A Study of Execution Environments for Software Components

Kung-Kiu Lau and Vladyslav Ukis

School of Computer Science

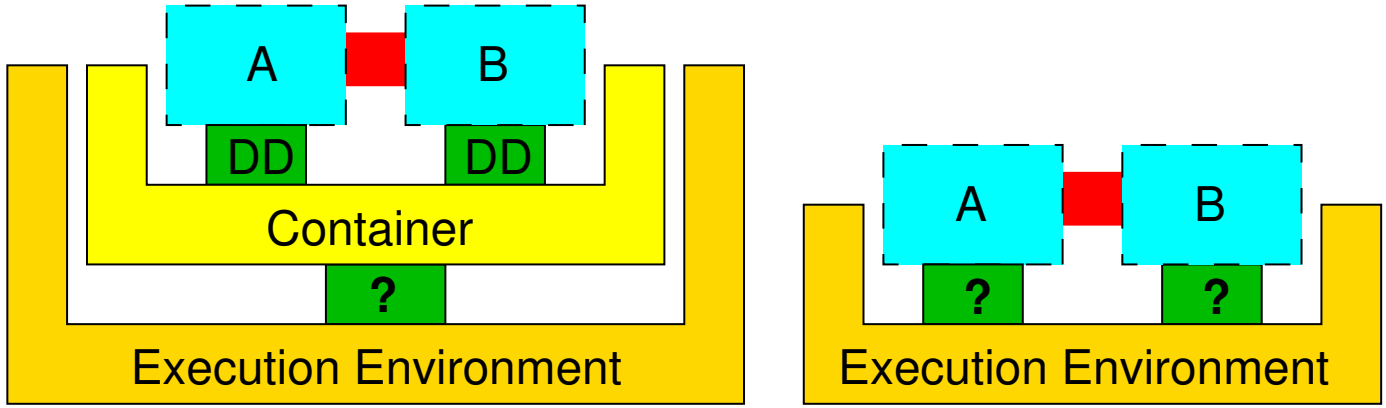
The University of Manchester, UK

{kung-kiu,vukis}@cs.man.ac.uk

Overview

- Common execution environments for software components (desktop and web)
- Analysis of these environments based on J2EE and .NET
- Results can be used to check component compatibility with these environments

Execution Environments for Software Components



- An execution environment is not a container
- Components may run in an execution environment without a container
- A container runs in an execution environment

Widely used Component Execution Environments

Current ubiquitous execution environments for components:

- Desktop (standalone)
- Web Server (internet)

They are supported by industrial platforms

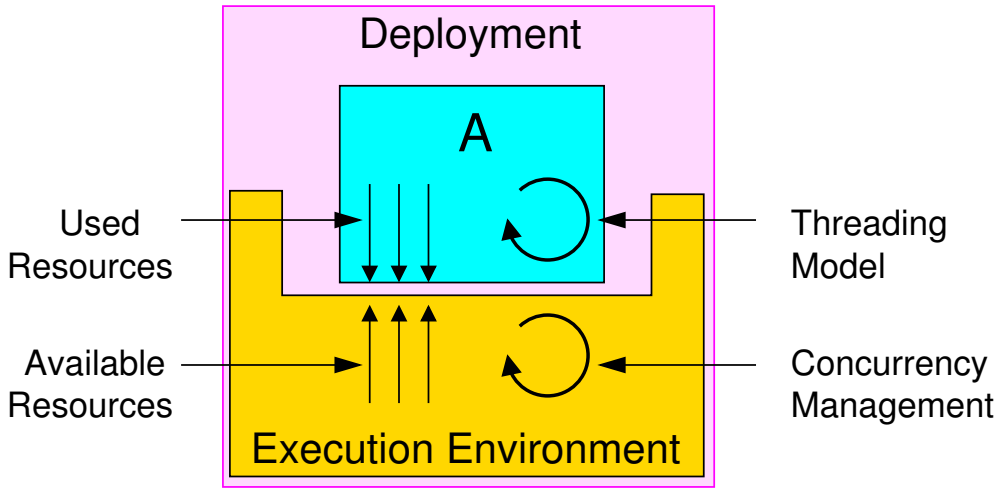
- J2EE (OSGi) — EJB (desktop), JSP (web server)
- .NET — default (desktop), ASP.NET (web server)

Our analysis of execution environments is based on these platforms.

