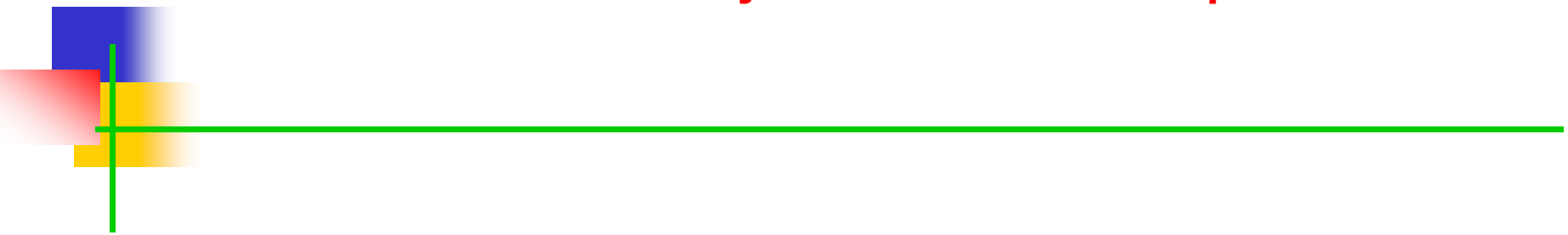
Distributed Objects and Components



Chapter 8 from the TextBook

Distributed Objects and Components

- Distributed objects and components are two of the most important styles of middleware in use today.
- Distributed object middleware: A range of middleware solutions based on distributed objects include Java RMI and CORBA.
- Java RMI vs. CORBA?

Issues with Object-Oriented Middleware

- Implicit dependencies
  - A distributed object offers a ***contract*** to the outside world in terms of the interface (or interfaces) it offers to the distributed environment.
  - Problem?
  - Requirement: To specify interfaces and dependencies
- Interaction with the middleware
  - Despite the transparency concept, many calls are middleware-related (e.g., calls to the RMI registry)
  - Requirement: To simplify the programming of distributed applications (separation of concerns)

