

A Study of Execution Environments for Software Components

Kung-Kiu Lau and Vladyslav Ukis

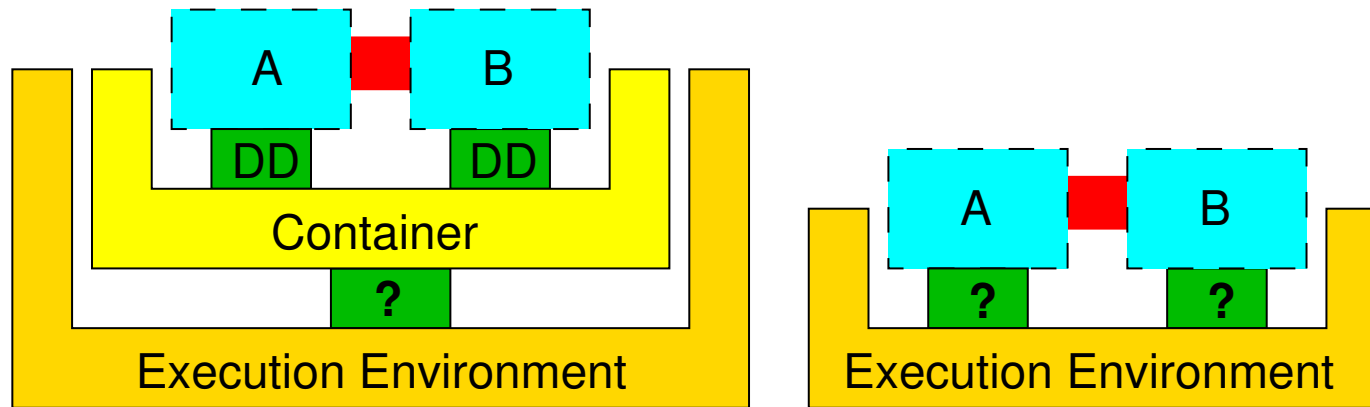
**School of Computer Science
The University of Manchester, UK**

{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

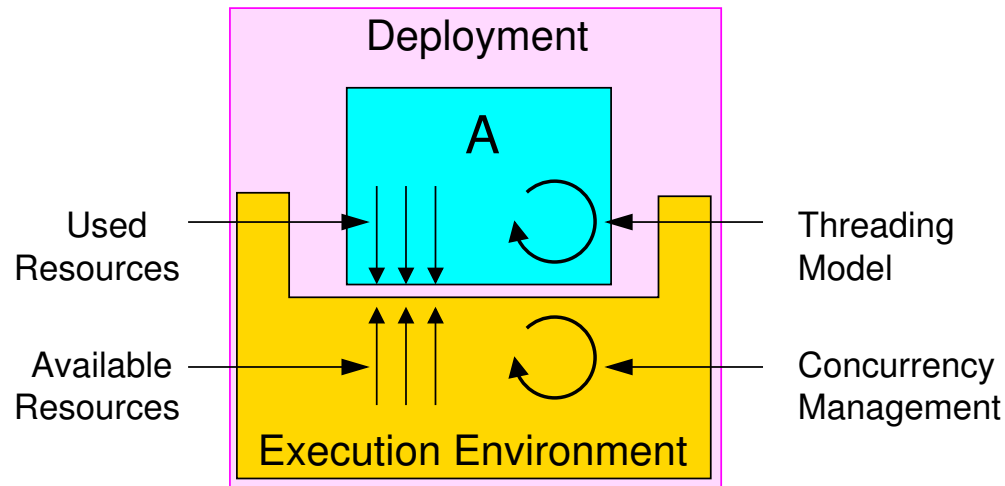
- **J2EE** (OSGi) — EJB (desktop), JSP (web server)
- **.NET** — default (desktop), ASP.NET (web server)

Our analysis of execution environments is based on these platforms.

Component Deployment

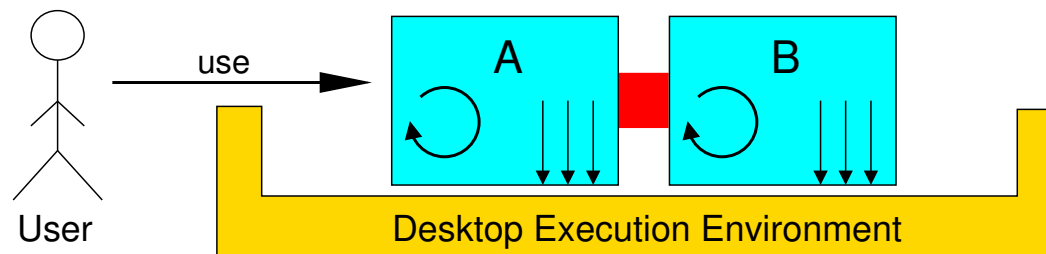
- **Question:** Given a binary component, how can we determine which execution environments it can be **deployed** into?
- We will analyse desktop and web environments as used in J2EE and .NET platforms to give the answer

