com.nicta.sensor.impl.BandwidthSensor</interceptor>
    <interceptor transaction="Container">org.jboss.ejb.plugins.TxInterceptorCMT</interceptor>
    <interceptor transaction="Container">org.jboss.ejb.plugins.CallValidationInterceptor</interceptor>
    <interceptor transaction="Container" metricsEnabled="true">org.jboss.ejb.plugins.MetricsInterceptor</interceptor>
    <interceptor transaction="Container">org.jboss.webservice.server.ServiceEndpointInterceptor</interceptor>
    <interceptor transaction="Container">org.jboss.ejb.plugins.StatelessSessionInstanceInterceptor</interceptor>
    <interceptor transaction="Bean">org.jboss.ejb.plugins.StatelessSessionInstanceInterceptor</interceptor>
    <interceptor transaction="Bean">org.jboss.ejb.plugins.TxInterceptorBMT</interceptor>
    <interceptor transaction="Bean">org.jboss.ejb.plugins.CallValidationInterceptor</interceptor>
    <interceptor transaction="Bean" metricsEnabled="true">org.jboss.ejb.plugins.MetricsInterceptor</interceptor>
    <interceptor transaction="Bean">org.jboss.webservice.server.ServiceEndpointInterceptor</interceptor>
    <interceptor>org.jboss.resource.connectionmanager.CachedConnectionInterceptor</interceptor>
    <interceptor transaction="Container">au.com.nicta.effector.image.ImageScaleEffector</interceptor>
</container-interceptors>
```

ASF sensor and effector deployment in JBoss application server

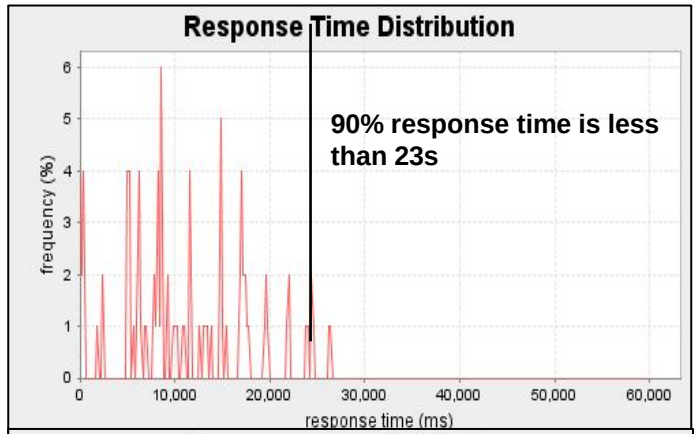
Evaluation

Image Size	Non-adaptive TPS	Adaptive TPS	% Improvement
Small (10KB)	54.25	89.23	64.48%
Small Medium (10~100KB)	5.78	8.95	54.84%
Medium (100~500KB)	1.94	3.10	59.79%
Large (500KB~2MB)	0.25	0.31	24%

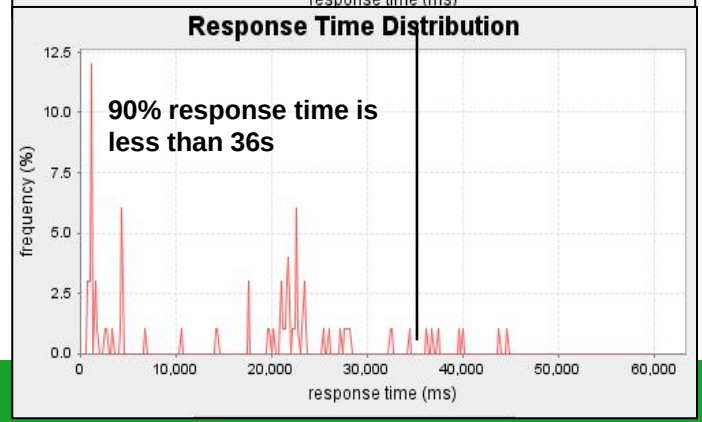


Simulated workload with peak workload as 65 clients.