Component Deployment

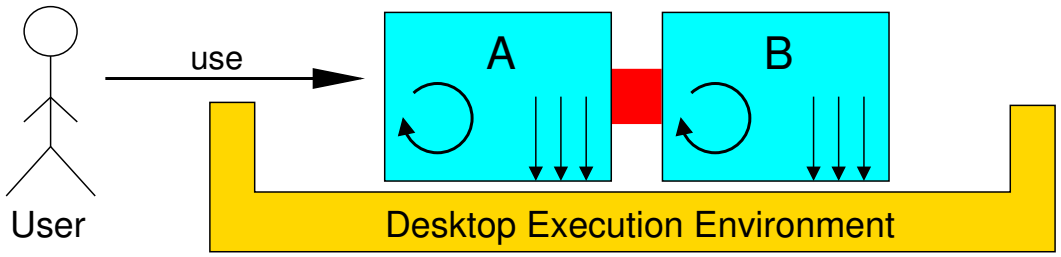
- Question: Given a binary component, how can we determine which execution environments it can be deployed into?
- We will analyse desktop and web environments as used in J2EE and .NET platforms to give the answer

Properties of Interest in an Execution Environment



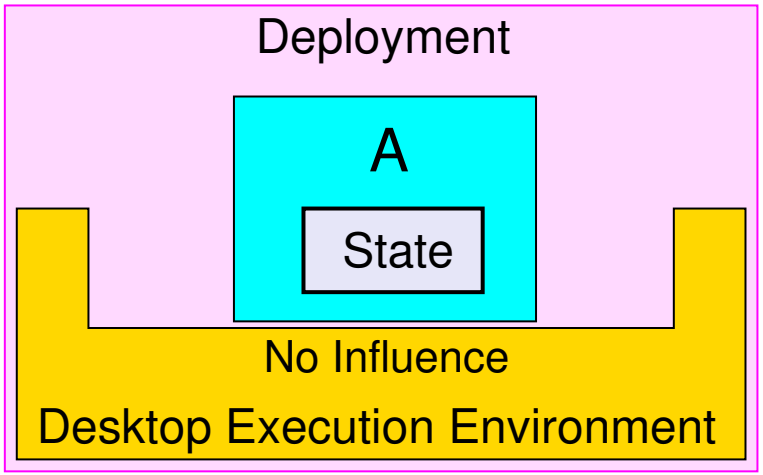
- In an execution environment we need to know
  - Available resources
  - Concurrency management
  - Transient state management
- With this knowledge, component's compatibility with the execution environment can be checked

Desktop Execution Environment



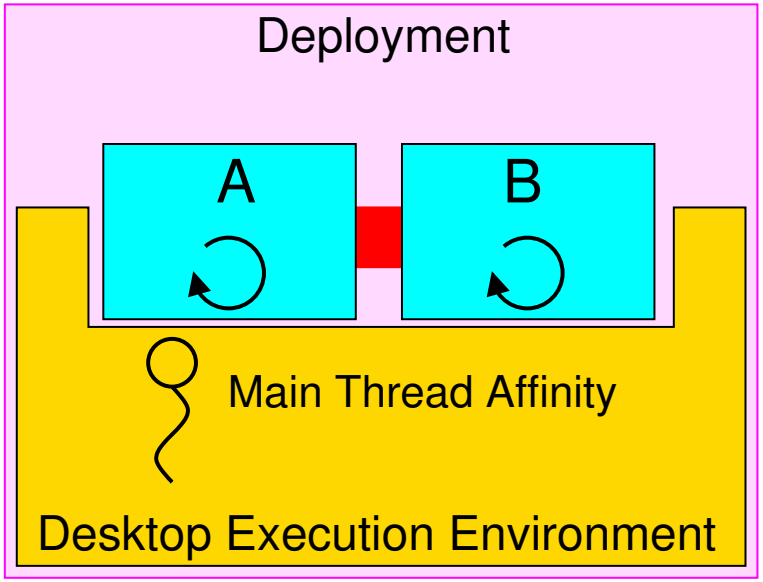
- Example systems
  - Adobe Acrobat Reader
  - Ghost Viewer
  - Unix commands, e.g. ls or ps
- A single user interacts with the system