Properties of Interest in an Execution Environment



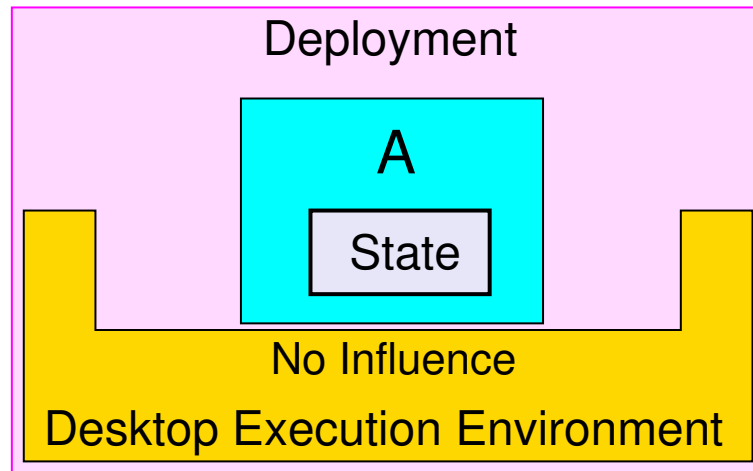
- In an execution environment we need to know
 - Available resources
 - Concurrency management
 - Transient state management
- With this knowledge, component's compatibility with the execution environment can be checked

Desktop Execution Environment



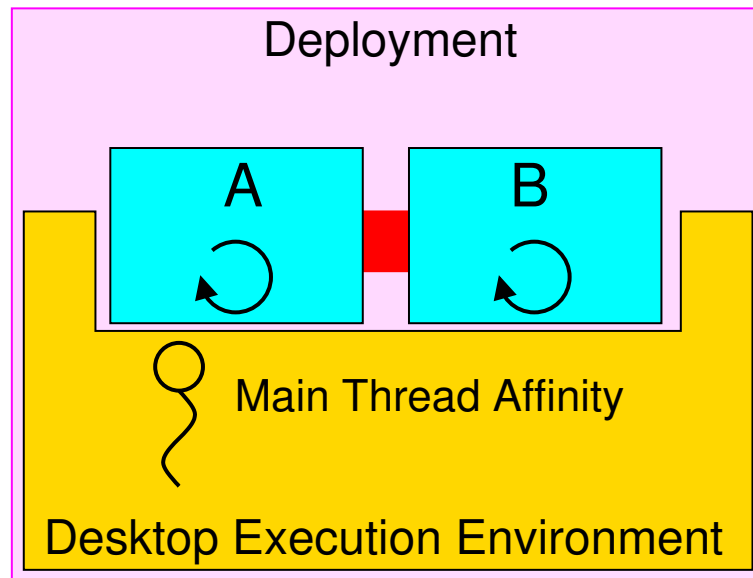
- Example systems
 - Adobe Acrobat Reader
 - Ghost Viewer
 - Unix commands, e.g. ls or ps
- A **single** user interacts with the system

Desktop Environment: Transient State Management



- Desktop environment does **not** influence component state

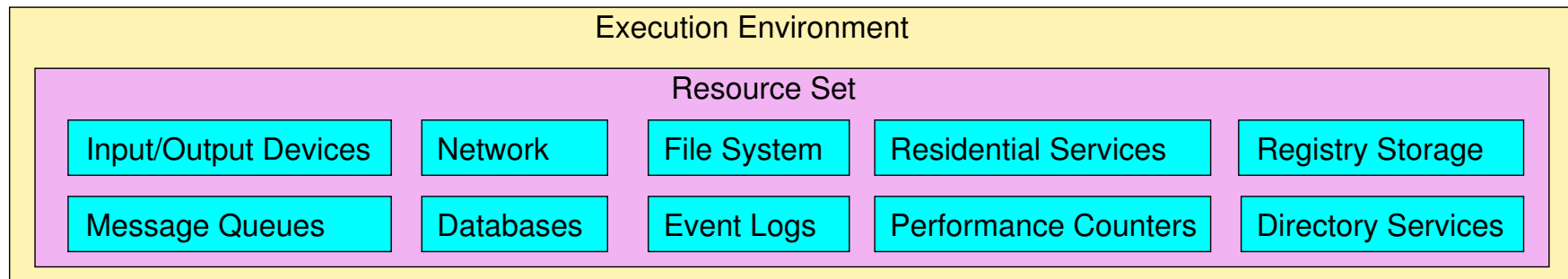
Desktop Environment: Concurrency Management