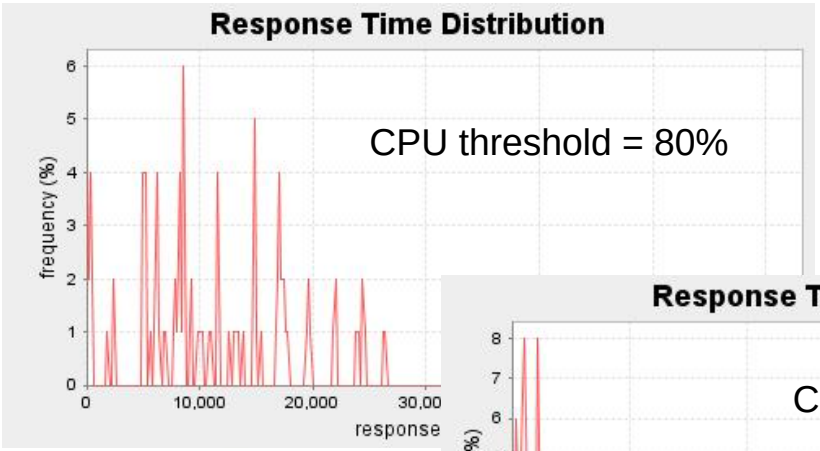
Adaptation enabled

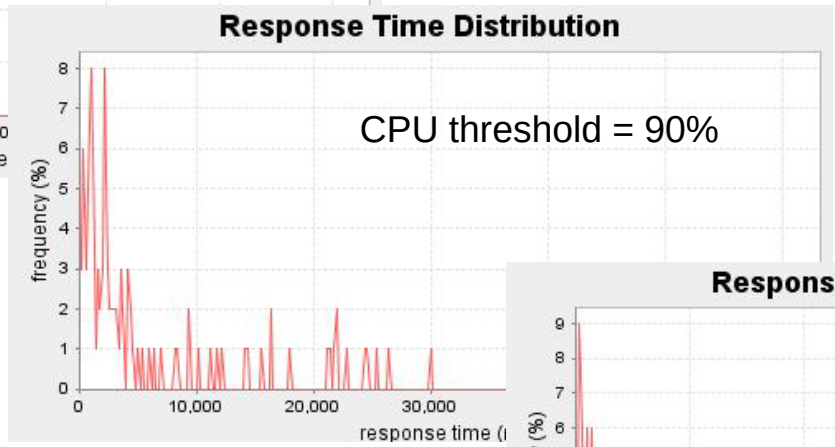


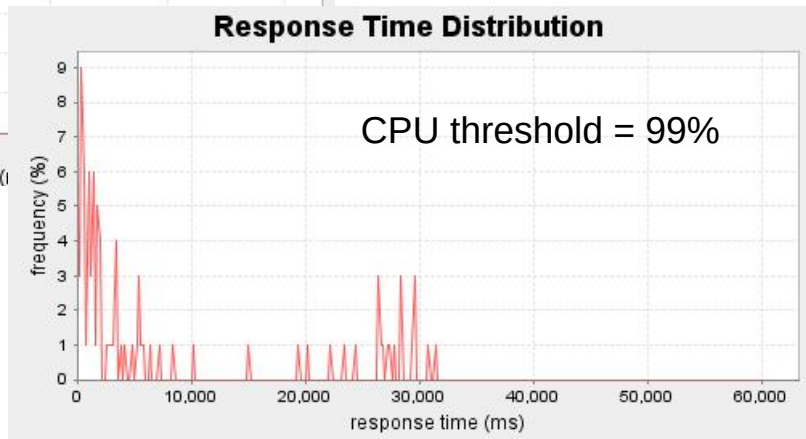
Adaptation disabled



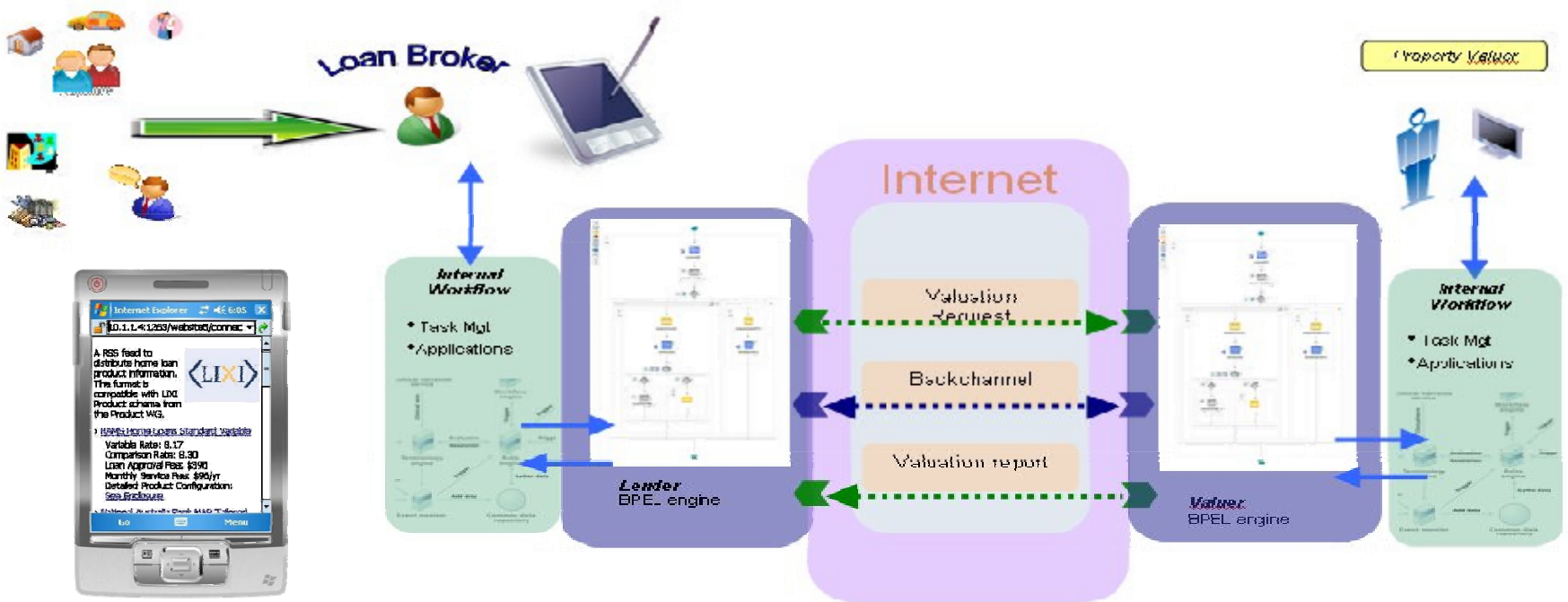
Effects of Dynamic Parameters







On Going Work: enabling adaptive mobile applications for rich context



Related Work

- Integrating models with autonomic computing
  - By Marin Litoiu, Murray Woodside, Tao Zheng, 'Hierarchical Model-based Autonomic Control of Software Systems', DEAS 2005.
  - Project led by Prof. Menasce, Autonomic Computing Through Analytic Performance Models, George Mason University.
  - Project led by Prof. Schwan, 'Developing Prediction Services in Distributed Systems', Georgia Institute of Technology.
- Software Architectures and techniques
  - Project led by Prof. Garlan, 'Rainbow - Architecture-based Adaptation of Complex Systems', CMU.
  - By P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, 'Composing Adaptive Software', *IEEE Computer*, 37(7):56-64, July 2004
- And many more…
  - ICAC, SEAMS, WOSS, DEAS….

Lessons Learnt and Conclusion

- Understanding application specific adaptation requirements
- Developing analytical models
- Programming and infrastructure support
- Testing and maintenance
- System under evolution