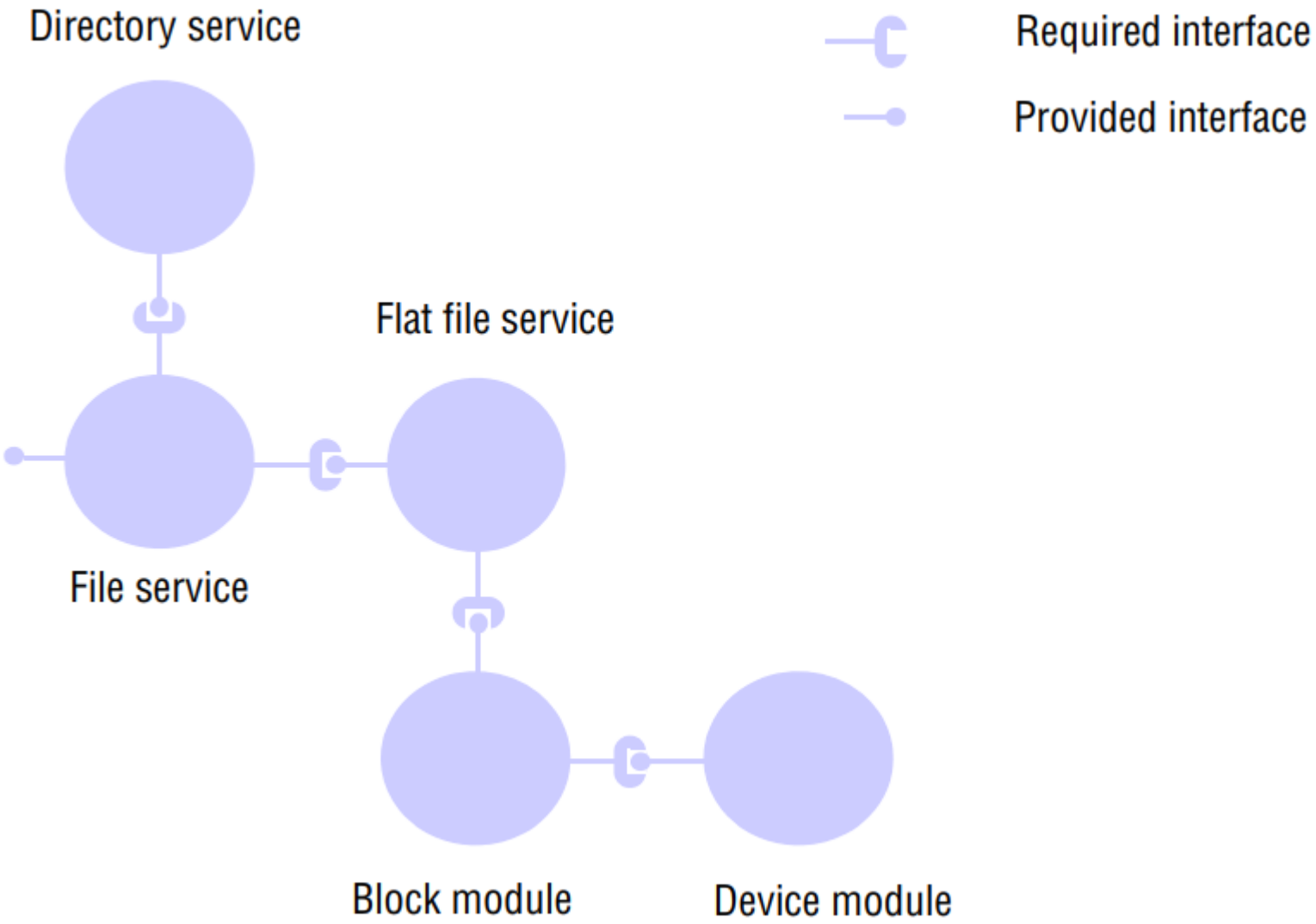
Issues with Object-Oriented Middleware

- Lack of separation of distribution concerns:
  - Problem?
  - Requirement: To extend separation of concerns to distributed system's services
- No support for deployment
  - Objects must be deployed manually on each machine.
  - Object-based middleware provides little or no support for complex/varied configurations of objects.
- Those requirements have led to the emergence of component-based approaches to distributed systems development alongside the style of middleware referred to as application servers.

Using Components

- Components?
- A component is specified in terms of a **contract (what does it include?)**
- Interfaces may be of different styles. In particular, many component-based approaches offer two styles of interface:
  - Interfaces supporting remote method invocation, as in CORBA and Java RMI
  - Interfaces supporting distributed events (as discussed in Chapter 6)