Desktop Environment: Transient State Management



- Desktop environment does not influence component state

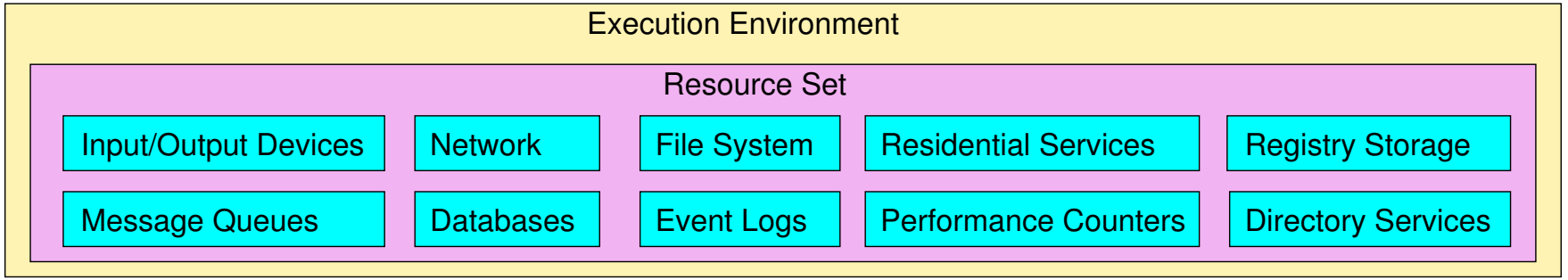
Desktop Environment: Concurrency Management



- The main thread is guaranteed to be one and the same for all requests to a system instance
- Thread affinity in components is preserved by the desktop environment

Resource Availability

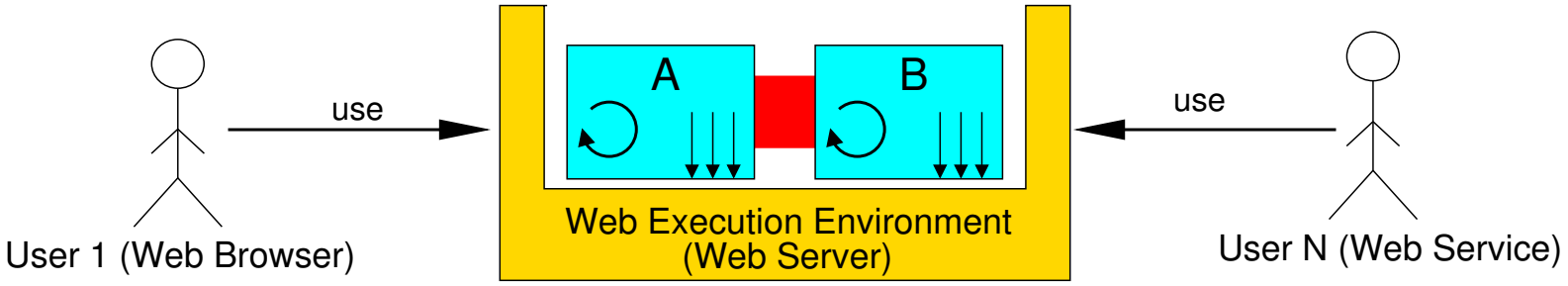
We have defined the following resource set:



- for both desktop and web environment
- by looking at J2EE and .NET APIs that enable access to resources

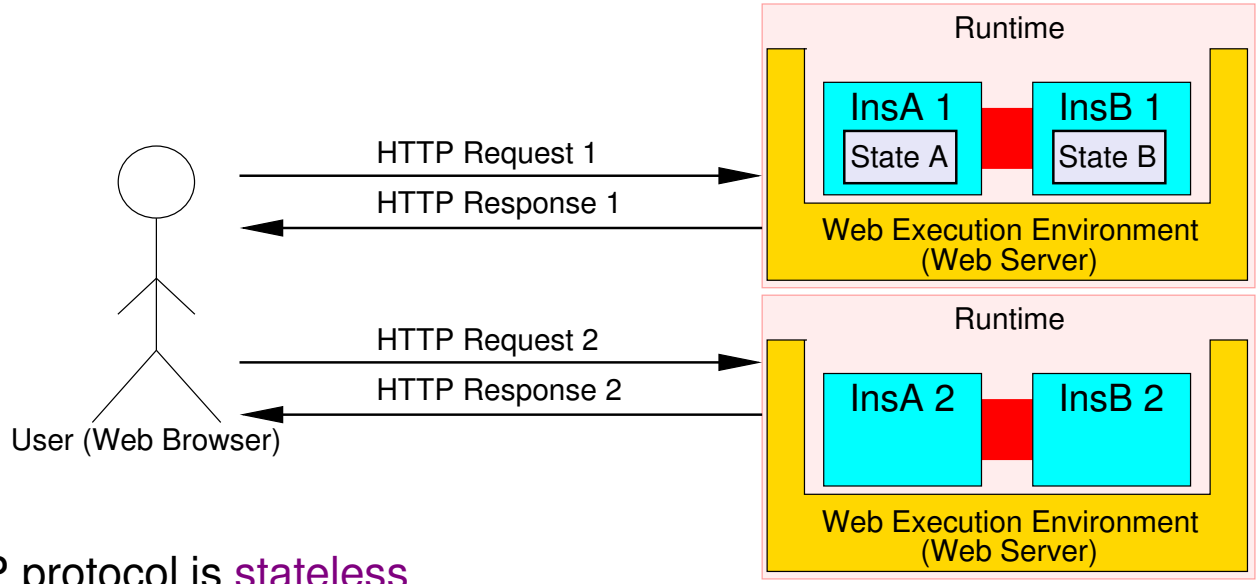
The resource set is restricted to and complete with respect to J2EE and .NET APIs