- 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

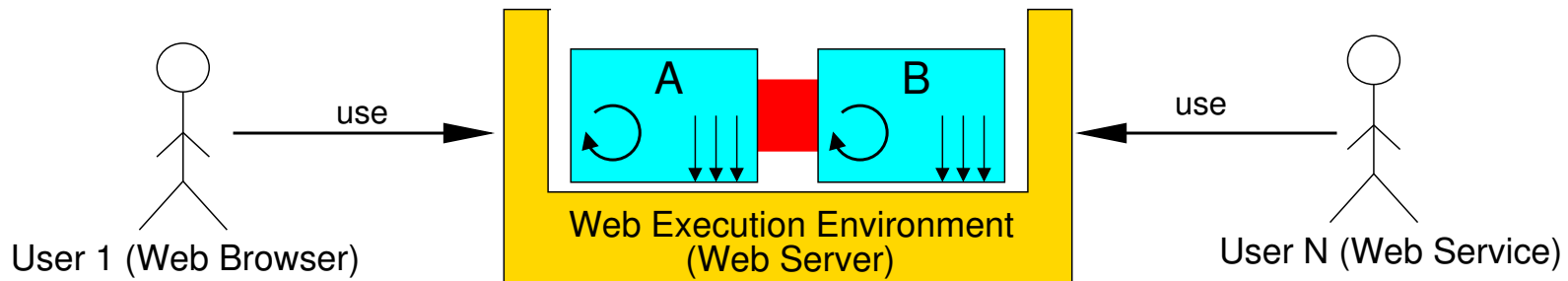
We have defined the following **resource set**:



- 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

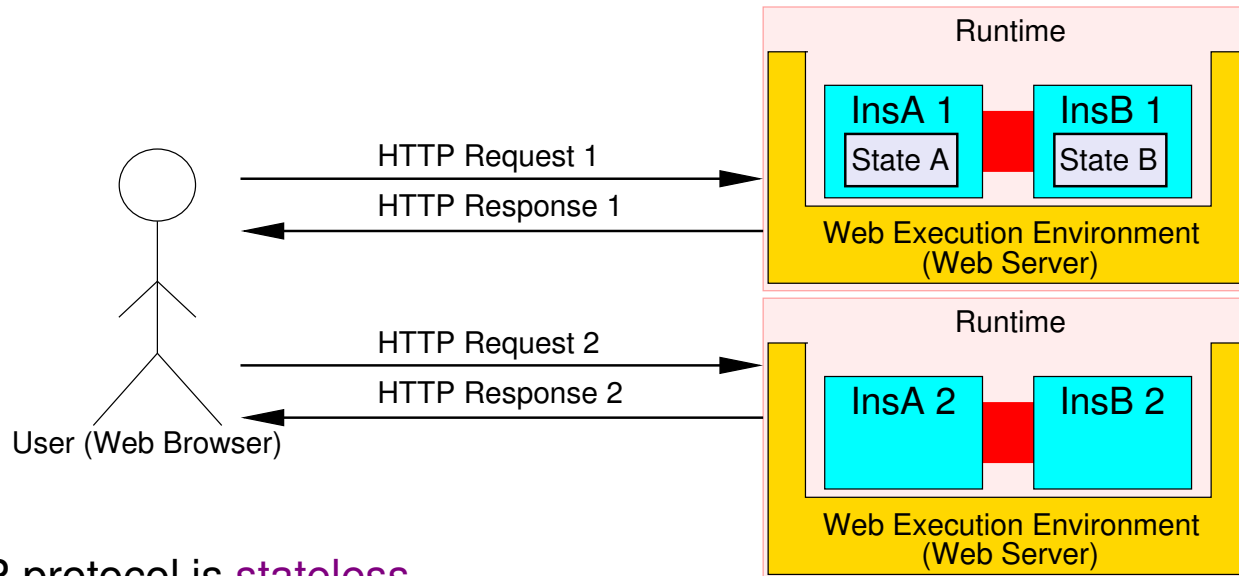


- Example systems

- Amazon
- ebay
- Google

- There are **many** users, possibly **simultaneously**, interacting with the system