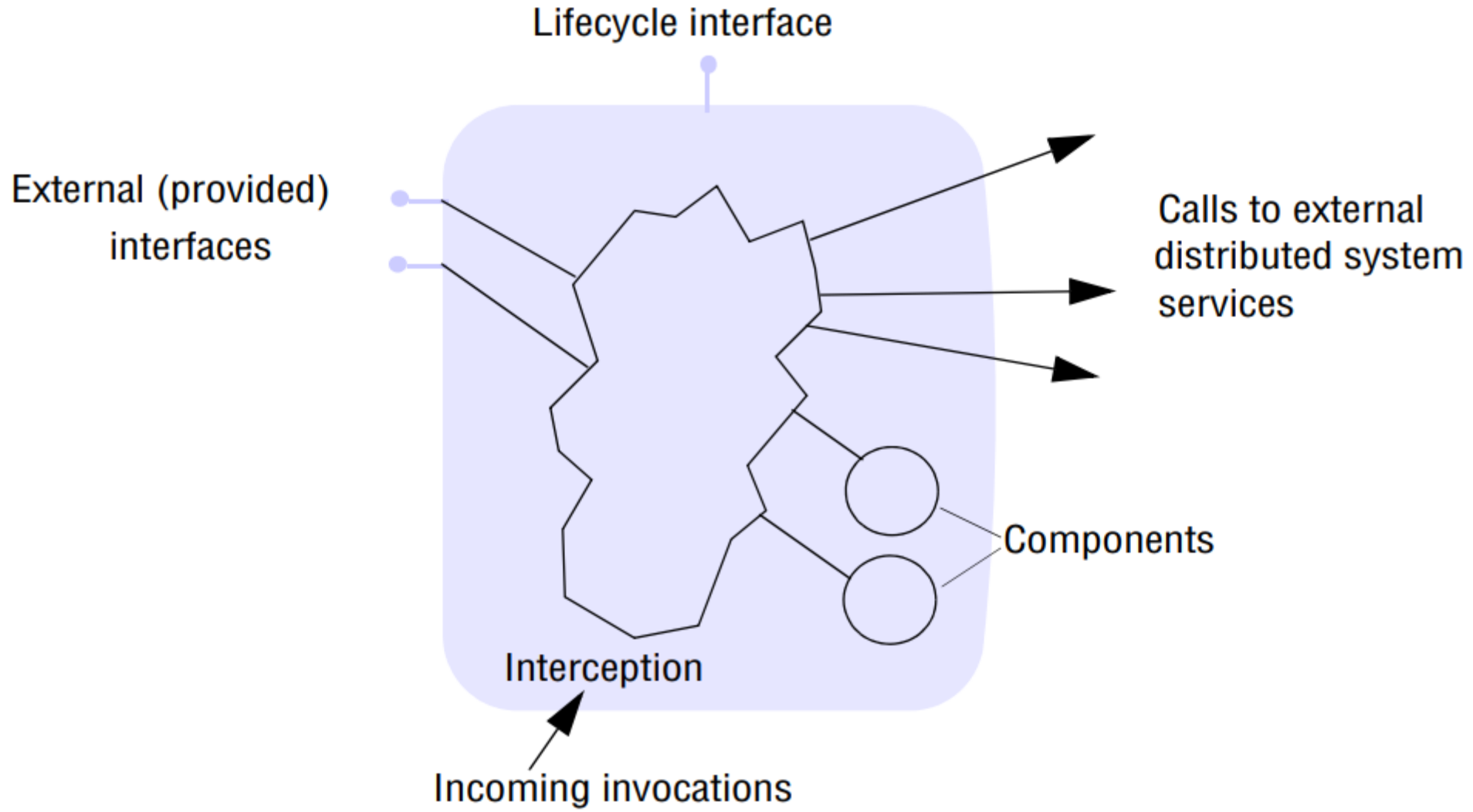
Using Components



Containers

- Containers support a common pattern often encountered in distributed applications, which consists of:
  - a front-end (perhaps web-based) client;
  - a container holding one or more components that implement the application or business logic;
  - system services that manage the associated data in persistent storage
    Three tier architecture?
- Different roles?

Containers


Lifecycle interface
External (provided) interfaces
Calls to external distributed system services
Components
Interception
Incoming invocations

Application Servers?

Middleware supporting the container pattern and the separation of concerns implied by this pattern is known as an ***application server***

This style of distributed programming is in widespread use in industry today: – range of application servers:

Technology	*Developed by*	*Further details*
WebSphere Application Server	IBM	www.ibm.com
Enterprise JavaBeans	SUN	java.sun.com
Spring Framework	SpringSource (a division of VMware)	www.springsource.org
JBoss	JBoss Community	www.jboss.org
CORBA Component Model	OMG	[Wang *et al.* 2001JOnAS]
JOnAS	OW2 Consortium	jonas.ow2.org
GlassFish	SUN	glassfish.dev.java.net

References/Required Readings

- Chapter 8 from textbook: Coulouris, George F., Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. Fifth Edition