Web Execution Environment



- Example systems
  - Amazon
  - ebay
  - Google
- There are many users, possibly simultaneously, interacting with the system

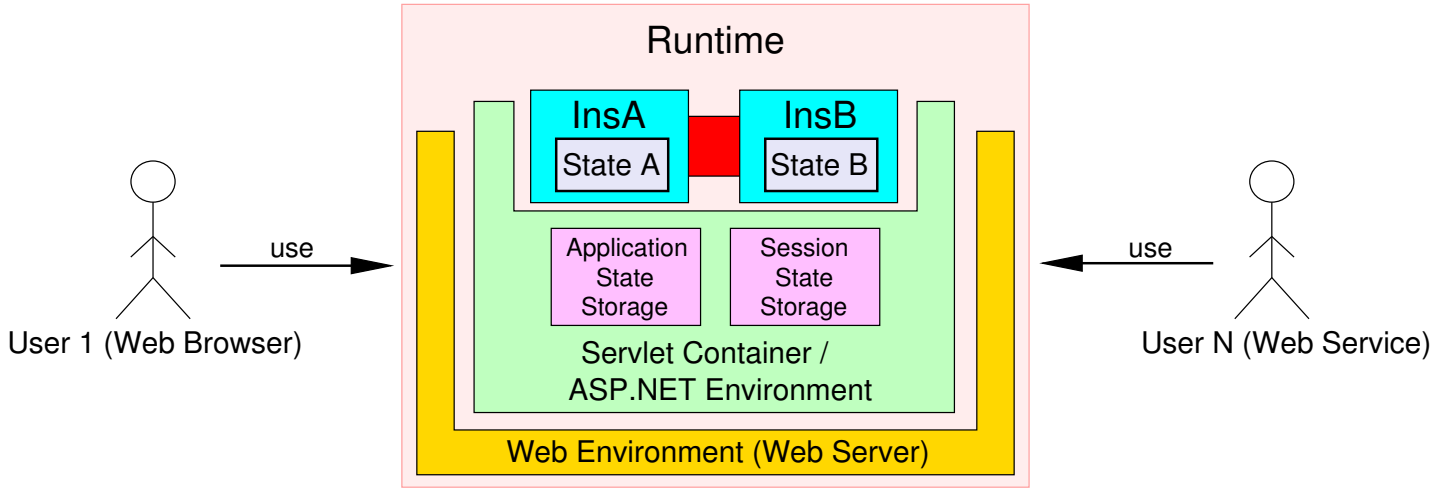
Web Environment: Transient State Management



- HTTP protocol is stateless
  - Stateless components execute smoothly in such an environment
  - Stateful components lose their state and fail
- Technologies that enable state retention in web environment
  - Java Server Pages (JSP)
  - Active Server Pages .NET (ASP.NET)

Web Environment: Transient State Management (Continued)

	HTTP	ASP.NET	JSP
System state retained between requests	No	No	Yes
Application state storage		Yes	Yes
Session state storage		Yes	Yes



Web Environment: Concurrency Management

- A system is exposed to, theoretically, an unlimited number of concurrent users
- The web server spawns a thread for each request it receives
- Thus, the main thread is not guaranteed to be the same for every request to a system instance during its lifetime
- Request handling depends on technology used for web application development: HTTP, JSP or ASP.NET

  (need to consider system instantiation modes: per application, per session, per request etc.)