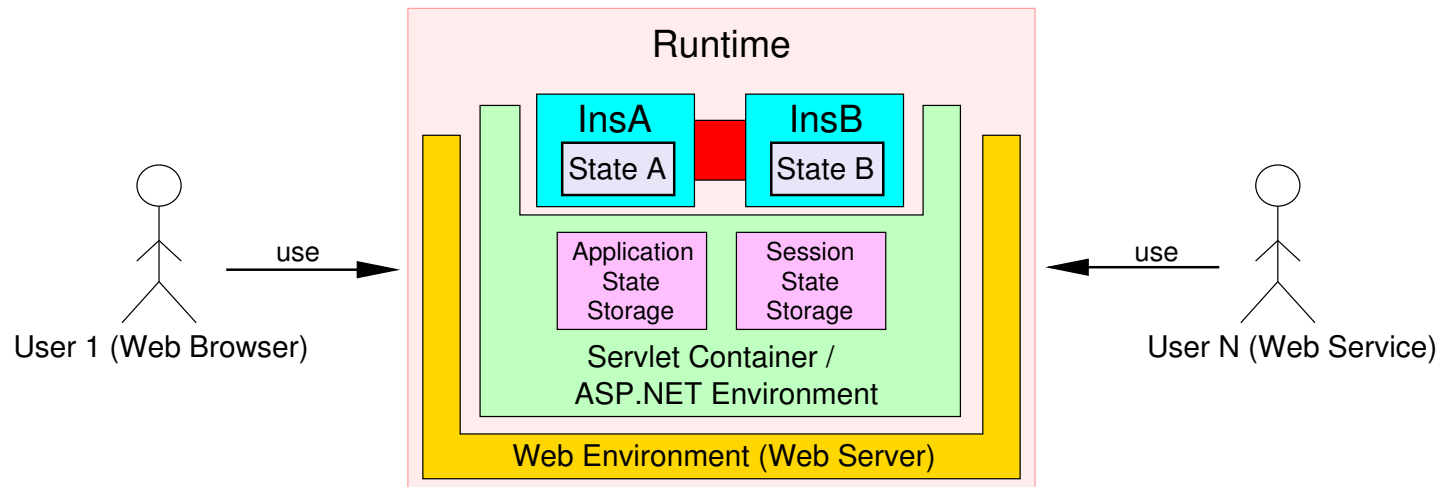
Web Environment: Transient State Management



- 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)

Conclusion (Continued)

Defining an execution environment:

The screenshot shows a dialog box titled "Execution Environment" with the following configuration:

- Type:** Web
- Resources:**
 - Network:** Available
 - File System:** Available
 - Input/Output Devices:** Available
 - Local Residential Services:** Available
 - Local Databases:** Available
 - Local Event Logs:** Available
 - Local Message Queues:** Available
 - Local Performance Counters:** Available
 - Local Directory Services:** Available
 - Local Registry Storage:** Available
- Properties:**
 - System instance per request
 - System instance per user session
 - System instance for all concurrent requests
 - Pool of synchronised system instances for all concurrent requests

Buttons: OK, Cancel

Conclusion (Continued)

Analysing deployment contracts:

The screenshot displays the Deployment Contracts Analyser application interface. The main window shows a component diagram with components labeled 'Component A' through 'Component 110'. A 'Components' list on the right includes 'Component1', 'Component10', 'Component10a', and 'Component100'. The 'Execution Environment' dialog is open, showing settings for 'Type' (Web), 'Resources' (Network, File System, Input/Output Devices, Local Residential Services), and 'Available/Not Available' options. The 'Deployment Contracts Analysis' dialog is also open, displaying a search bar and a list of analysis results for components 106, 107, 108, and 109. The analysis results for Component 107 and 108 indicate network requirements and hints.

Deployment Contracts Analyser - Project: ProjectAsyncD...

File Tools Analyses Help

Components Connectors

Add Component...

Component1
Component10
Component10a
Component100

Execution Environment

Type
 Desktop
 Web

Resources

Network
 Available Not Available

File System
 Available Not Available

Input/Output Devices
 Available Not Available

Local Residential Services
 Available Not Available

Deployment Contracts Analysis

Zoom Text: [Slider]

Search Analysis Results: [Search Box] Find All Find First

Check Each Component's Deployment Contract against Execution Environment's Resources:

Component 'Component106':
Summary: Component 'Component106' - OK

Component 'Component107':
Method-level attribute 'UsedFile' on method 'Method1': [UsedFile(FullName=abc.txt, Location=Remote, UsageNecessity=Mandatory, UsageMode=Read, Delete, Existence=Checked)]
Network required. Network is available in the assembly's target execution environment. OK
Remote File System required. HINT
Summary: Component 'Component107' - ERRORS: 0, WARNINGS: 0, HINTS: 1

Component 'Component108':
Method-level attribute 'UsedFile' on method 'Method1': [UsedFile(FullName=abc.txt, Location=Remote, UsageNecessity=Mandatory, UsageMode=Read, Write, Existence=Unchecked)]
Network required. Network is available in the assembly's target execution environment. OK
Remote File System required. HINT
Summary: Component 'Component108' - ERRORS: 0, WARNINGS: 0, HINTS: 1

Component 'Component109':
Method-level attribute 'UsedFile' on method 'Method1': [UsedFile(FullName=abc.txt, Location=Remote, UsageNecessity=Mandatory, UsageMode=Write, Create, Existence=Unchecked)]
Network required. Network is available in the assembly's target execution environment. OK

OK
Cancel