Concurrency Problems in Web Environment

- State corruption problem:
  - Component's state is accessed concurrently by multiple threads and is not protected by a thread synchronisation primitive
- Thread affinity problem:
  - Component's thread affine elements are not always accessed by one and the same thread
- Shared statics and singletons problem:
  - a component contains static variables or singletons
  - the system is instantiated more than once
  - the system instances execute concurrently
  - statics/singletons are not used by a thread-safe component part

Desktop v Web Execution Environment

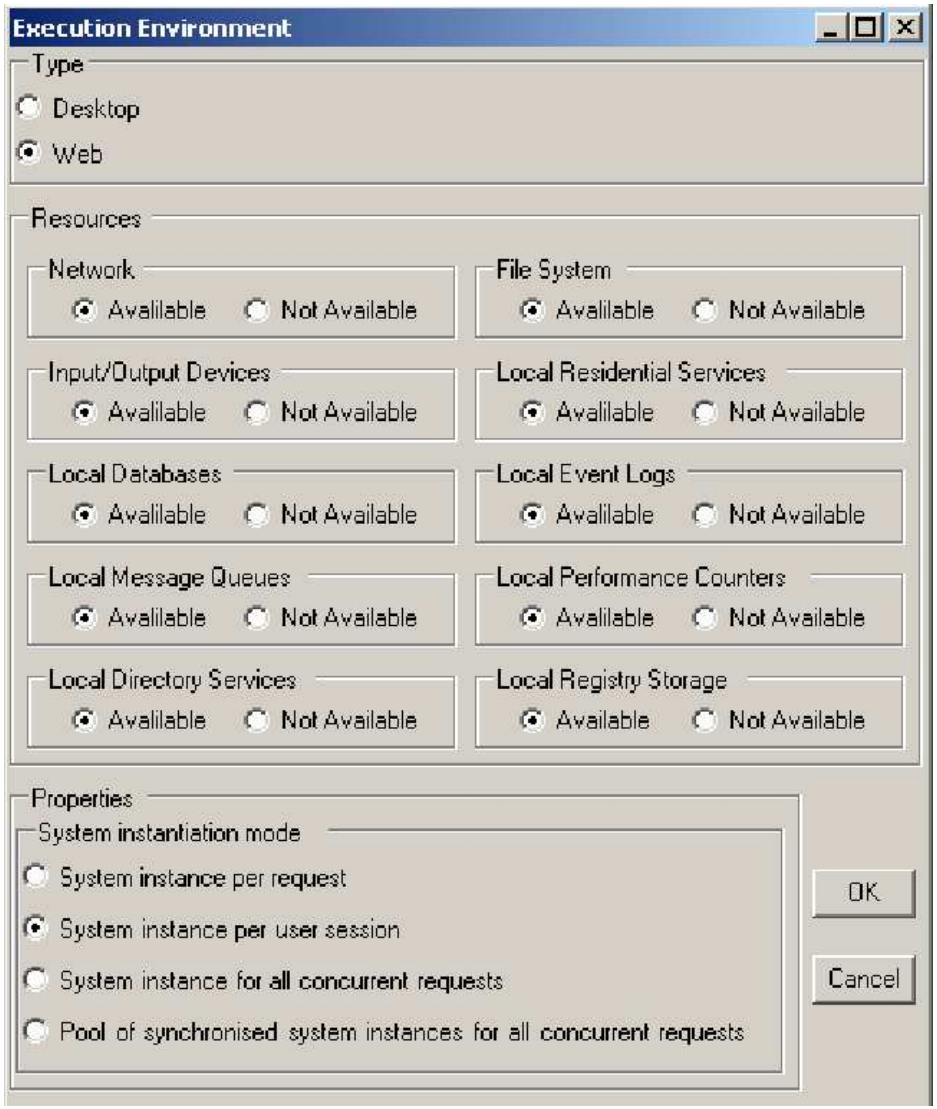
	Desktop Environment	Web Environment
System instantiation	Once for all requests	Depends on technology
State retention issues	No	Yes
Main thread affinity	Yes	No
Exposure to multiple threads	No	Yes

Conclusion

- A component is meant to be generic
- However, it may not run in every execution environment
- A component's suitability for desktop and web execution environments can be checked
  - statically at deployment time before runtime

    (using our Deployment Contracts Analyser Tool)

Defining an execution environment:



Analysing deployment contracts: