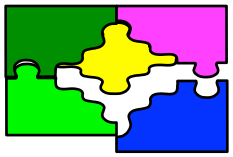
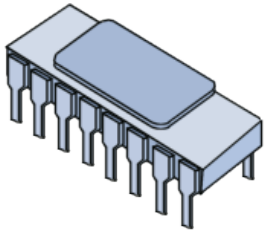
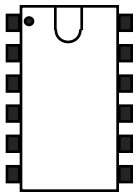


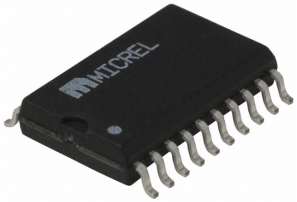


Architecture des Systèmes d'Information



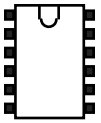
Component Based Development

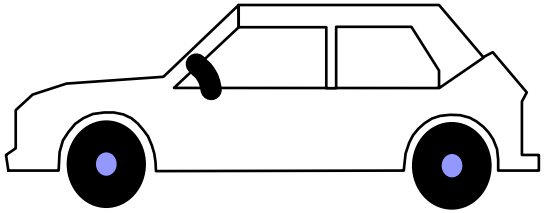
Traduction en cours



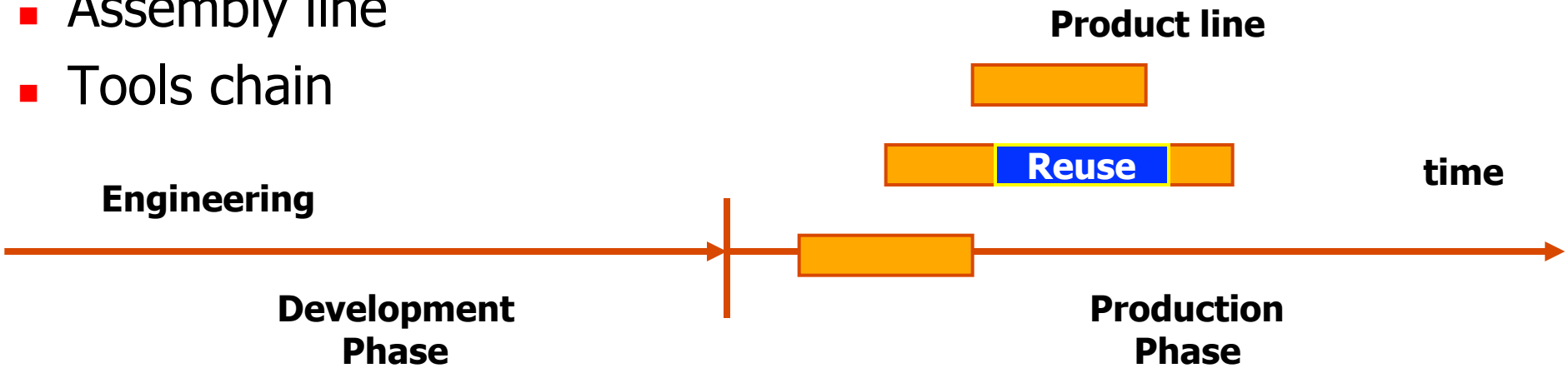
Component Based development for complex systems

- Develop large software systems like :
  - Hardware
  - Car

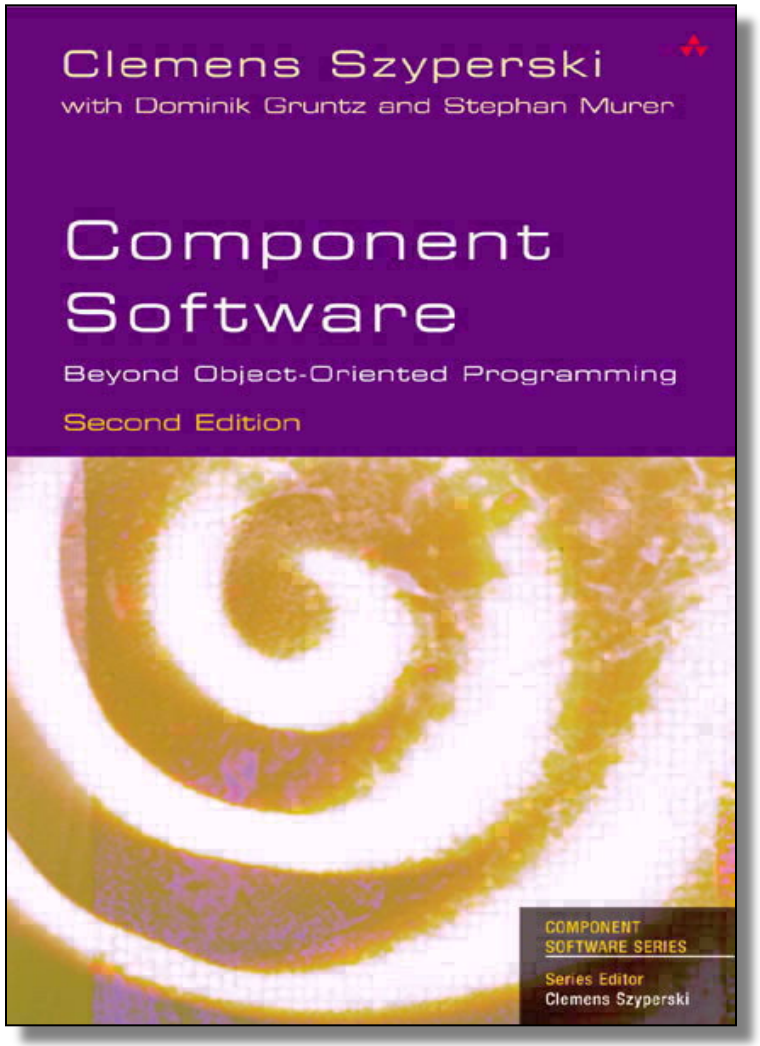




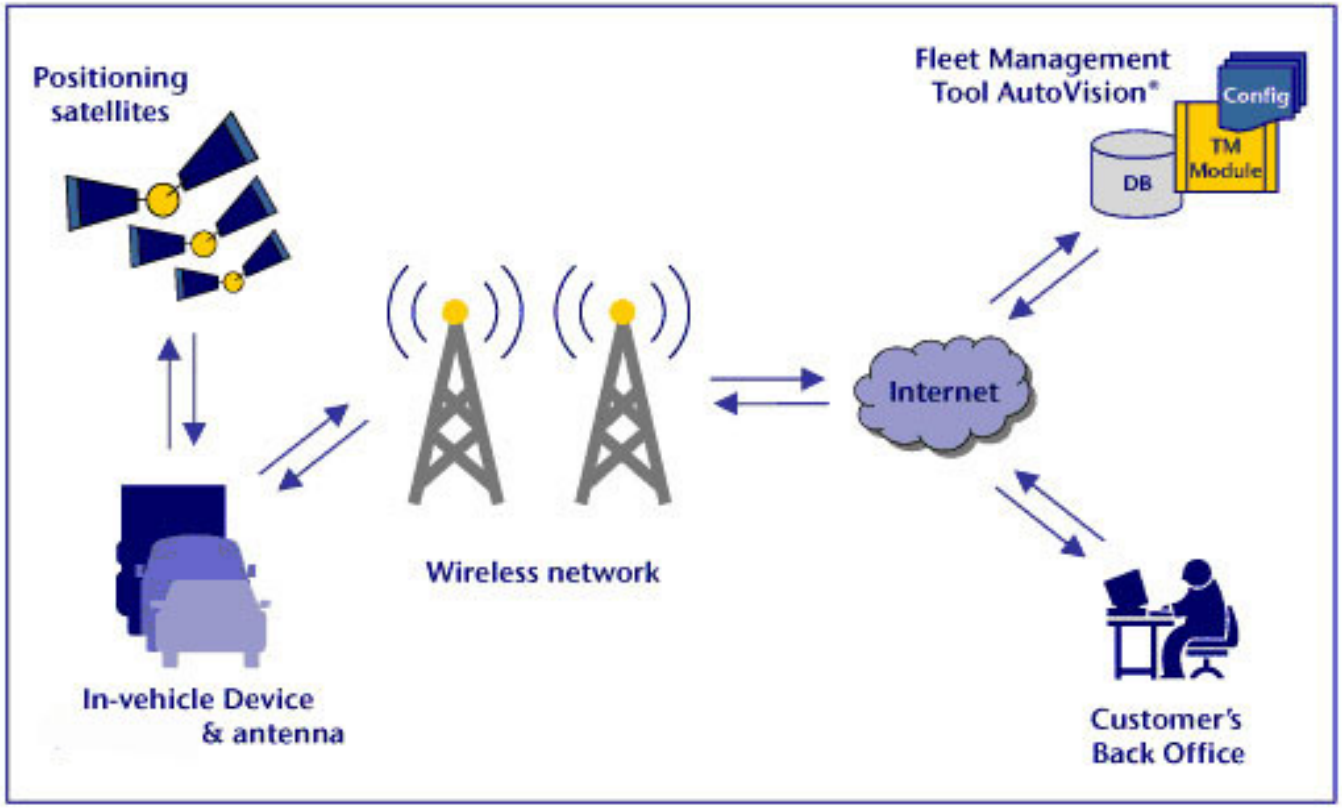
- Manufacturing rather than engineering
  - Product line
  - Assembly line
  - Tools chain



Clemens Szyperski

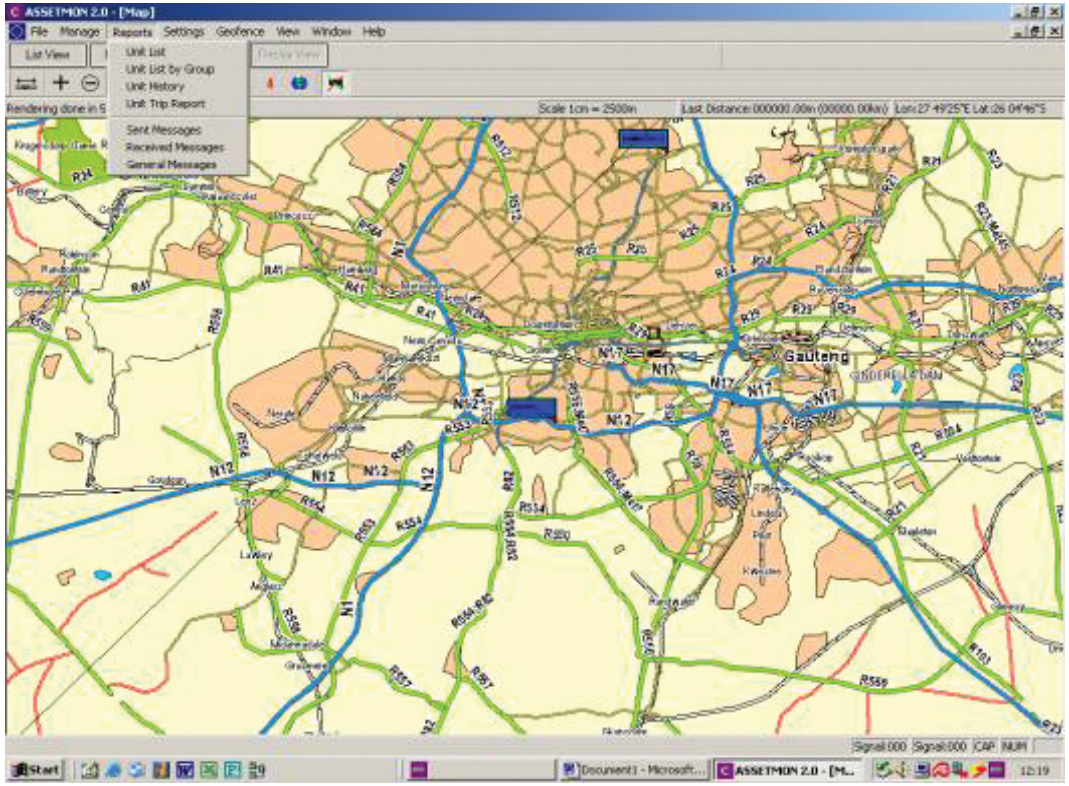


Exemple : Fleet Management System



http://www.alkantelecom.com/Fleet.html

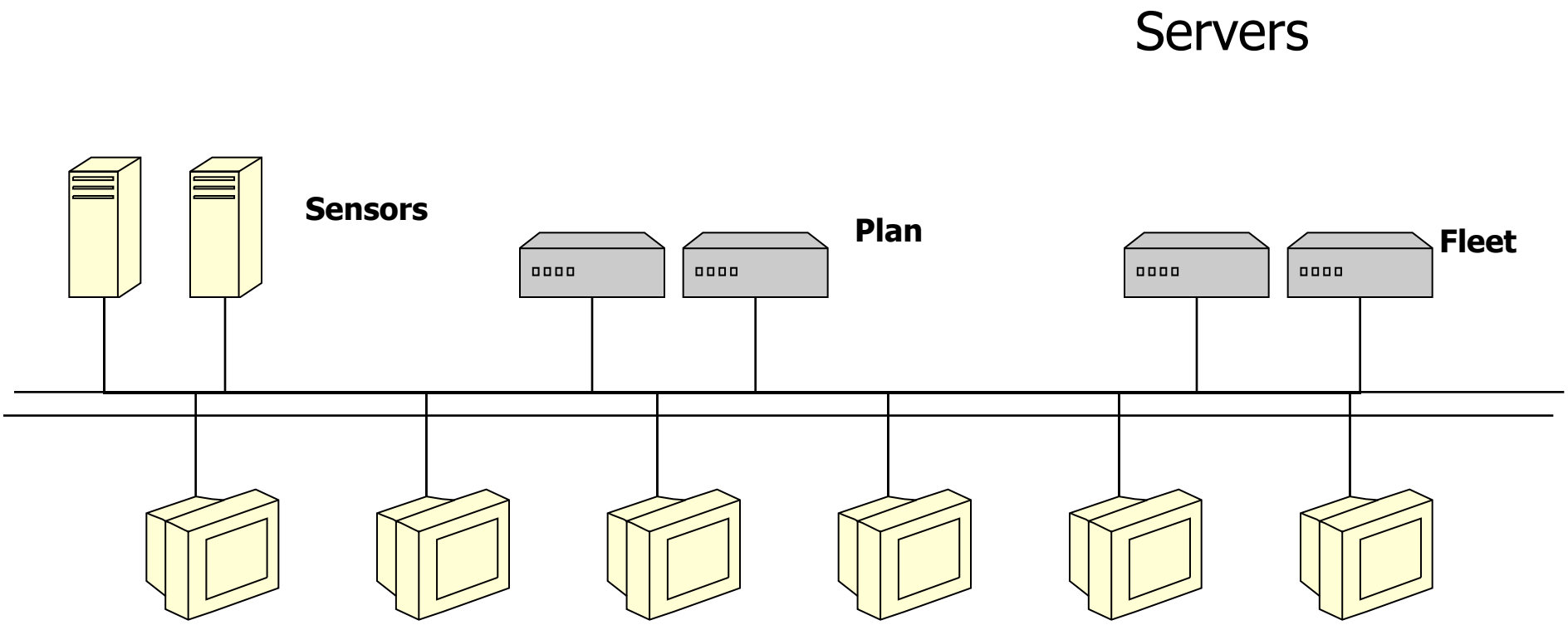
Exemple : Fleet Management System



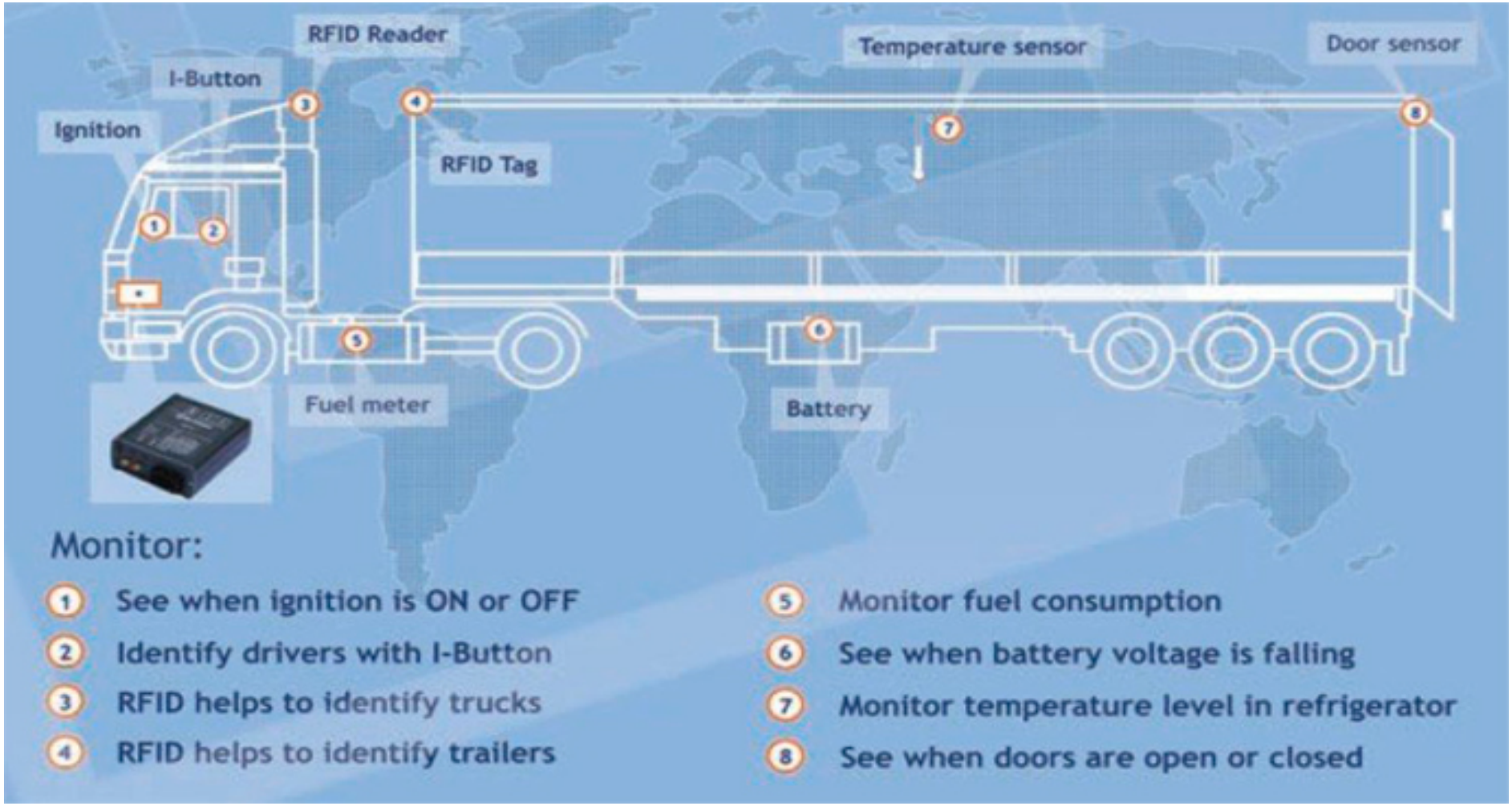
http://www.location.net.in/magazine/2006/may-jun/22_3.htm

http://www.atilog.com/GEFA-gestion-parc-auto.html

Fleet distributed management system

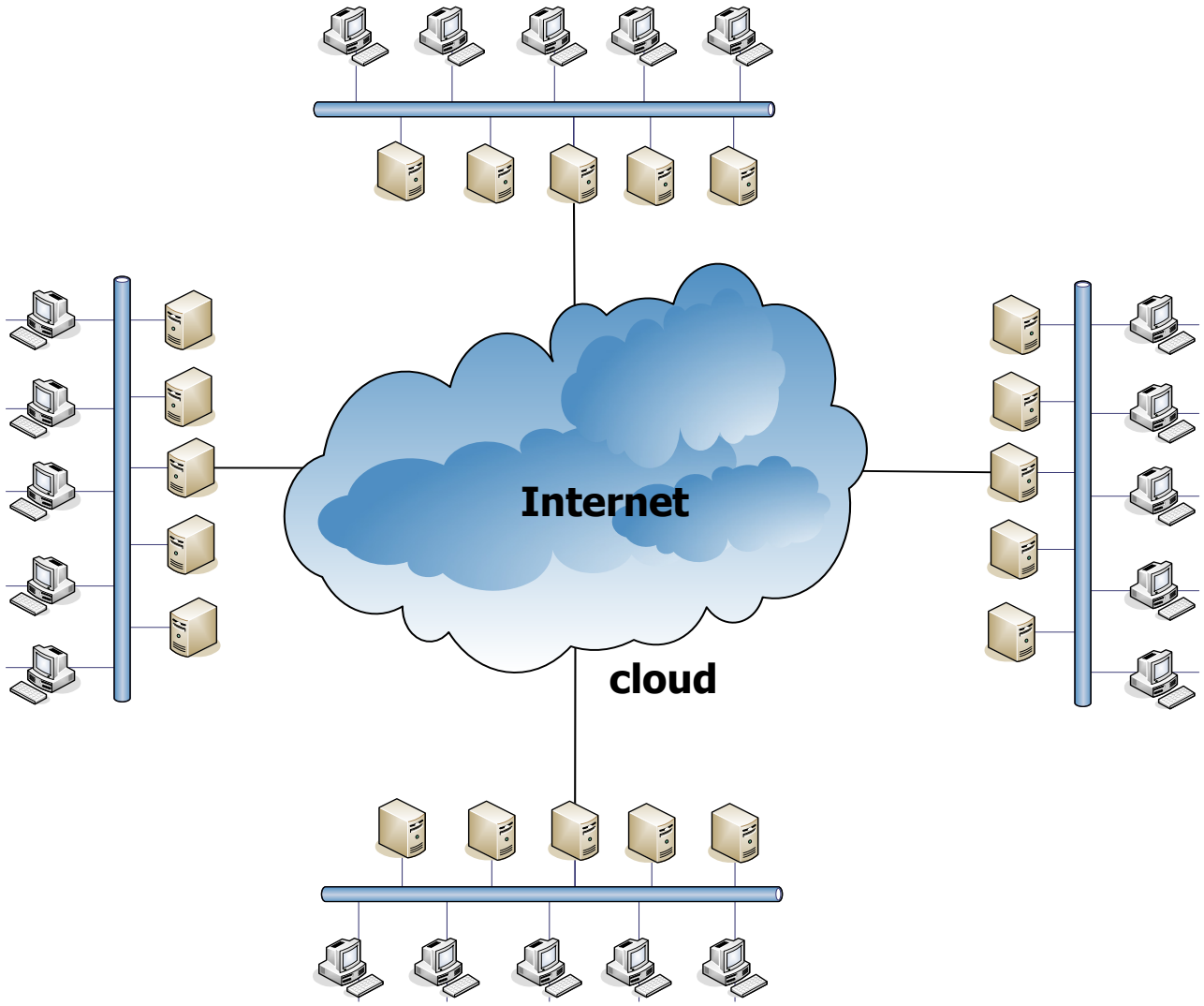


Sensors

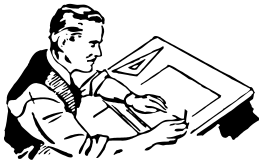


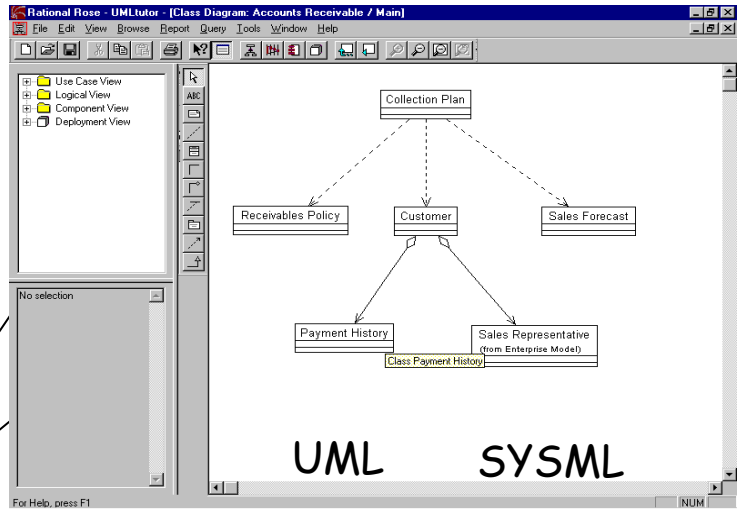
http://www.tglc.com/images/gps-flotte_3101.jpg

B to B : Business-to-business



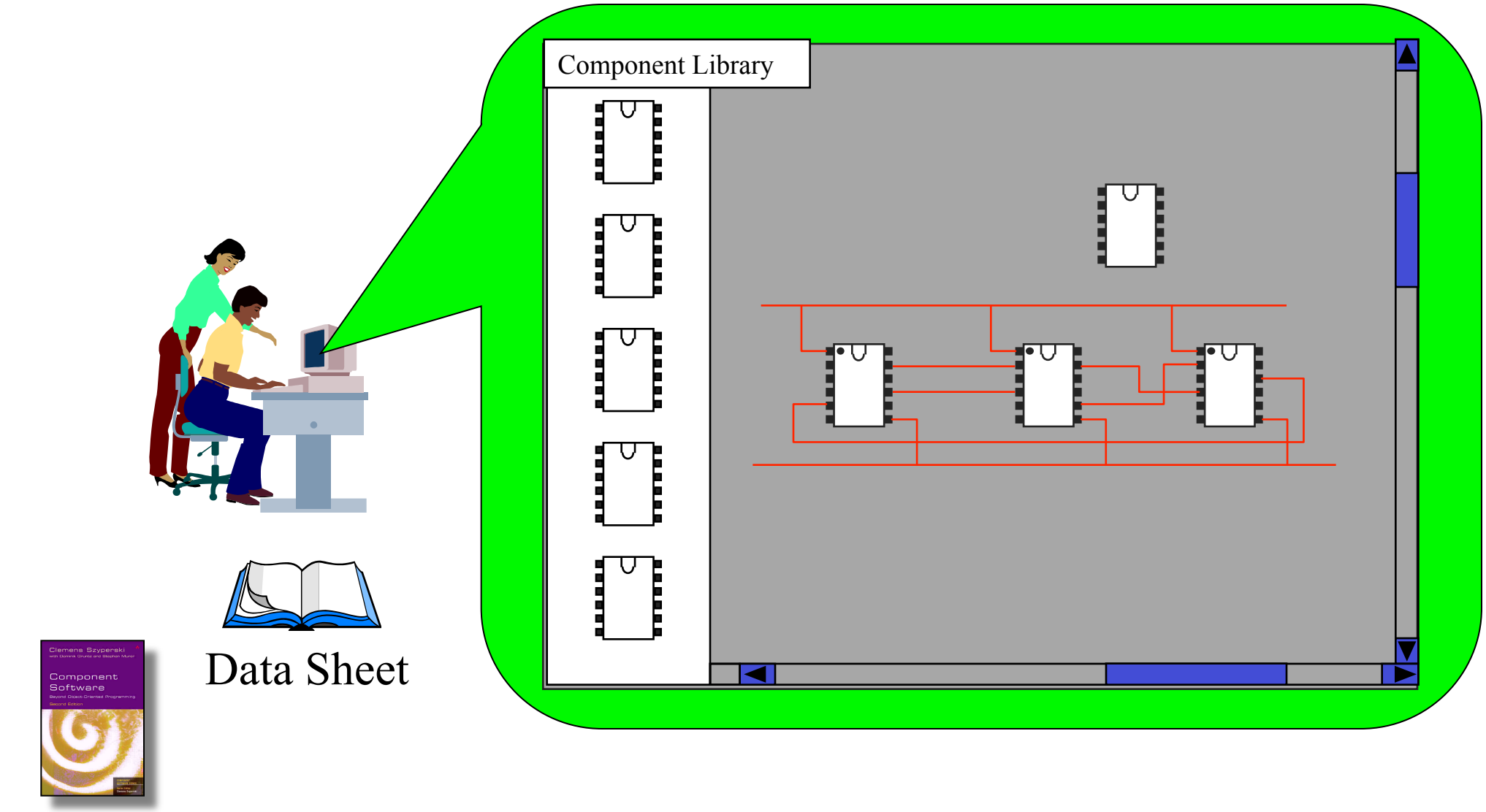
Tool chain : Computer Aided System Design



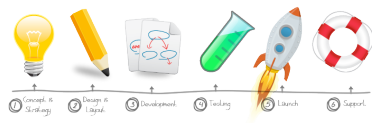


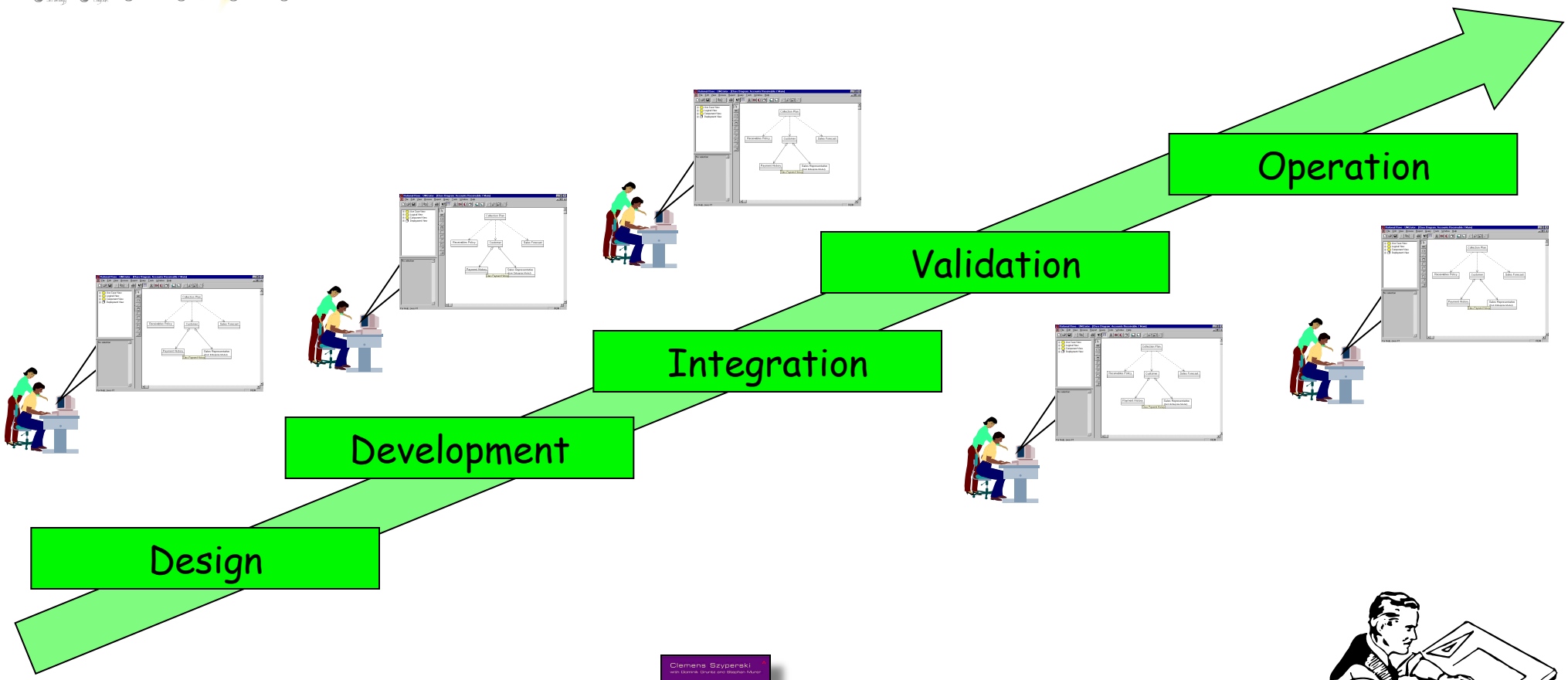
Visual Tools

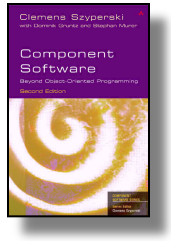
Hardware Components Analogy



Computer aided design for the whole process







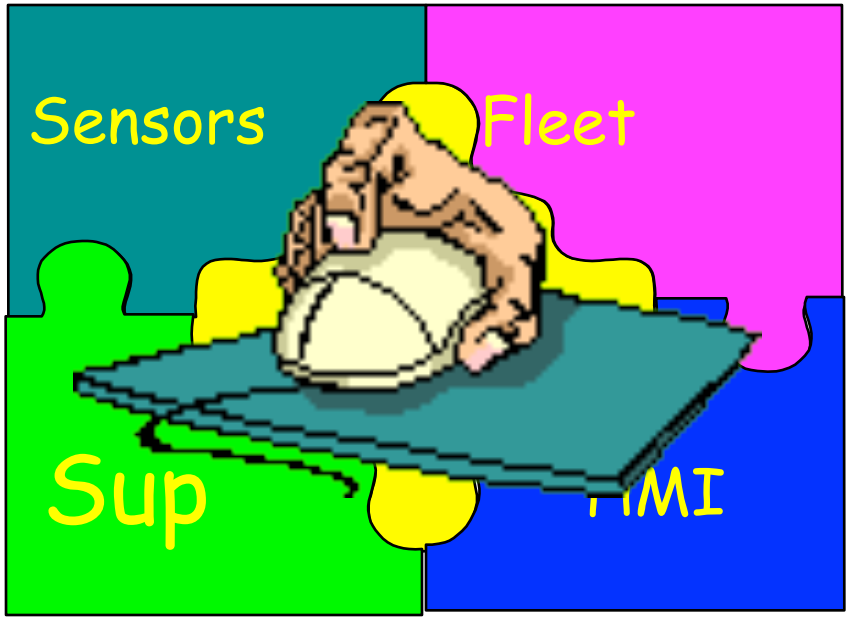
Actors

- Customers
- Interface designer
- Domain (Application) Developers
- Frameworks Developers
- Tool Providers
- Domain Experts
- Business Manager

Mouse Click System Development

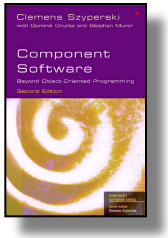
Sensors

Fleet

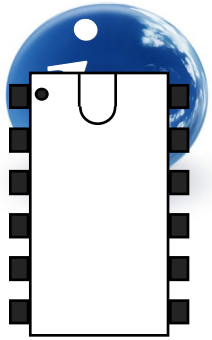


Sup

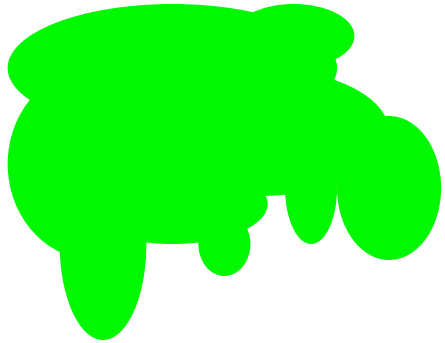
HMI



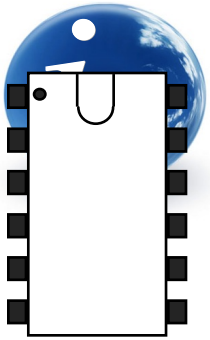
What We May Have :

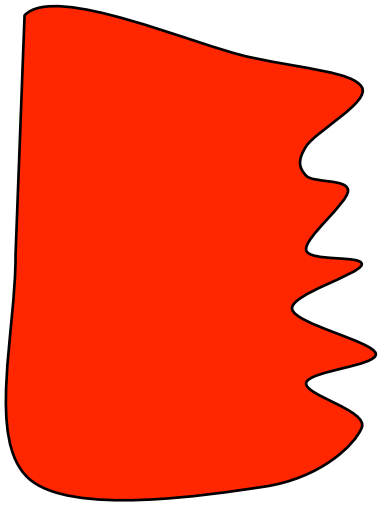


What We May Have :



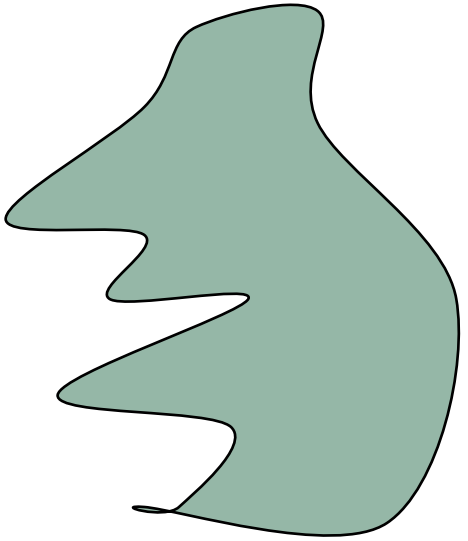
Service B





Service C

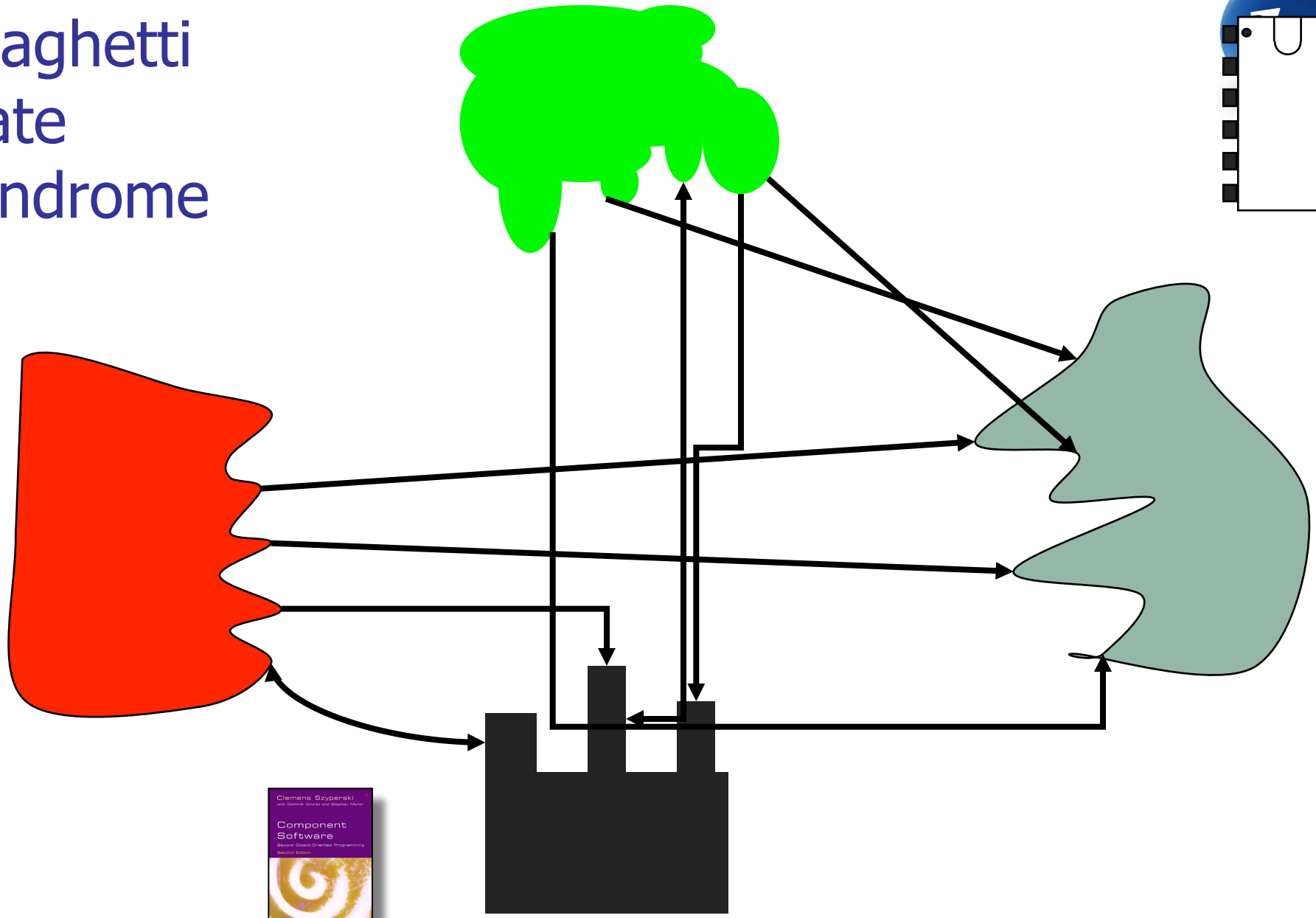
Service A



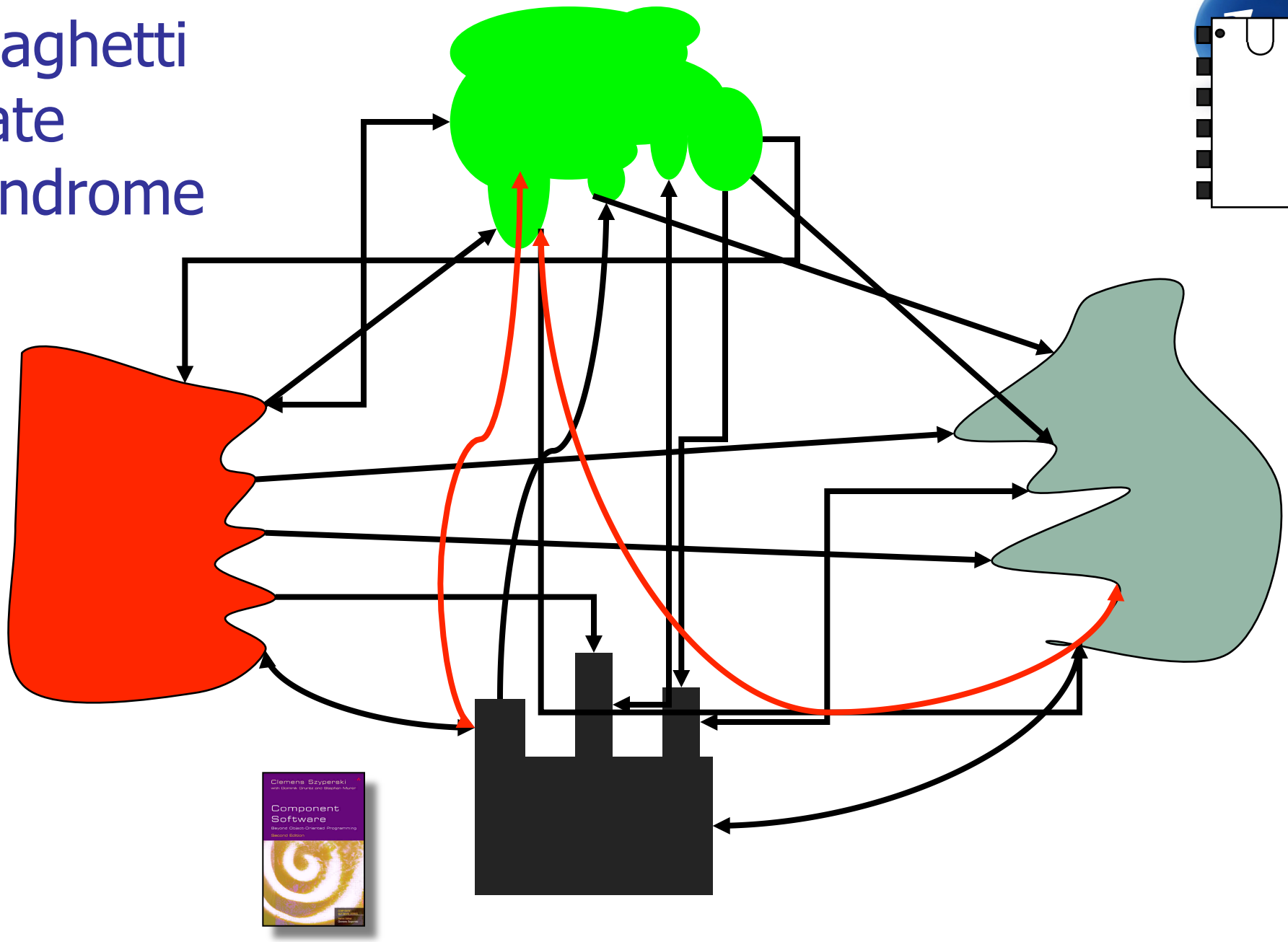
Service D

Service E

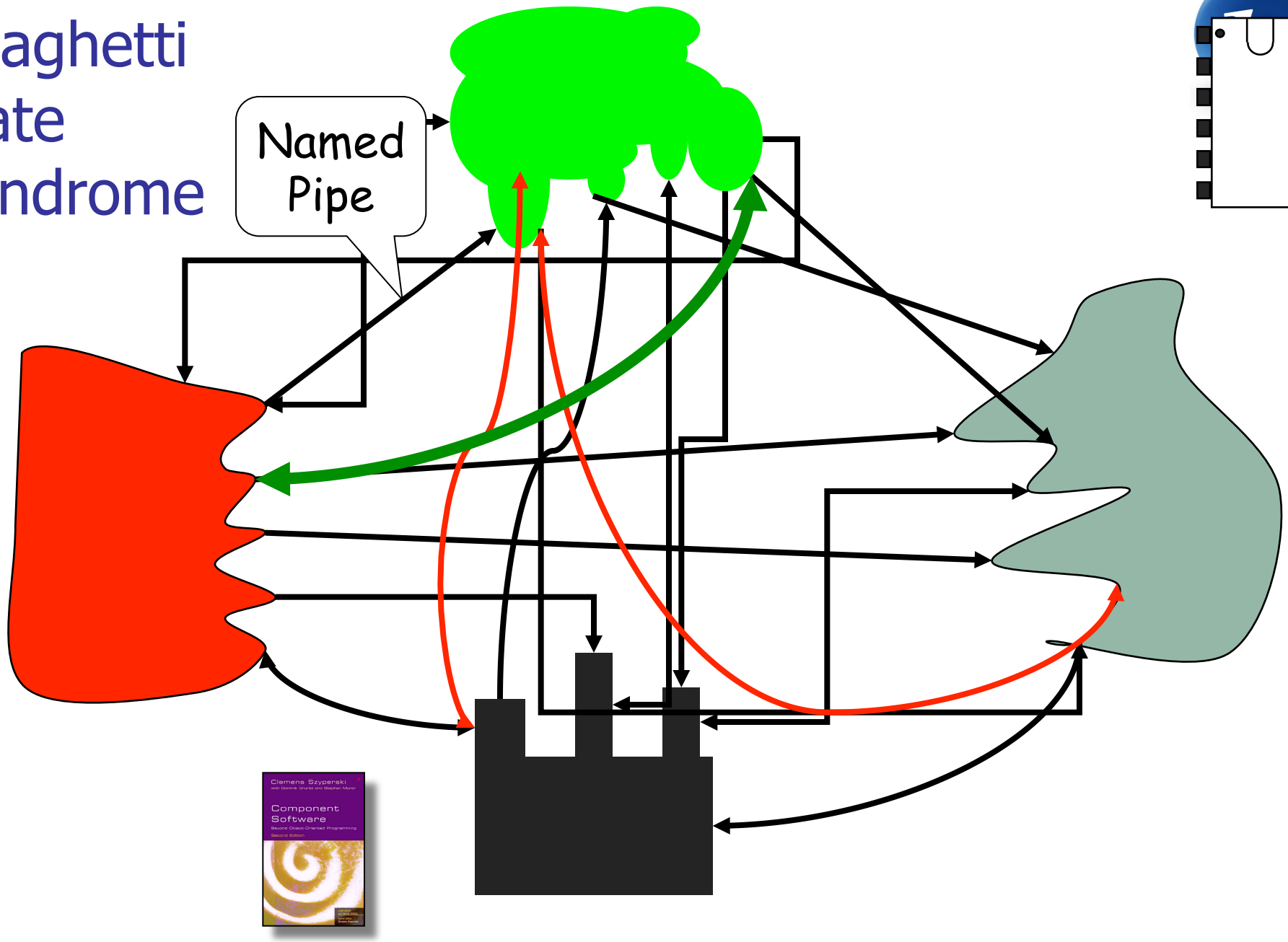
Spaghetti Plate Syndrome



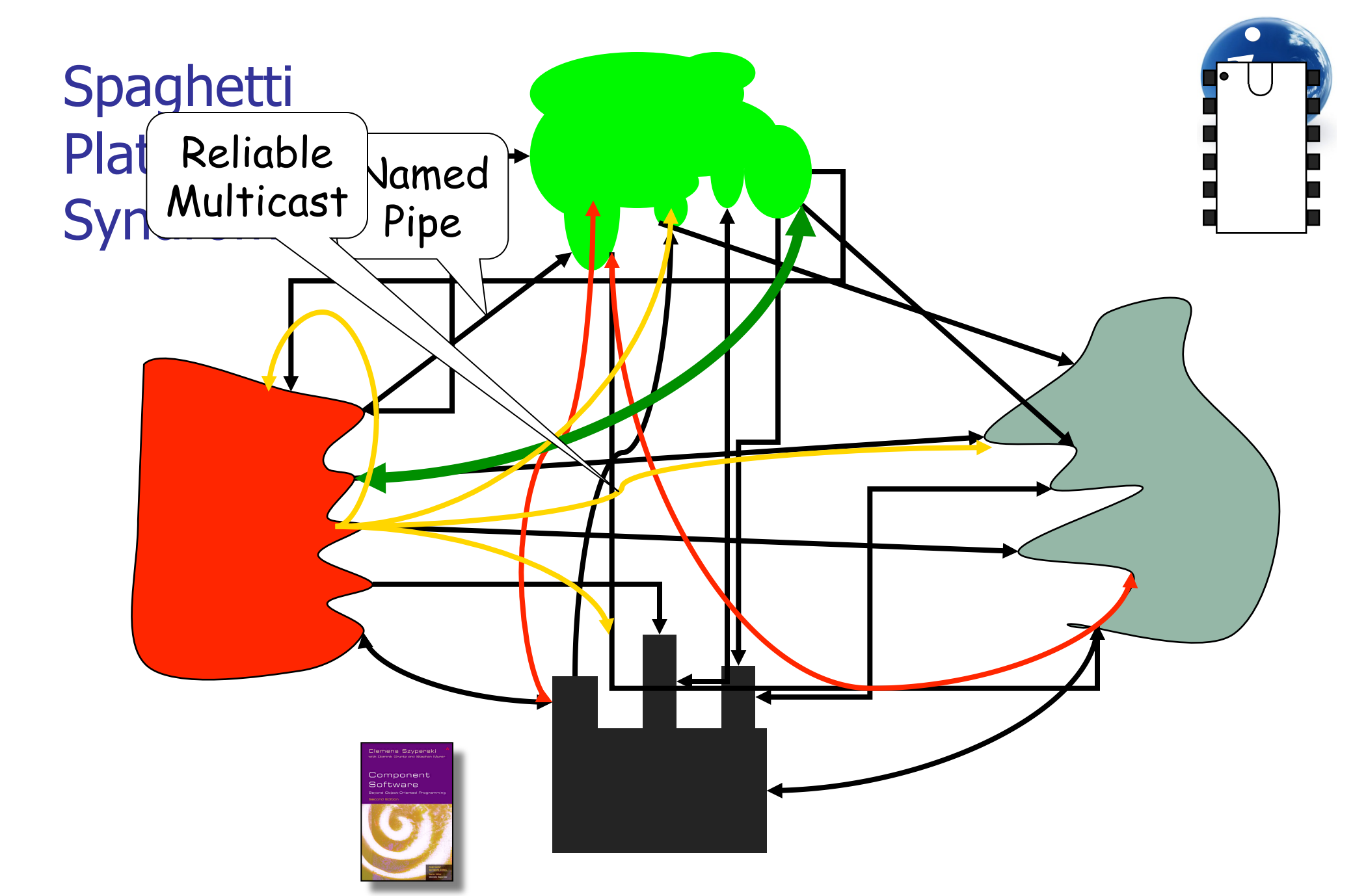
Spaghetti Plate Syndrome



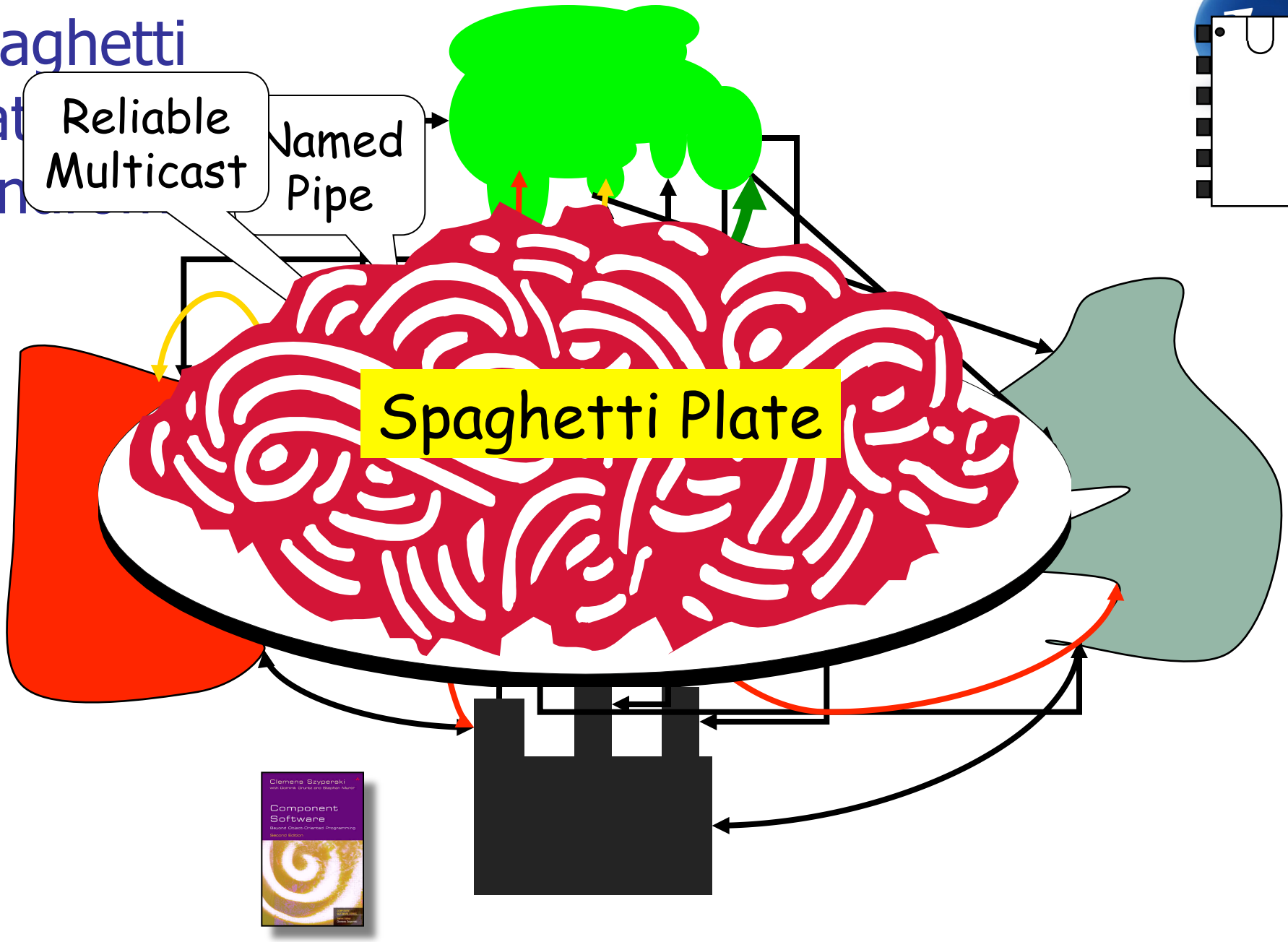
Spaghetti Plate Syndrome



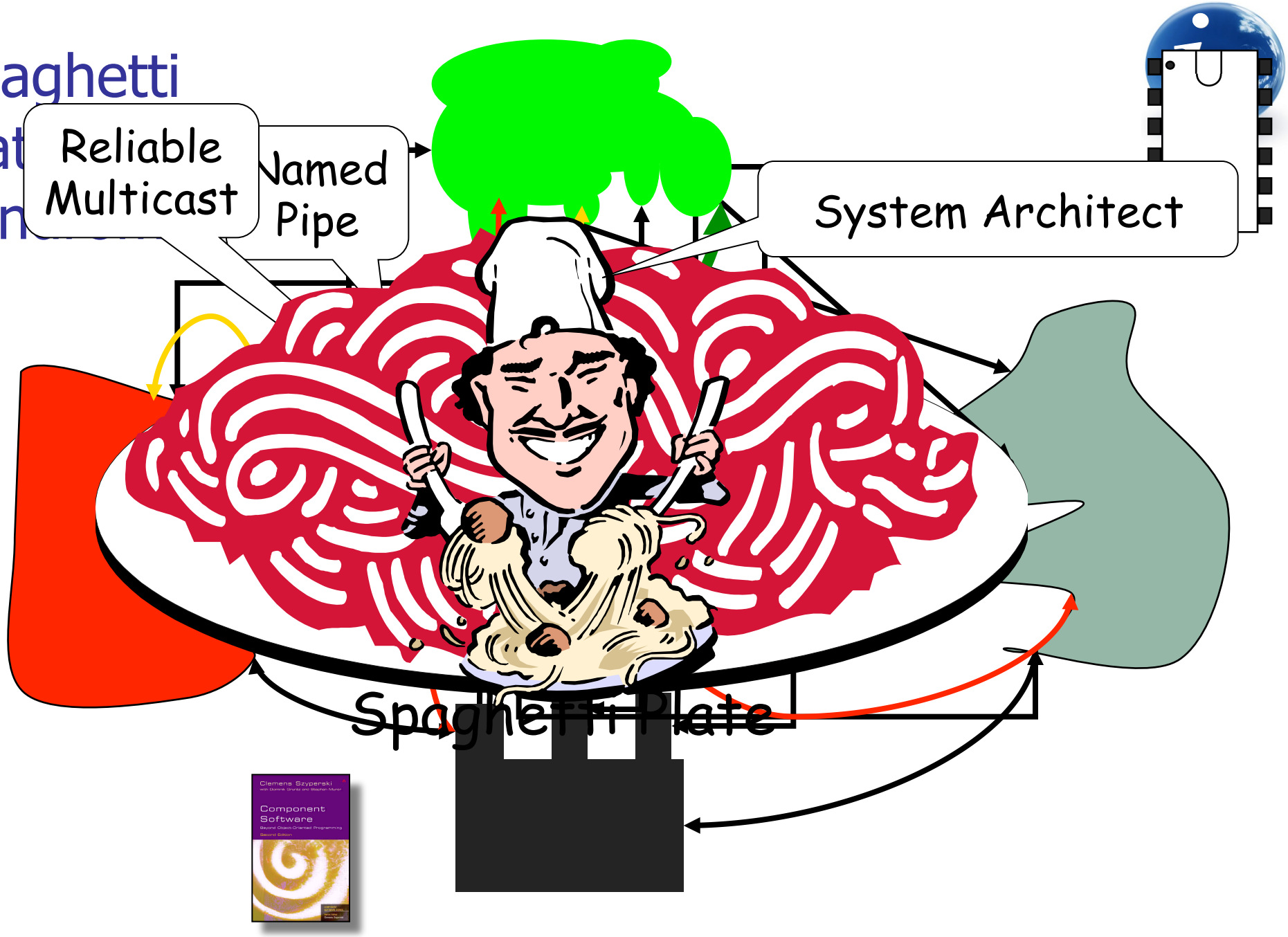
Spaghetti Plat… Syn…



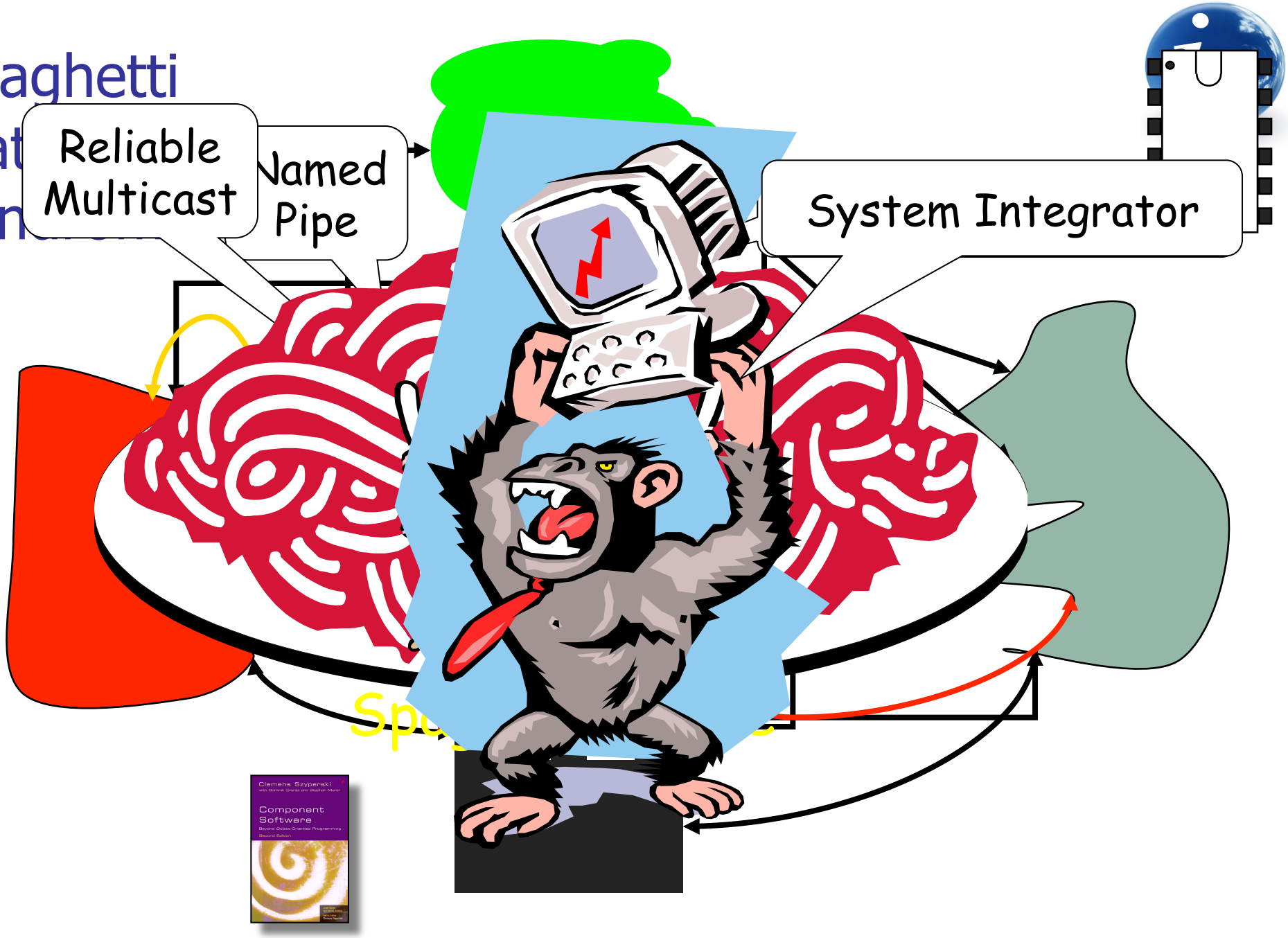
Spaghetti
Plat
Syn



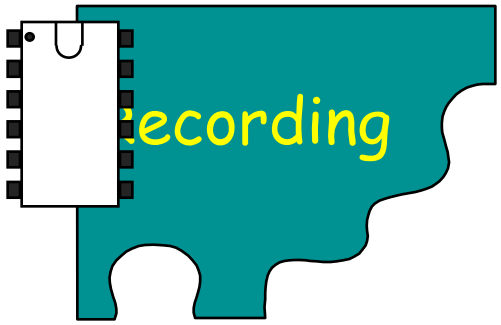
Spaghetti Plat… Syn…

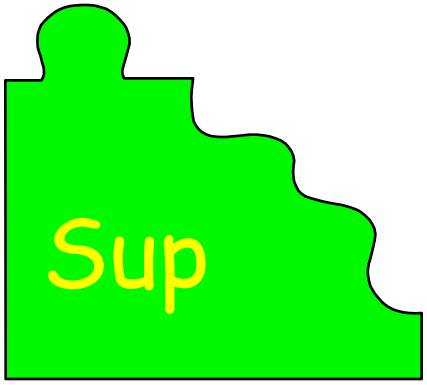


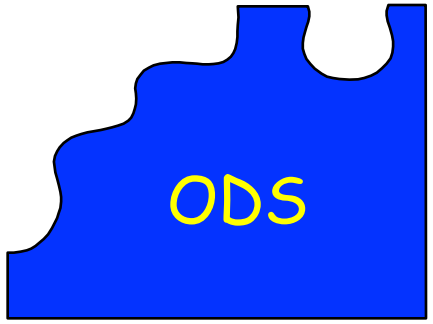
Spaghetti Plat… Syn…



What We Want !

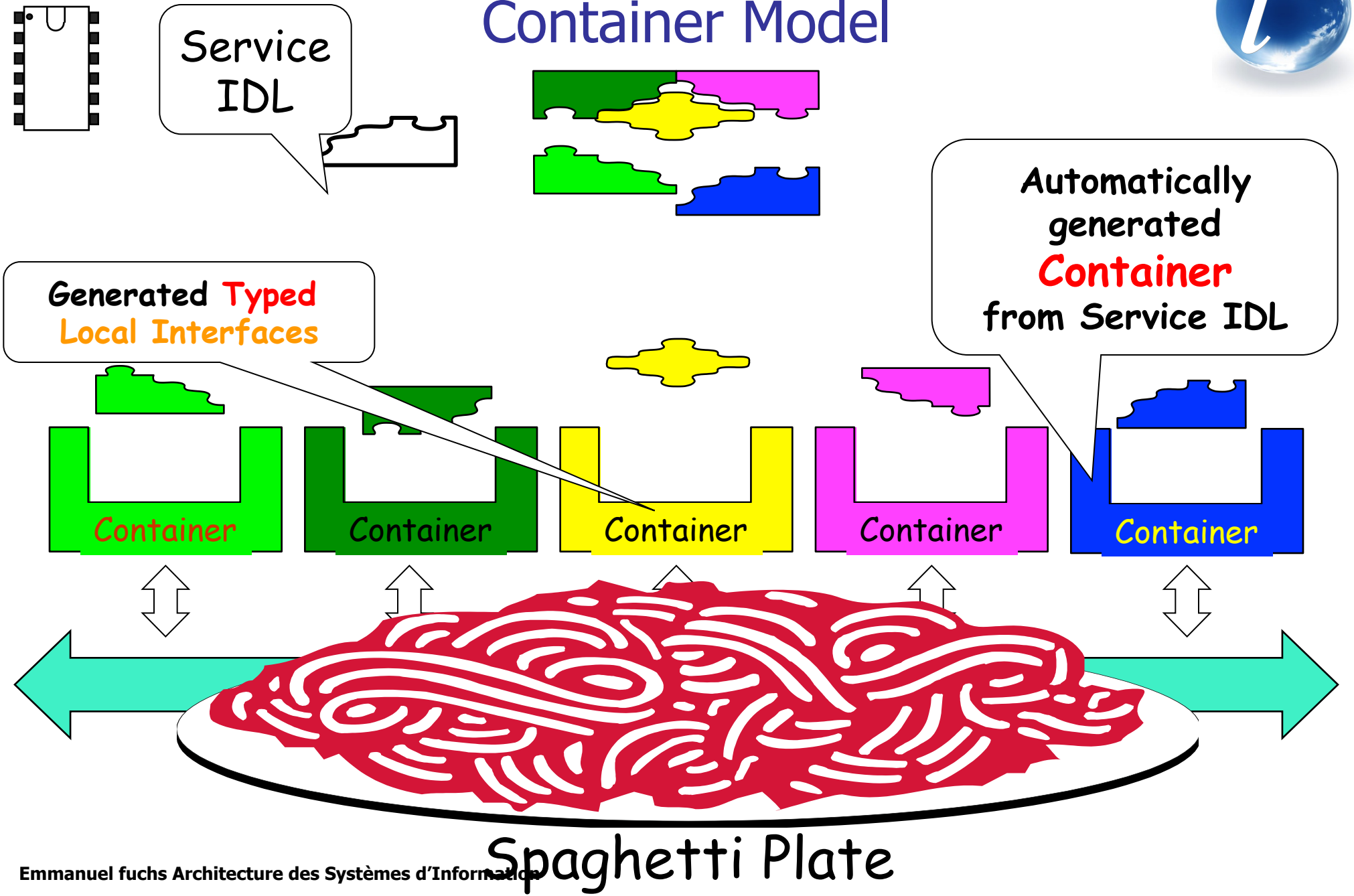




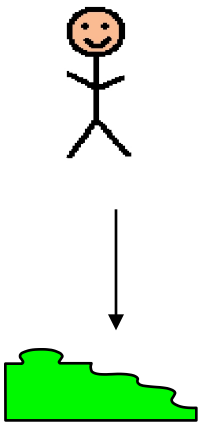


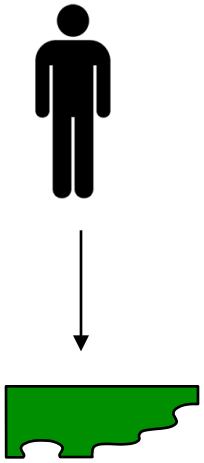
Seamless Integration

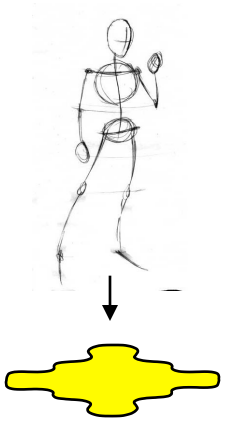
Container Model

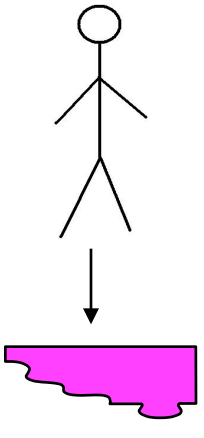


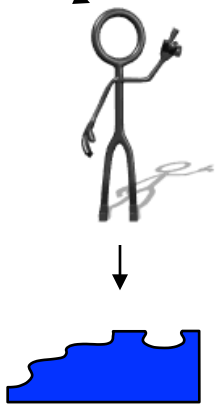
Components Providers



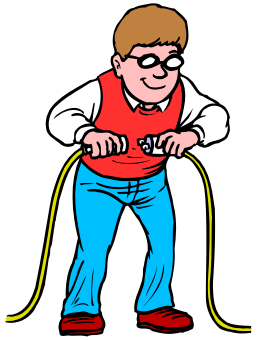


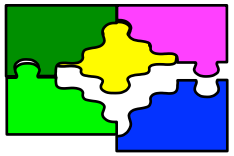


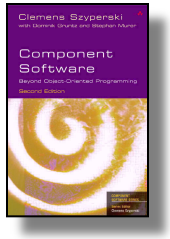


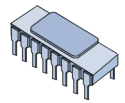


Components Integrator

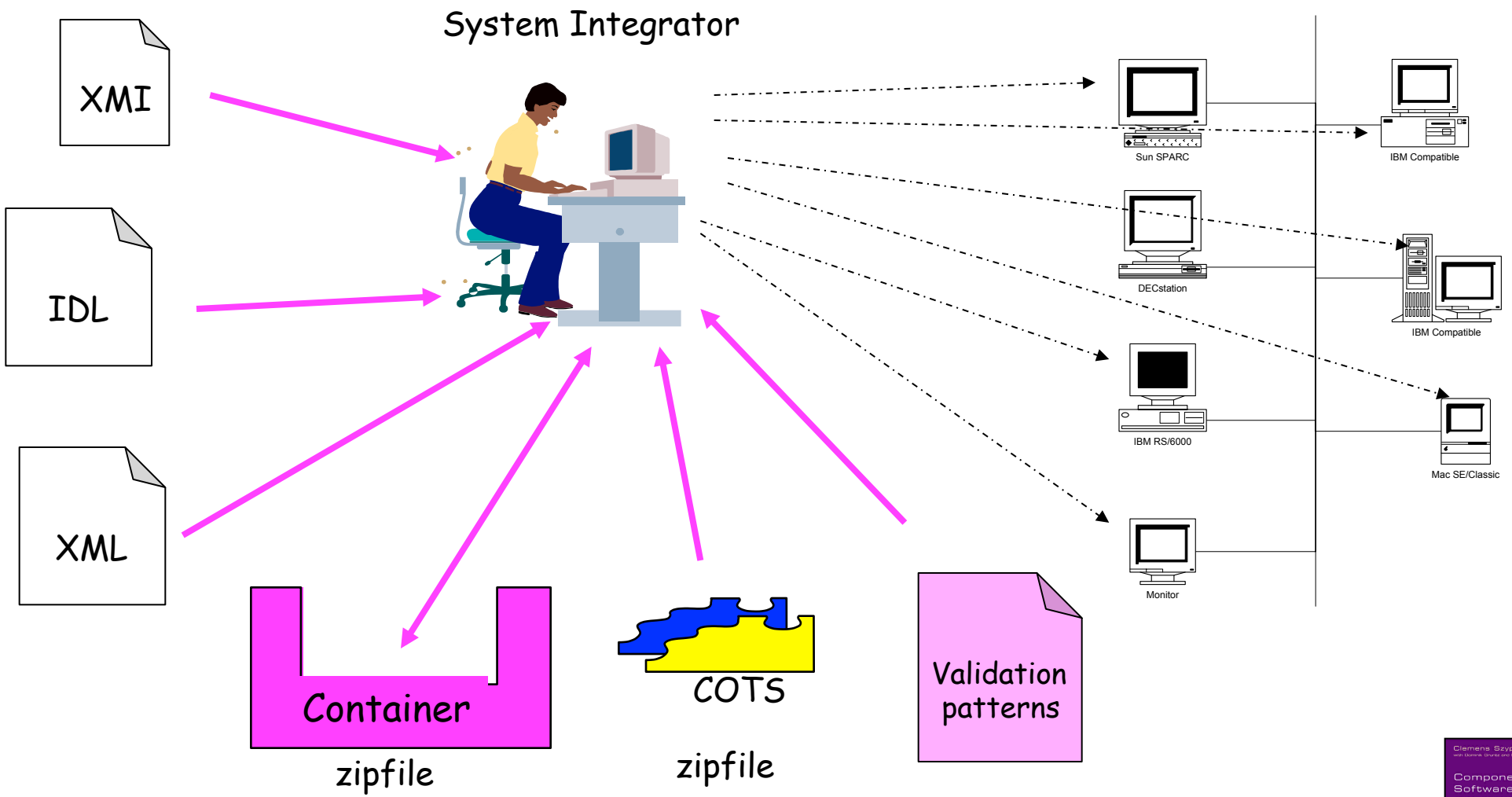






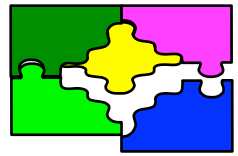


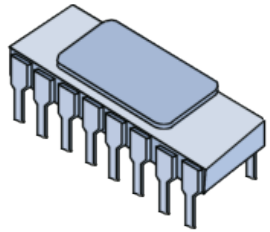
System Deployment and Integration

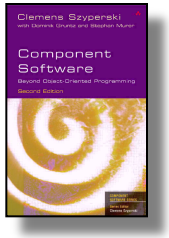


System Architecture

- **A business model**
  - **Domain engineering**
  - **Domain services specification**

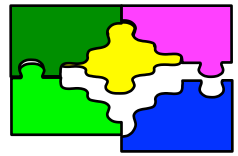






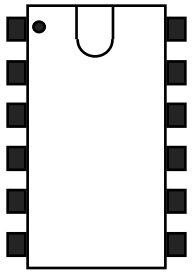
System Architecture

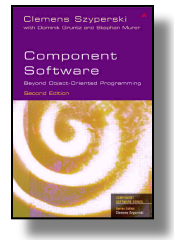
- **A business model**
  - **Domain engineering**
  - **Domain services specification**



- **A component execution model**

  (seamless integration)
  - **Components interconnection**
  - **component model (CM)**
    - Container





Domain Service



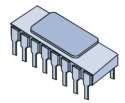
- A service is defined by :
  - An Interface
  - A protocol



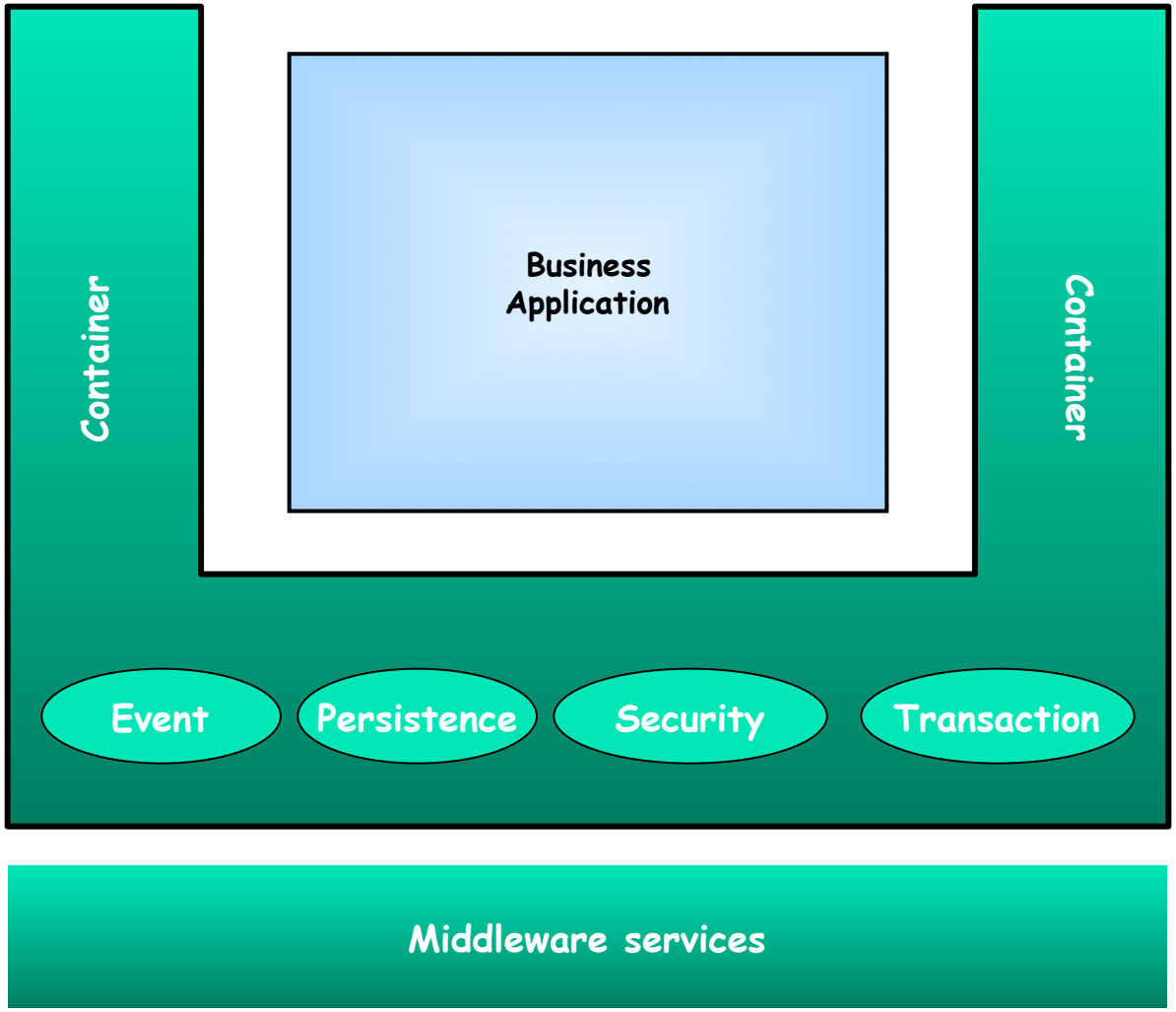
Without any assumptions on :

Execution Platform

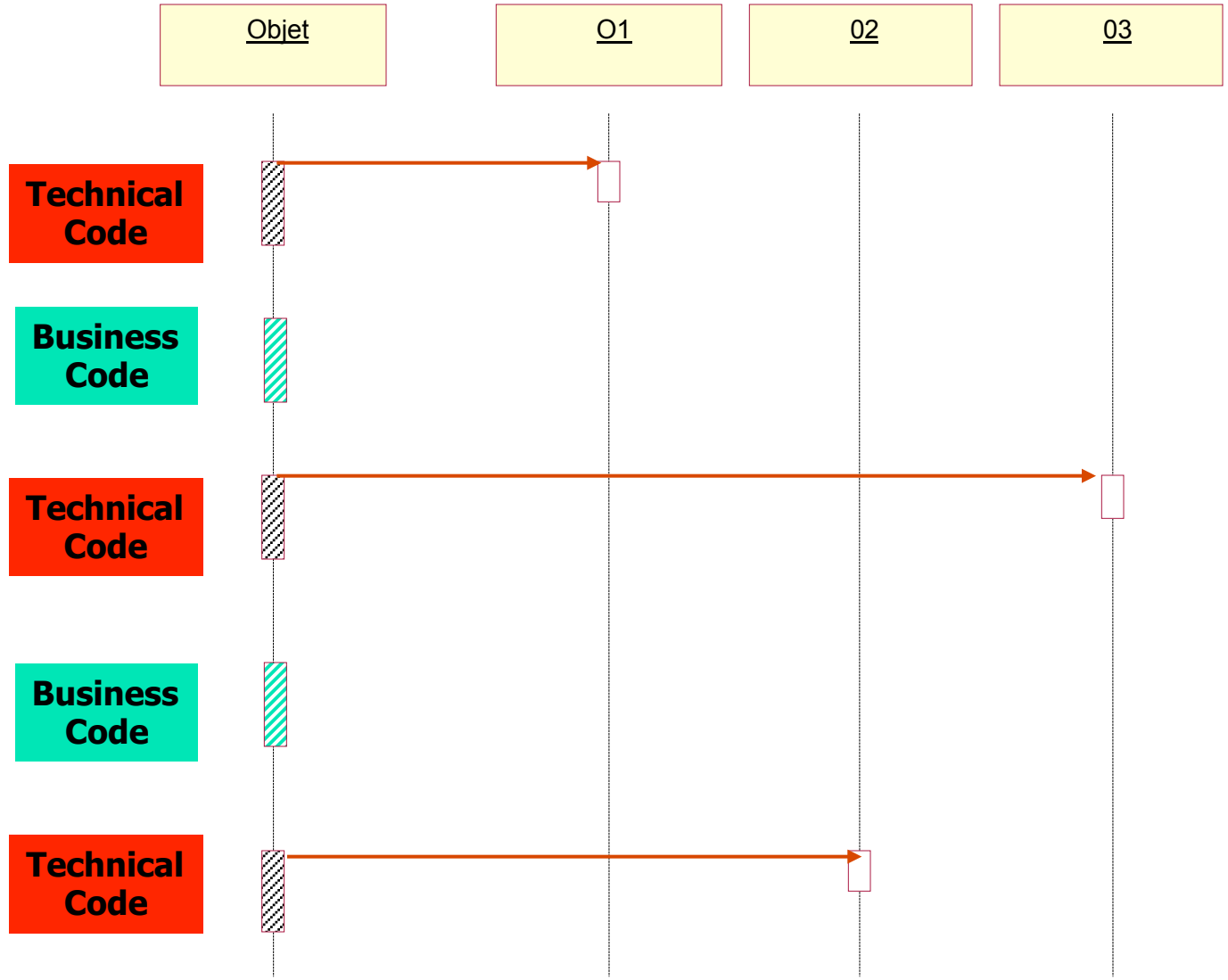
(separation of concerns)

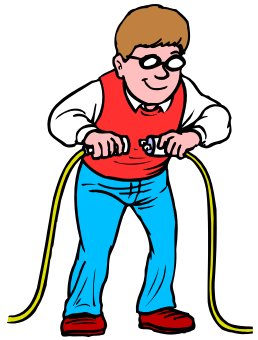


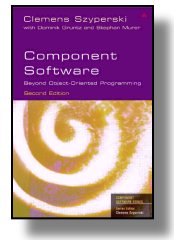
Platform Service dependencies : Container



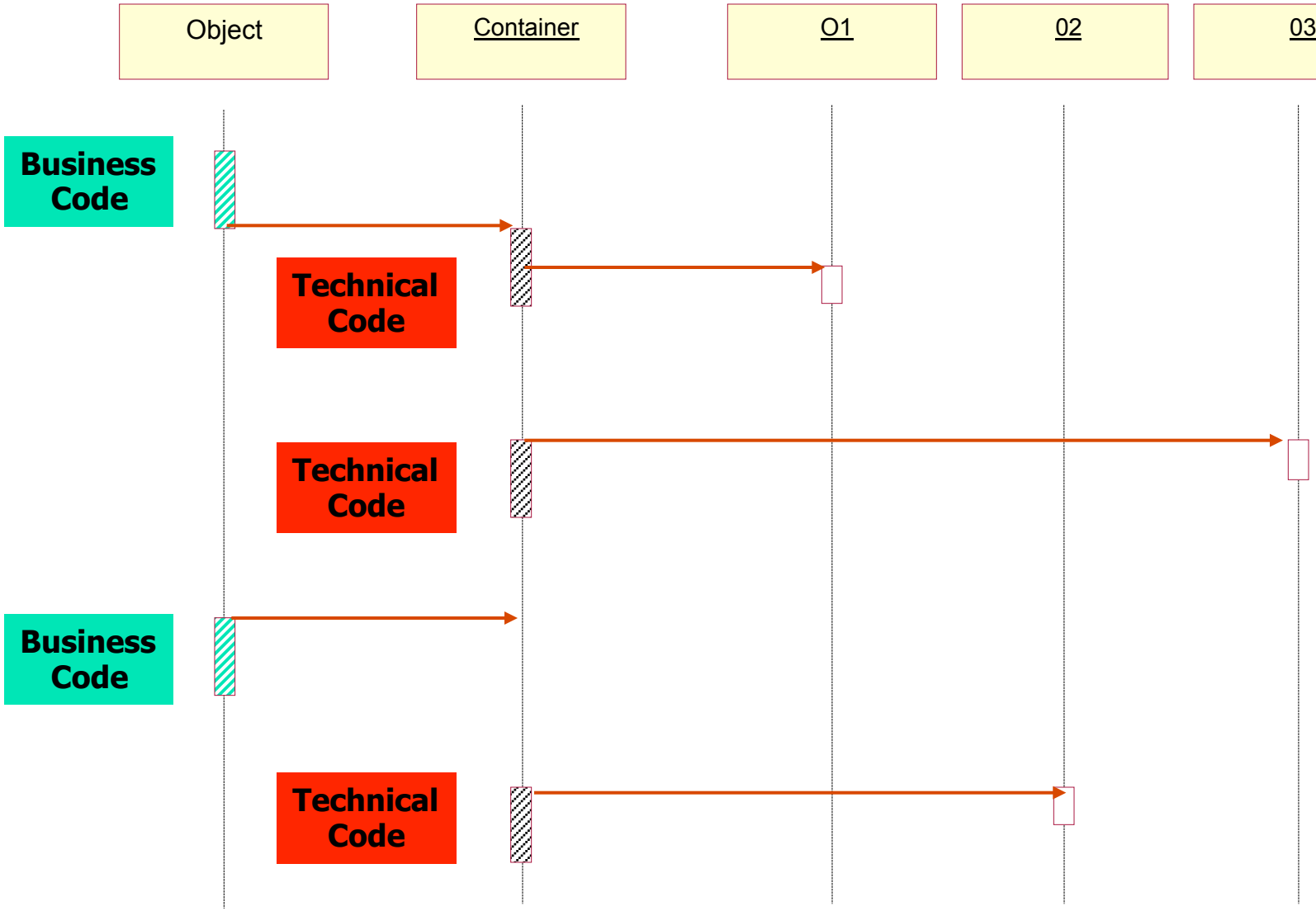
Container and separation of concerns : without container

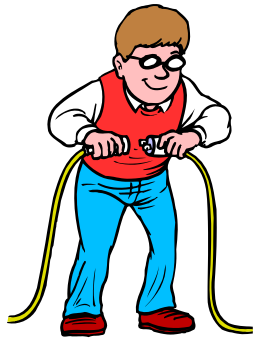


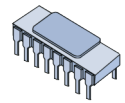




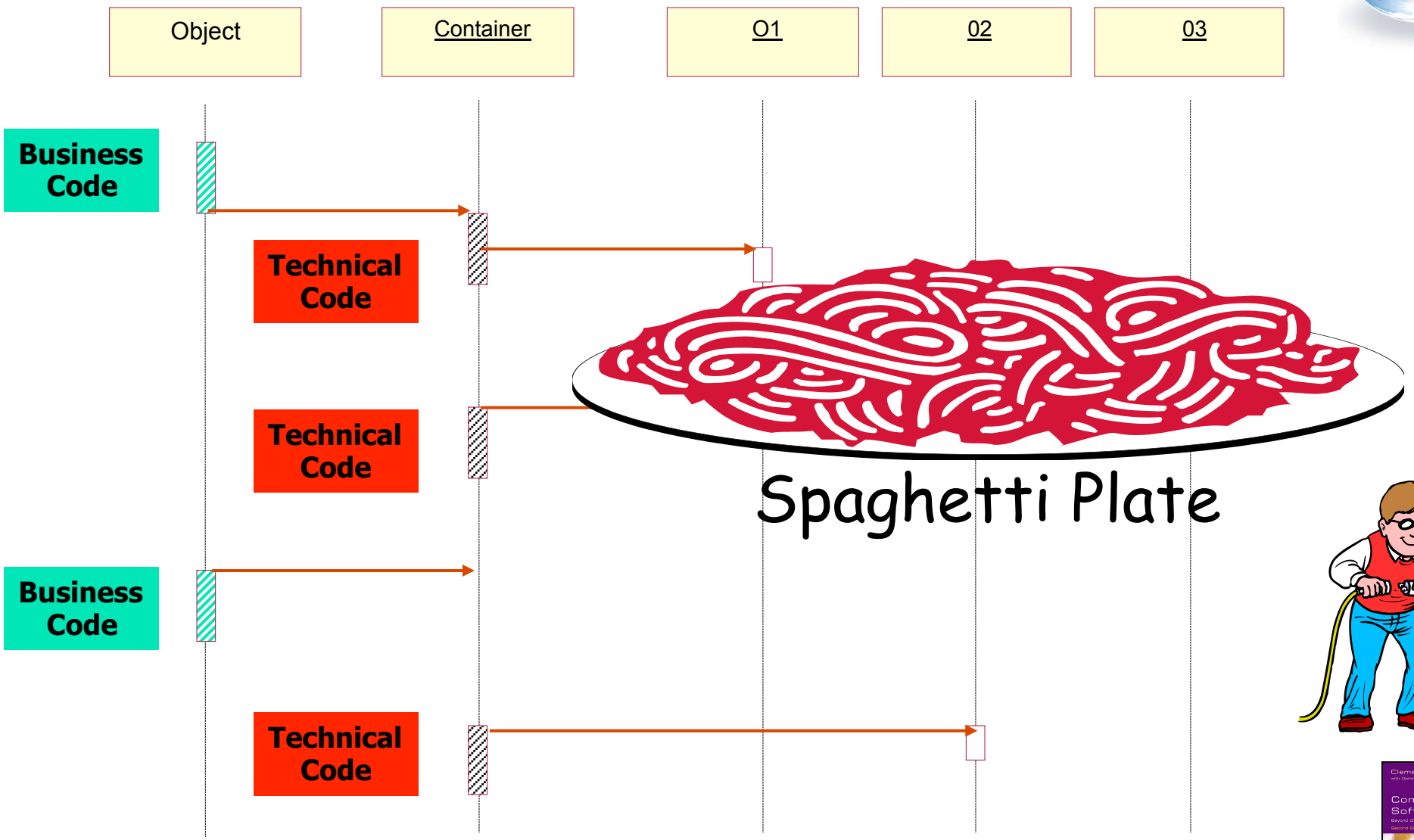
Container and separation of concerns : with container

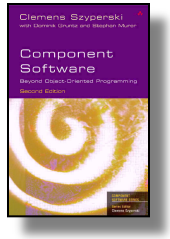




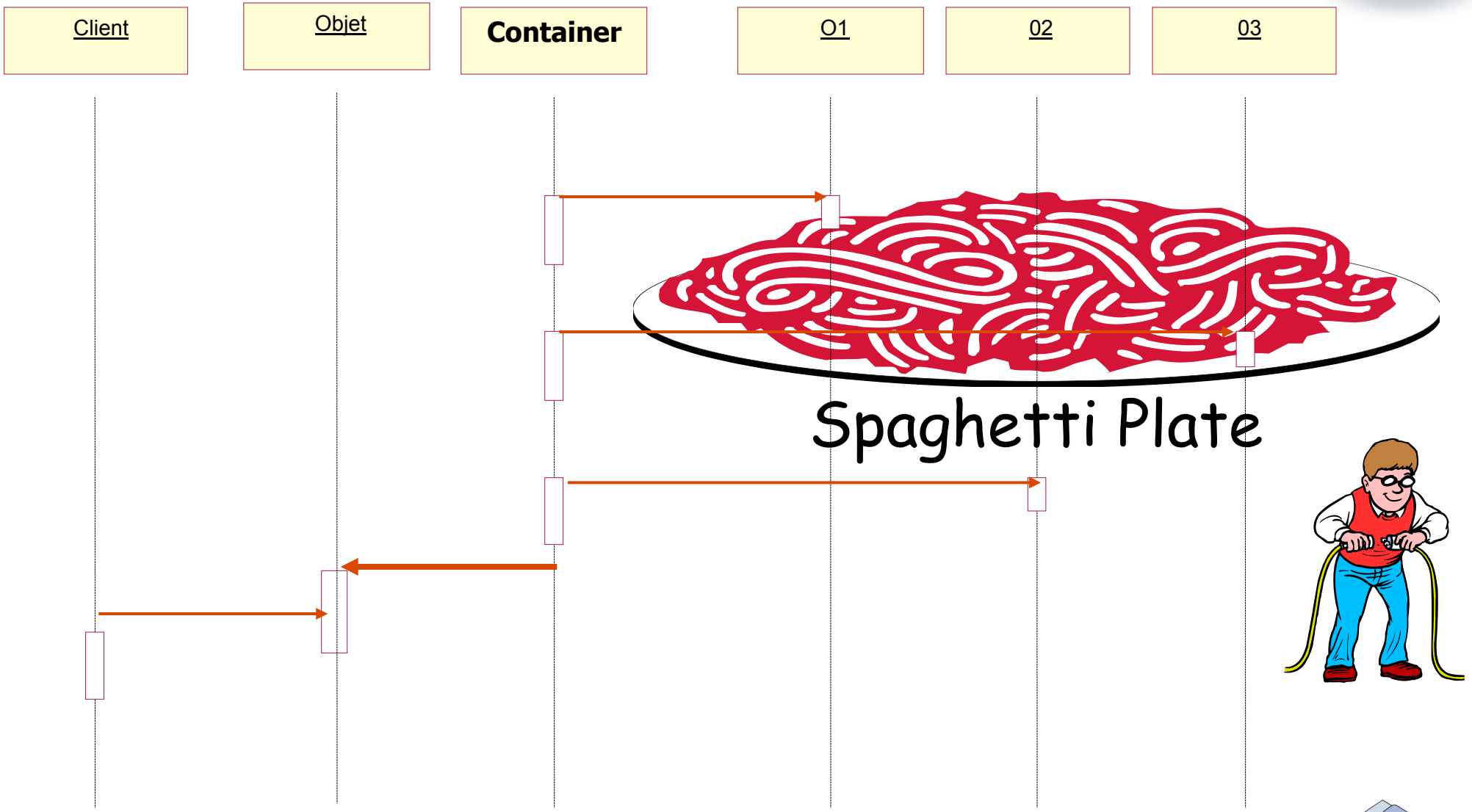


Container and separation of concerns : with container

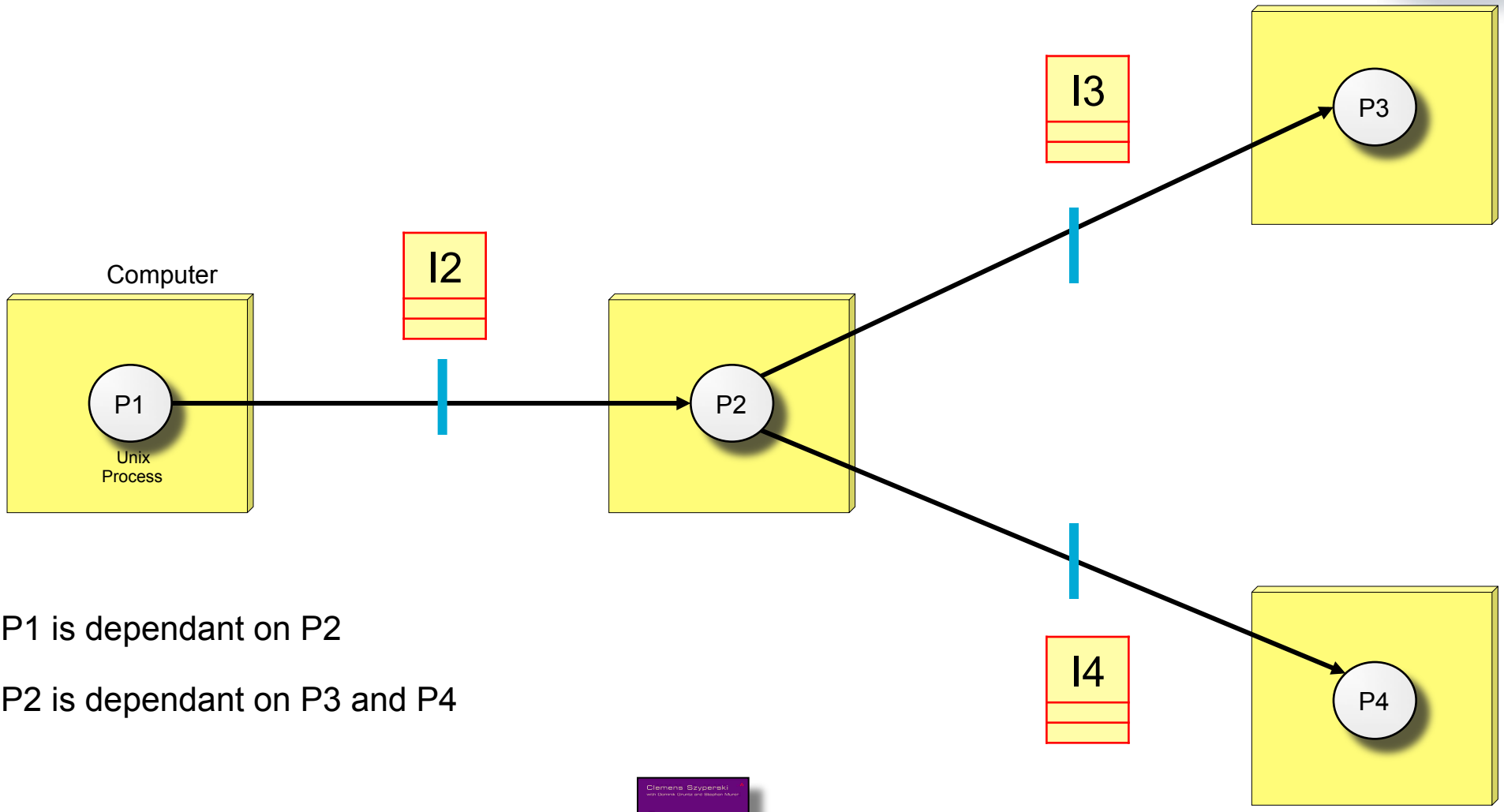




Container and Object Deployment

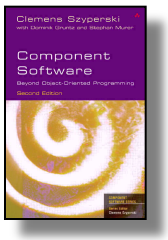


Software dependencies

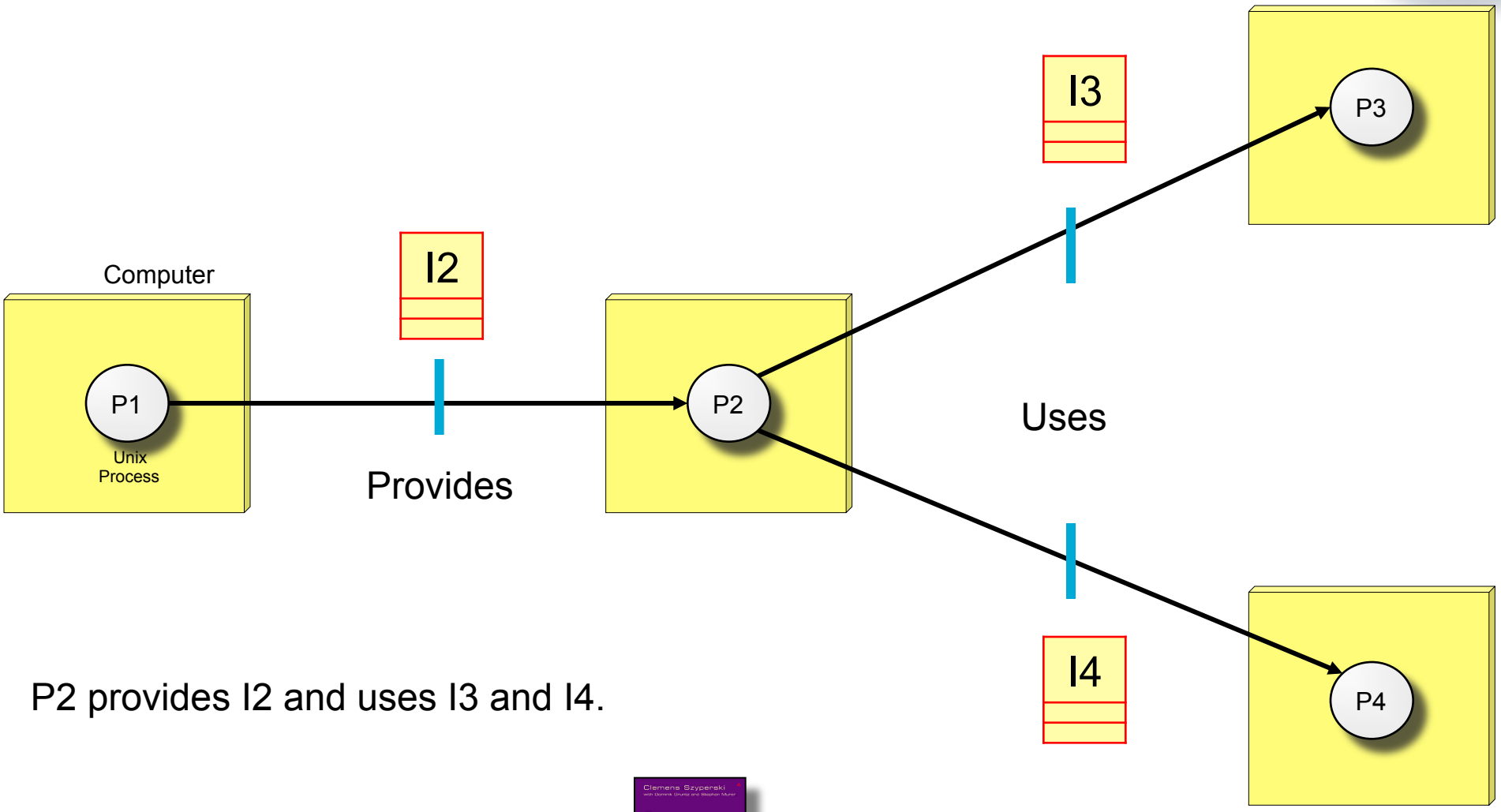


P1 is dependant on P2

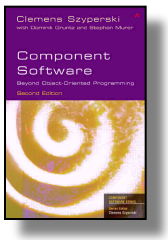
P2 is dependant on P3 and P4



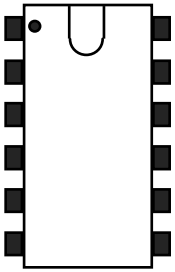
Express Software dependencies



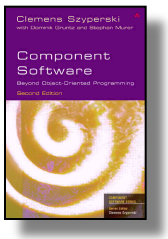
P2 provides I2 and uses I3 and I4.



Component Model

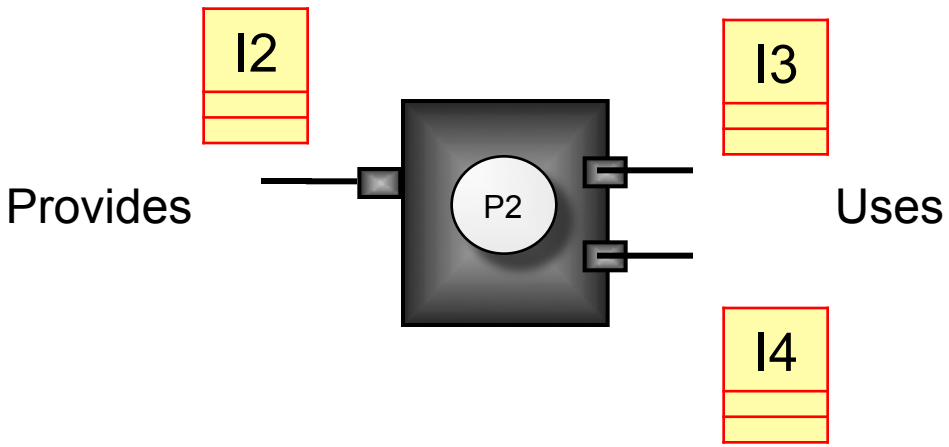


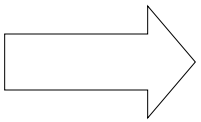
P1 uses I2
P2 provides I2 and uses I3 and I4.
P3 provides I3
P4 provides I4

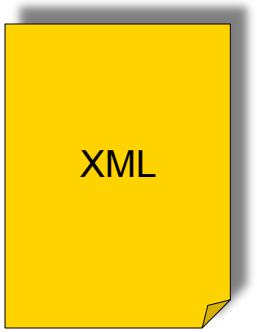


Express Software dependencies

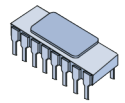
- Component Model provides notation to express software dependencies provided services (XML).
- Dependencies against other component and operating system as well.
- Dependencies description is used to plug and play the software component.





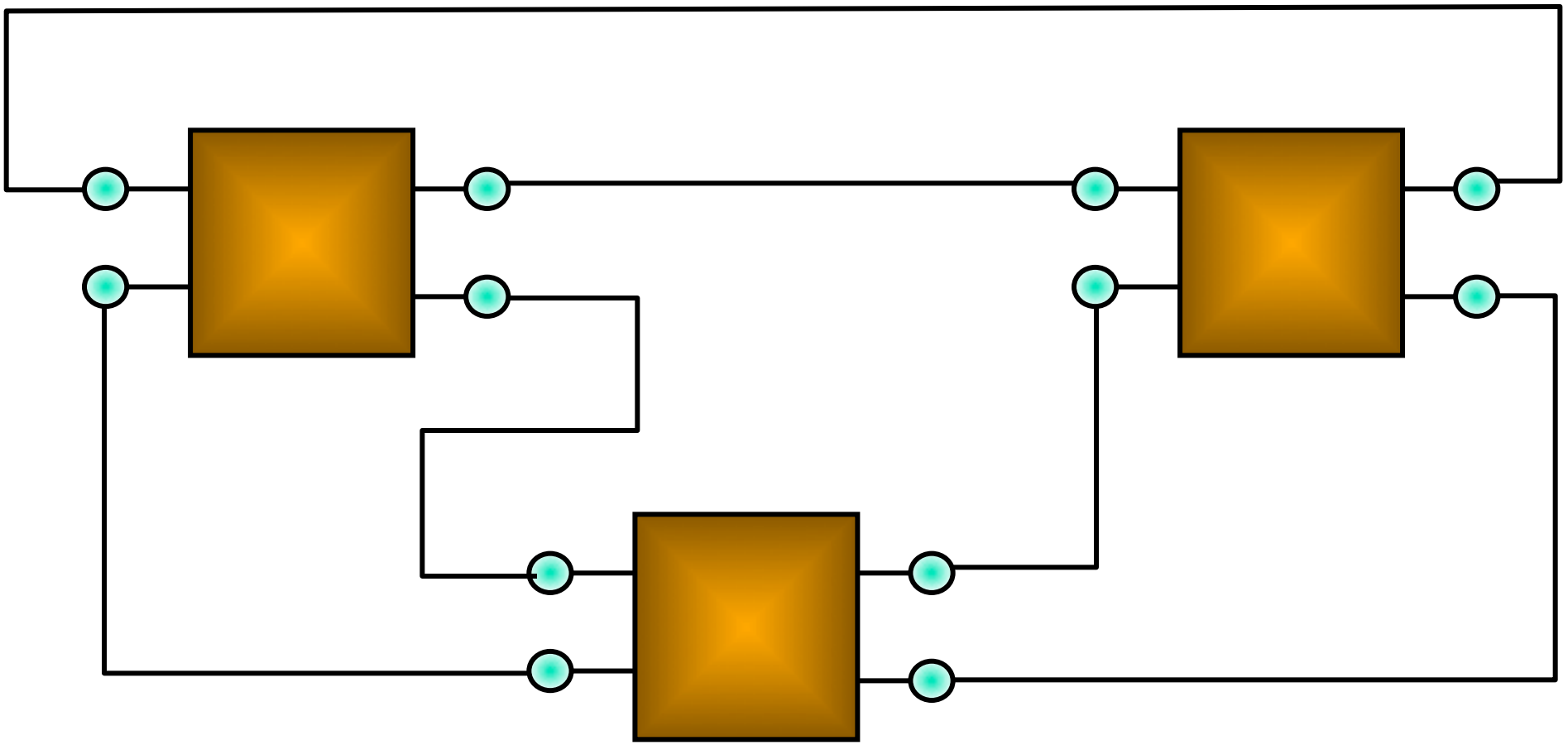


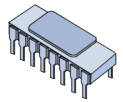
Component Deployment Descriptor



Every things are components

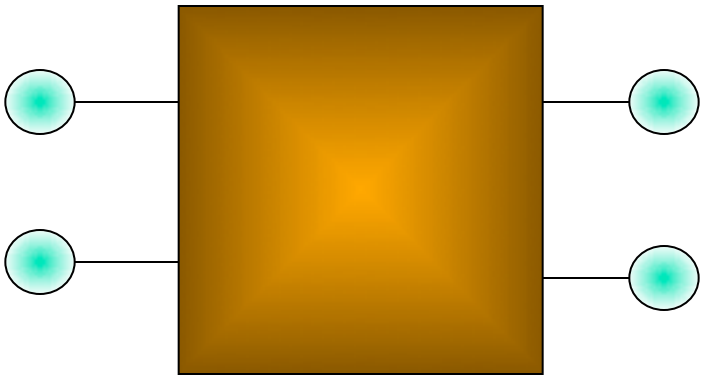
- As long they have ports and connections semantic

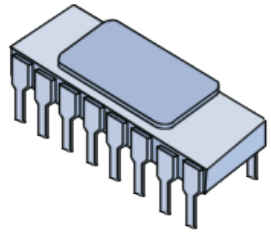


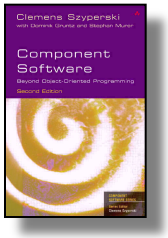


Abstract Component Model

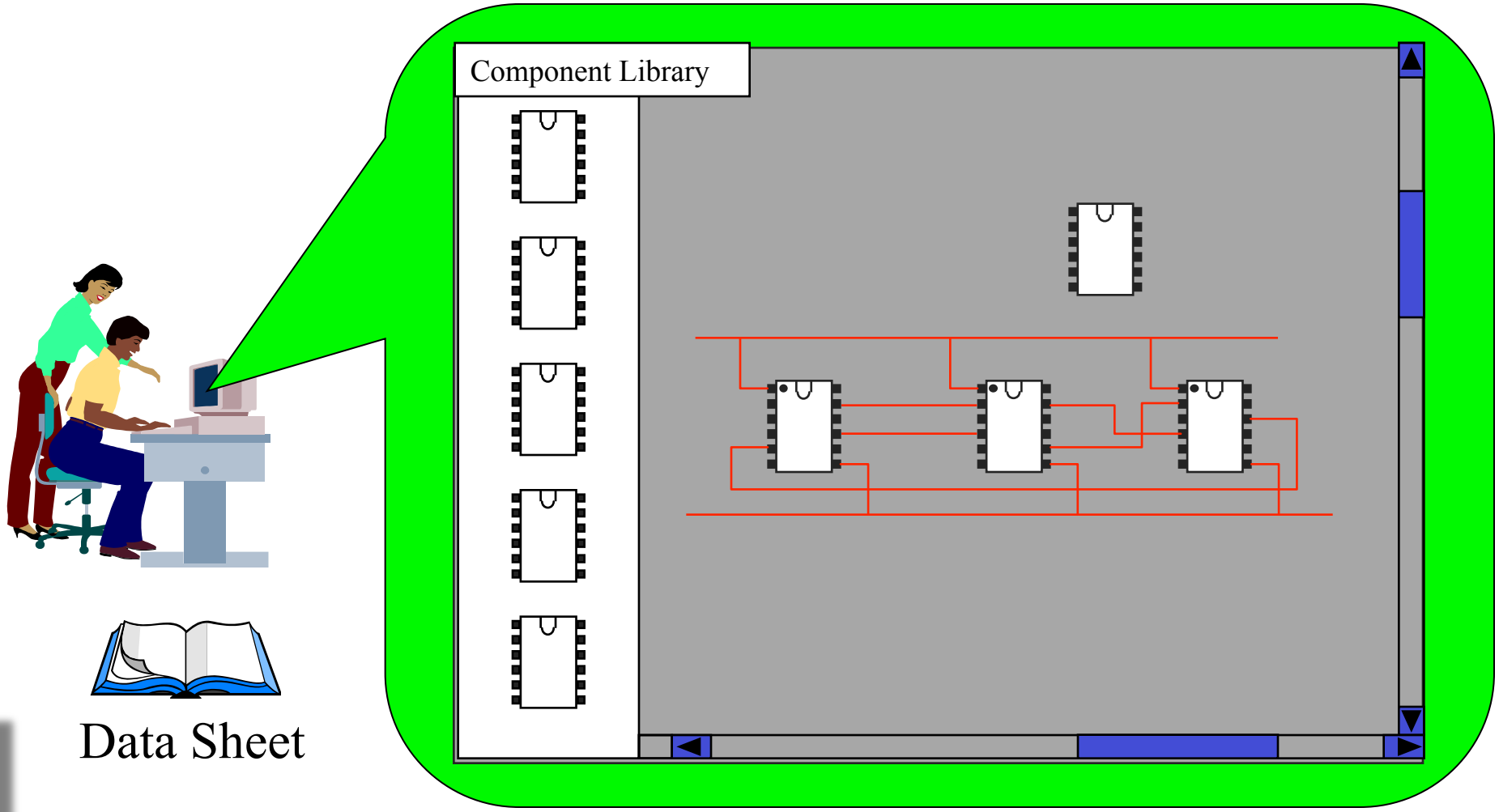
- Defines ports and connectors semantics for distributed object.



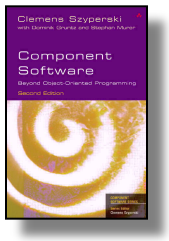




Hardware Components Paradigm

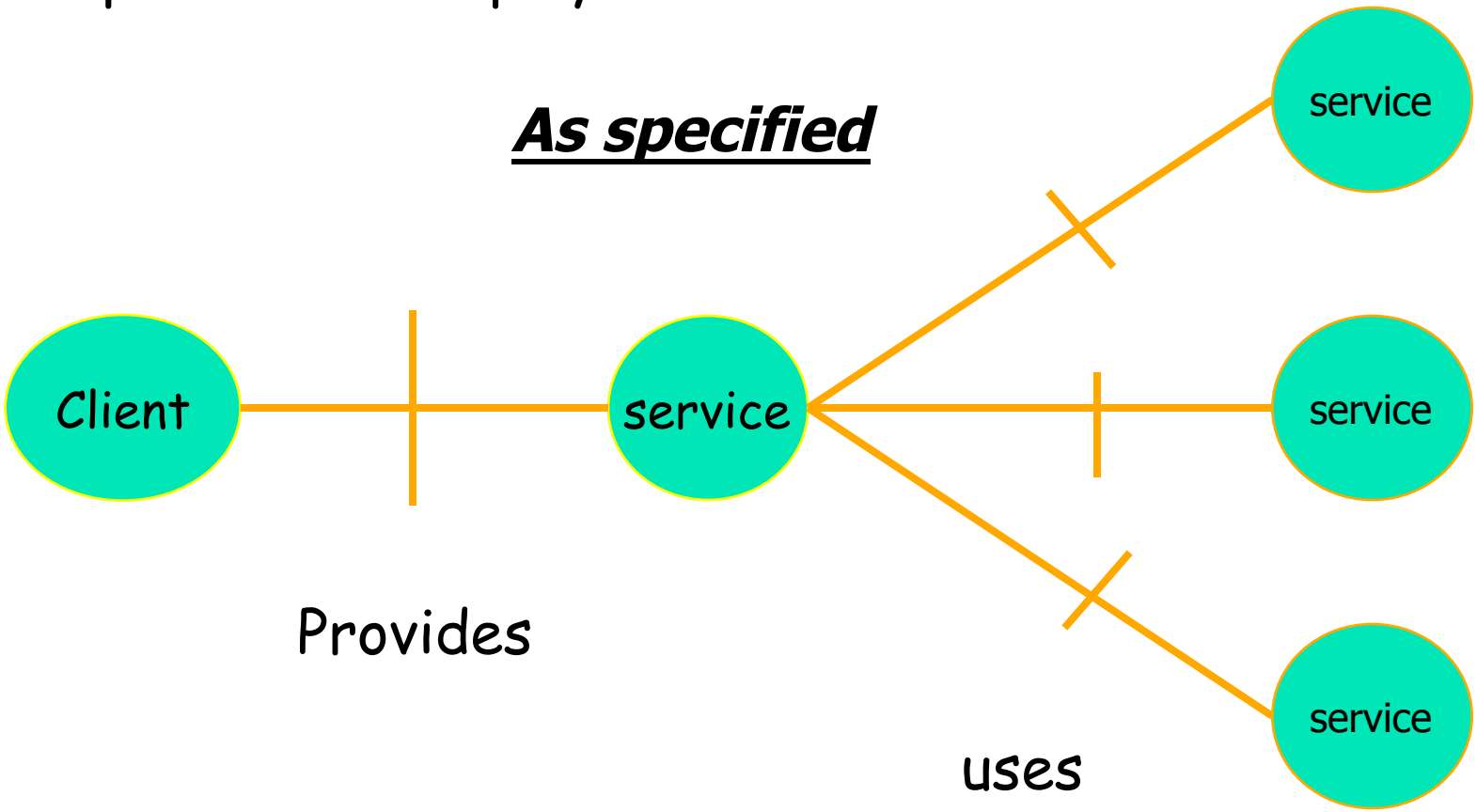


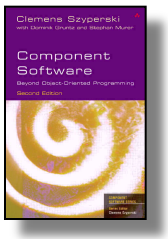
Data Sheet



Component and service

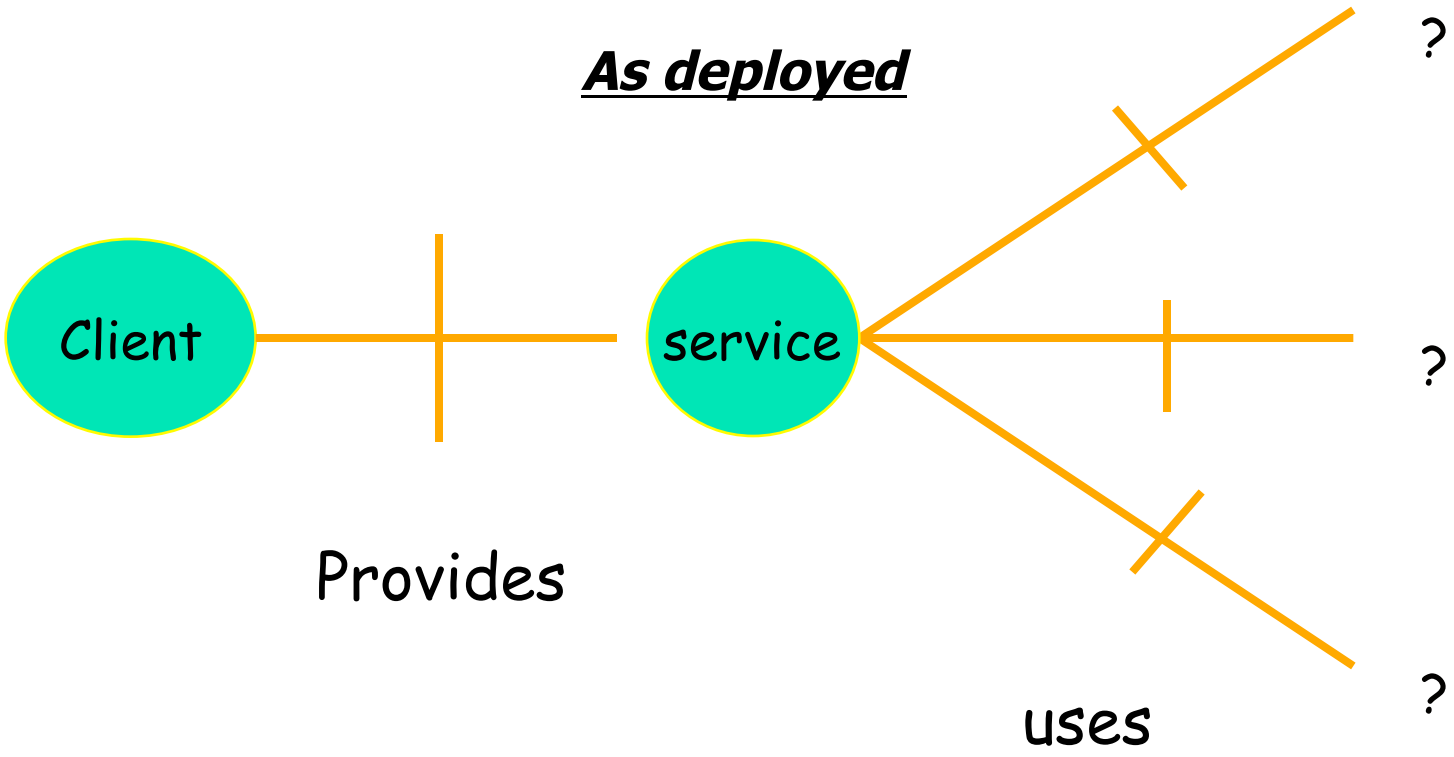
A component is a deployable service

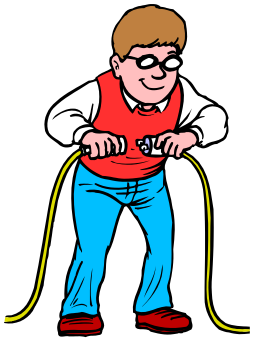


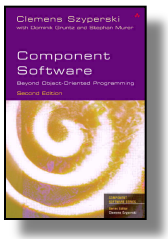


Component and object

A component is a deployable objet





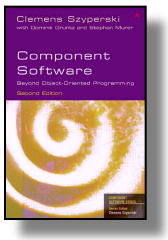


Object Interface (IDL CORBA 2)

Usual CORBA object

interfaces

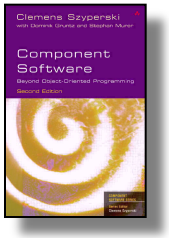
..



Component Interface (IDL CORBA 3)

Usual CORBA object

```
Provides {
        interfaces
        ..
}
```

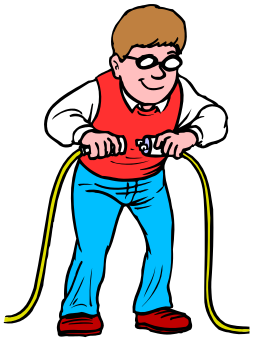


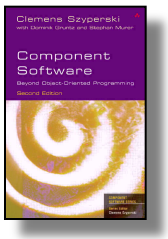
Component Interface

Service

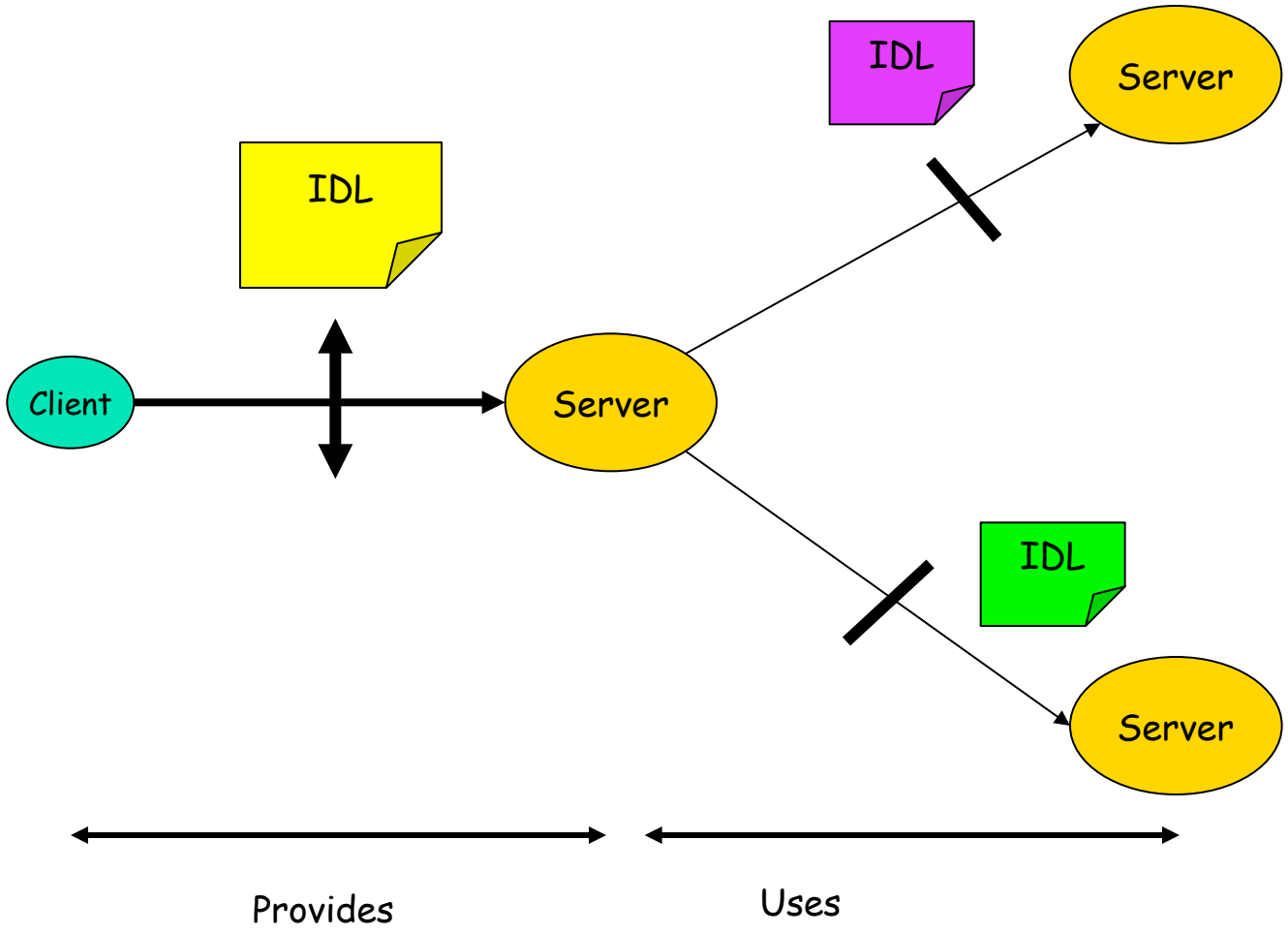
```
Provides {
        interfaces
        ..
}

uses {
        interfaces
        ..
}
```

Connection to Service

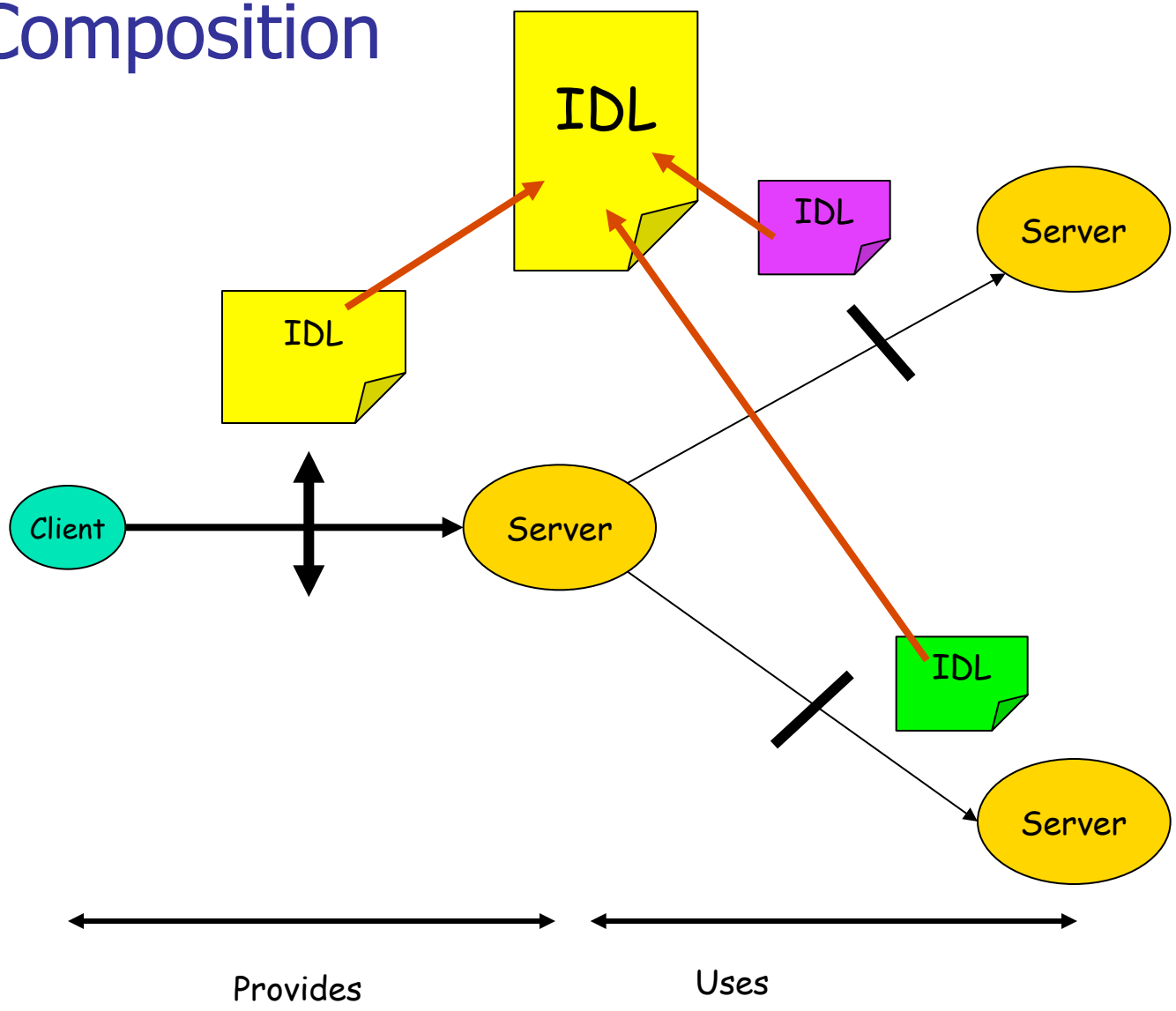




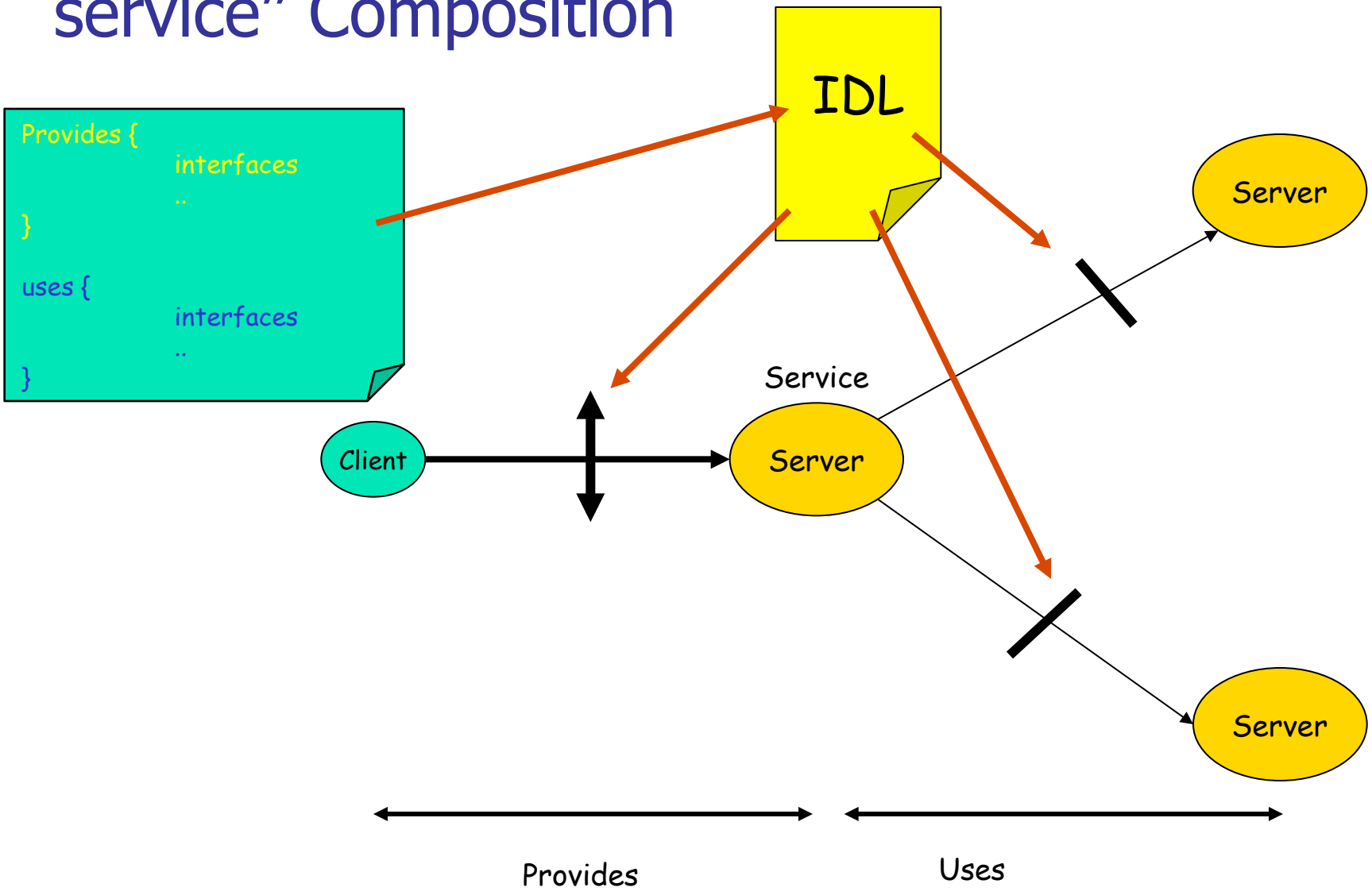
Service Composition



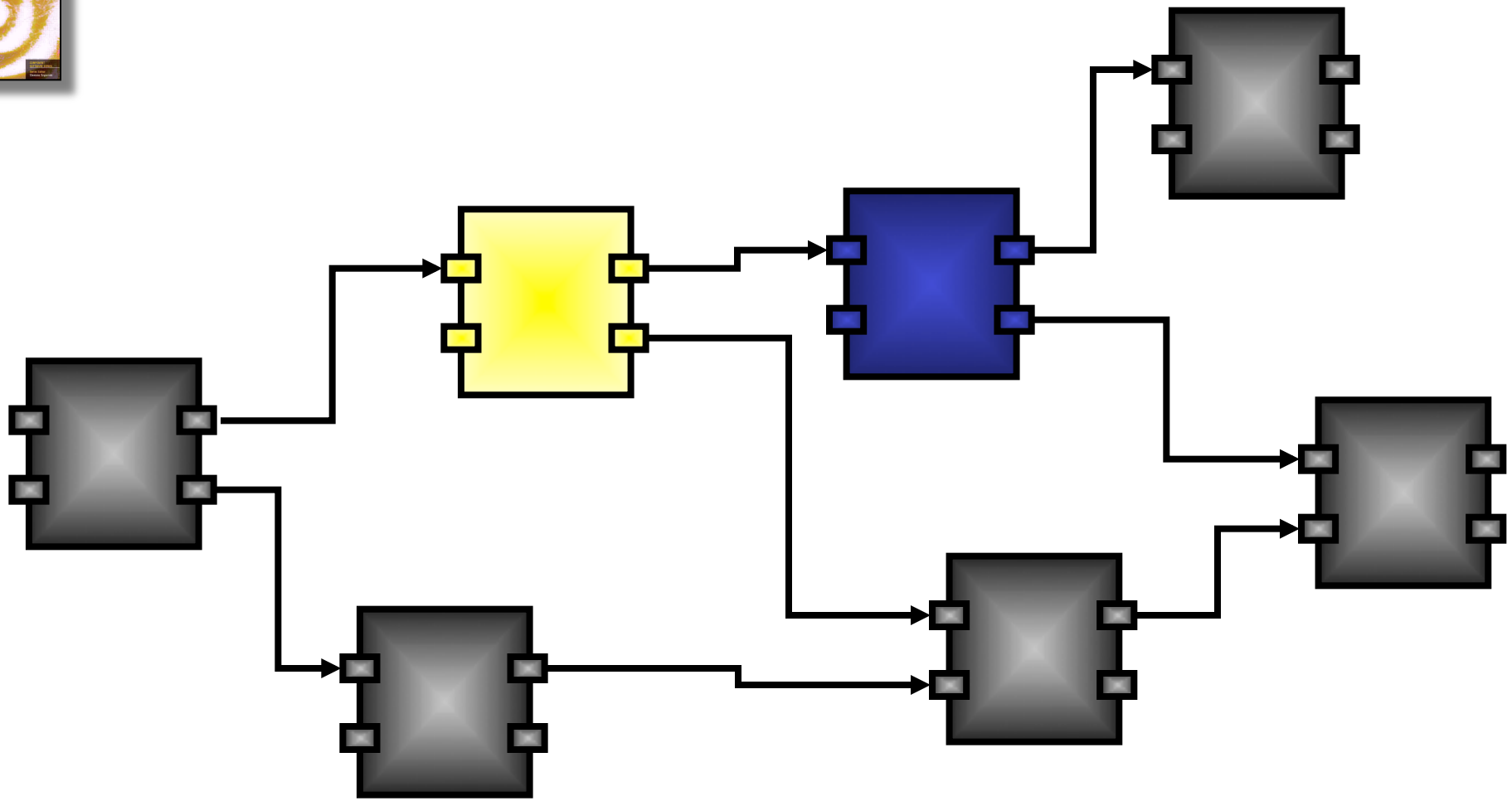
Service Composition



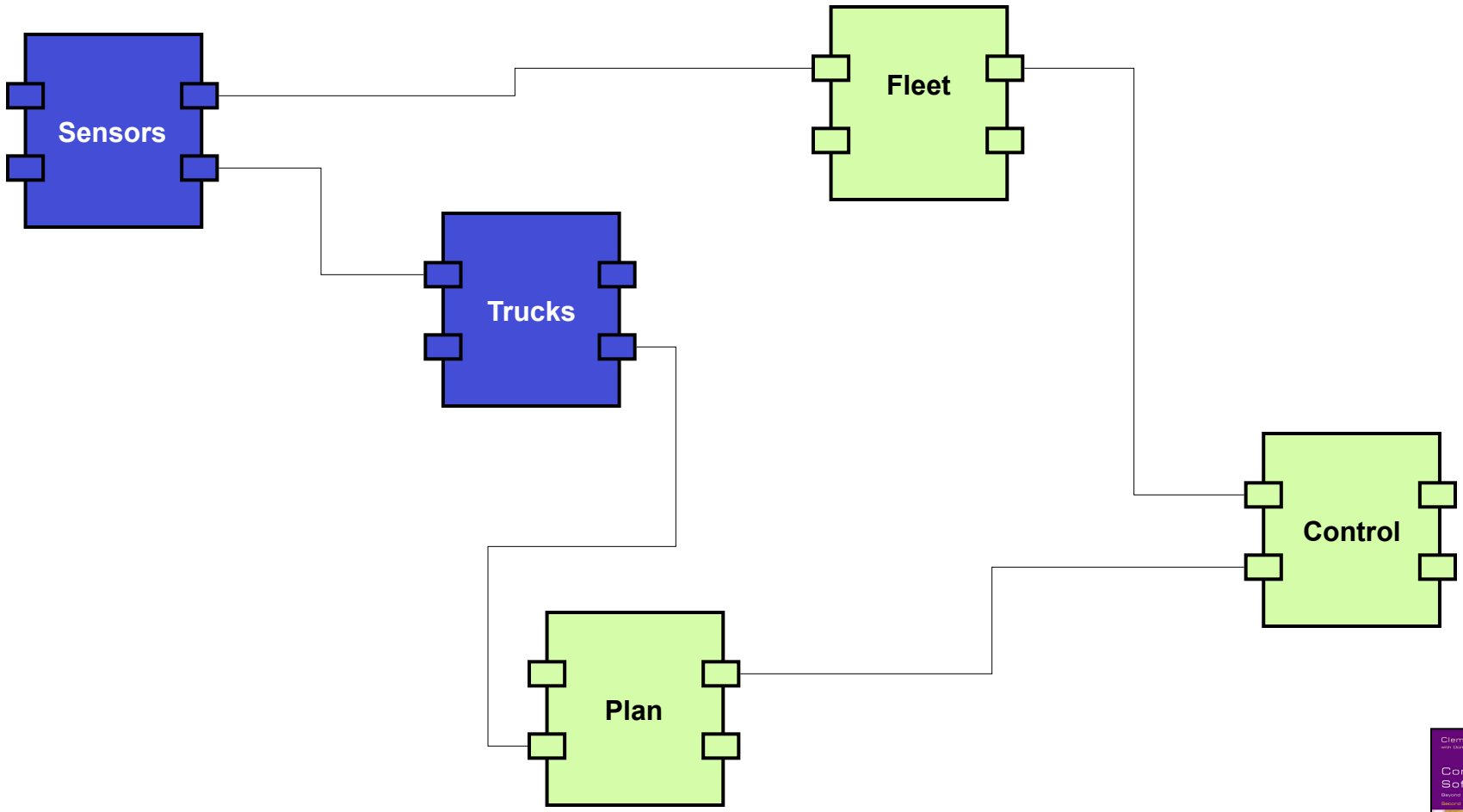
service" Composition

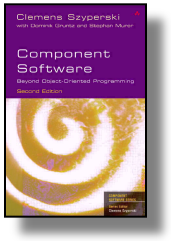


Component Distributed Assembly

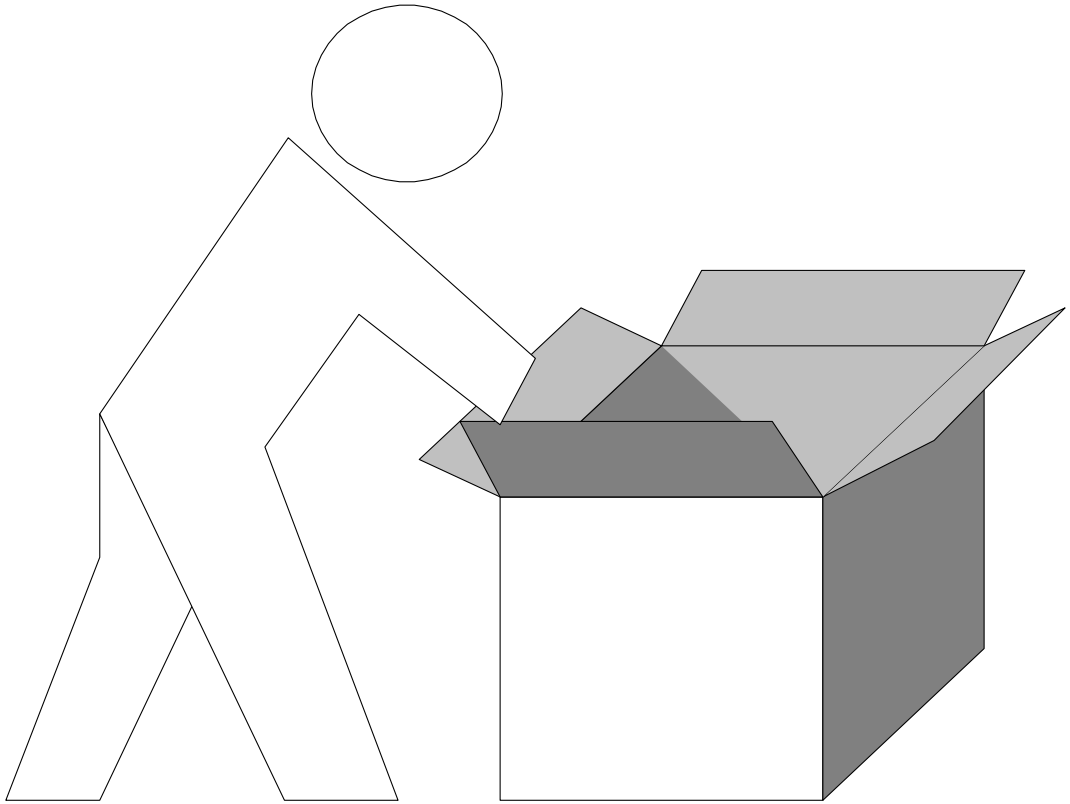


Component Distributed Assembly



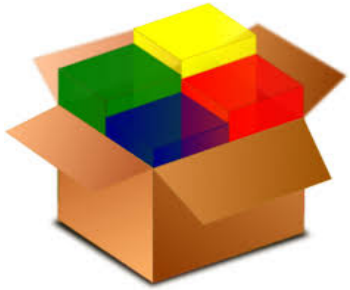


Component Packaging



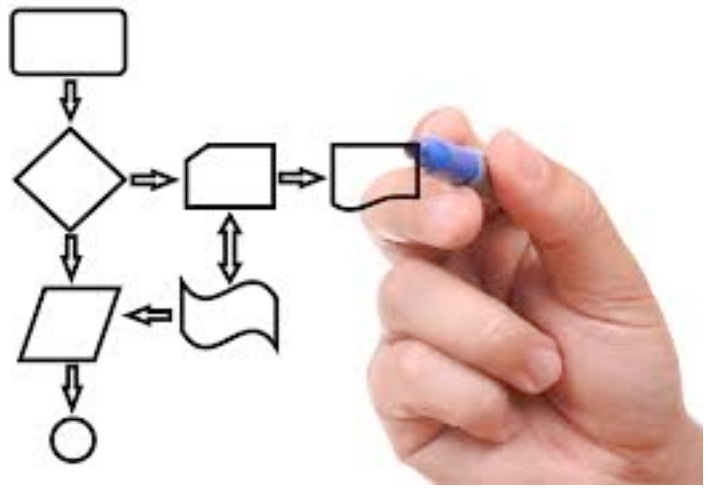
Cycle produit logiciel (Composant)

- Cycle de développement
  - Développement
  - Packaging
  - Shipping
  - Déploiement
  - Exécution
- Actors
  - Components Provider
  - Application Assembler
  - Components Deployer
  - Components Server Provider
  - Components Container Provider
  - Components Persistence Provider
  - Components System Administrator

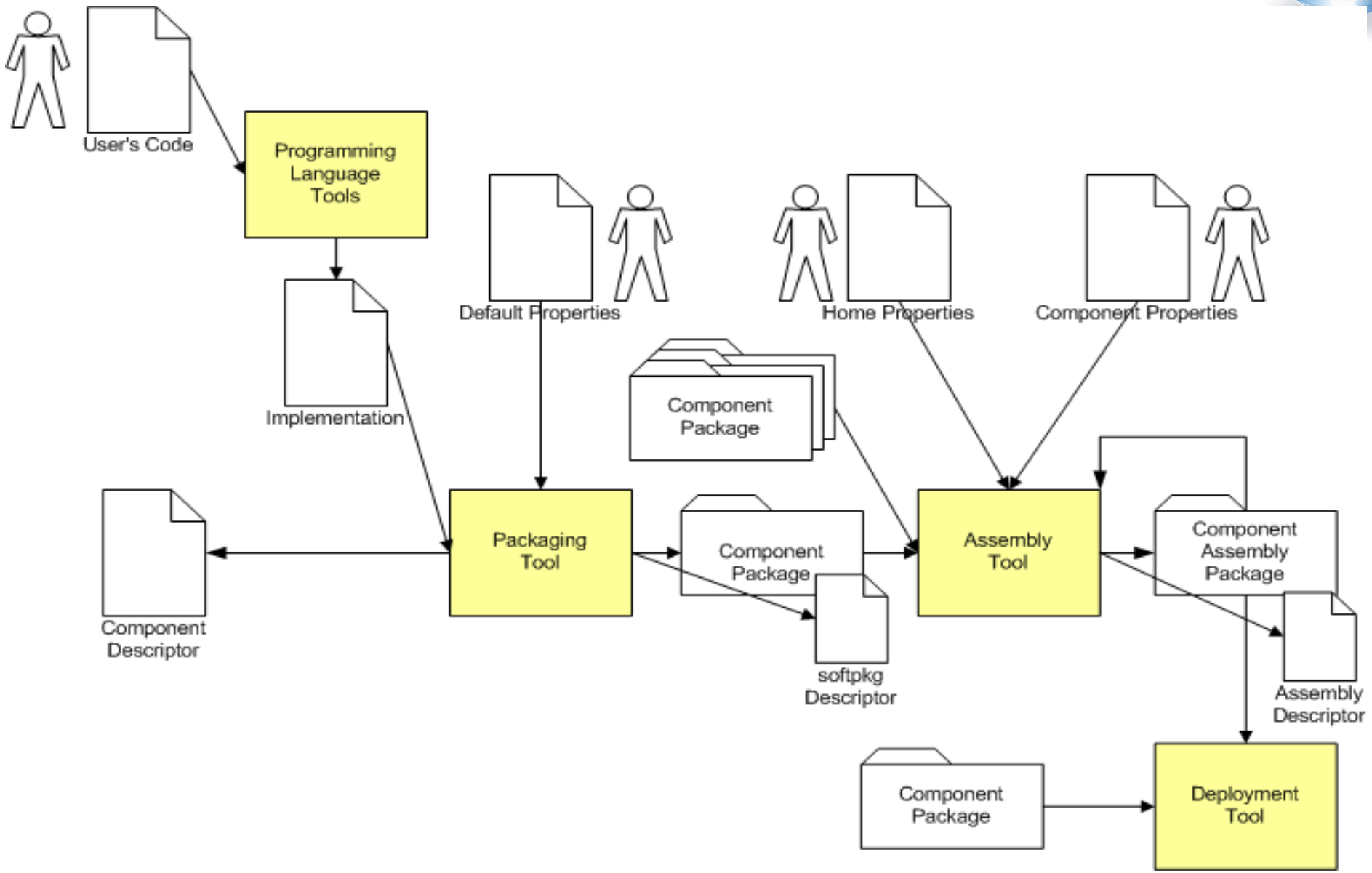


Component Development Life Cycle

- A component is specified.
- A component is implemented.
- A component is packaged.
- A component may be assembled with other components (usually in a design tool).
- Components and/or assemblies are deployed.



Packaging Assembly Deployement Tools



Well Known Components Model

- Well know local component model:
  - Eclipse
  - Firefox Mozilla : Xpcom (Cross Platform Component Object Model)
  - Open Services Gateway Initiative (OSGi)
  - Sun JavaBeans
  - Java Swing
  - Enterprise Java Beans (EJB)
  - Java Server Face (JSF)
  - COM/DCOM

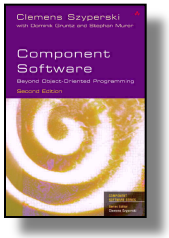
Firefox
the browser, reloaded.

Well Known Components Model

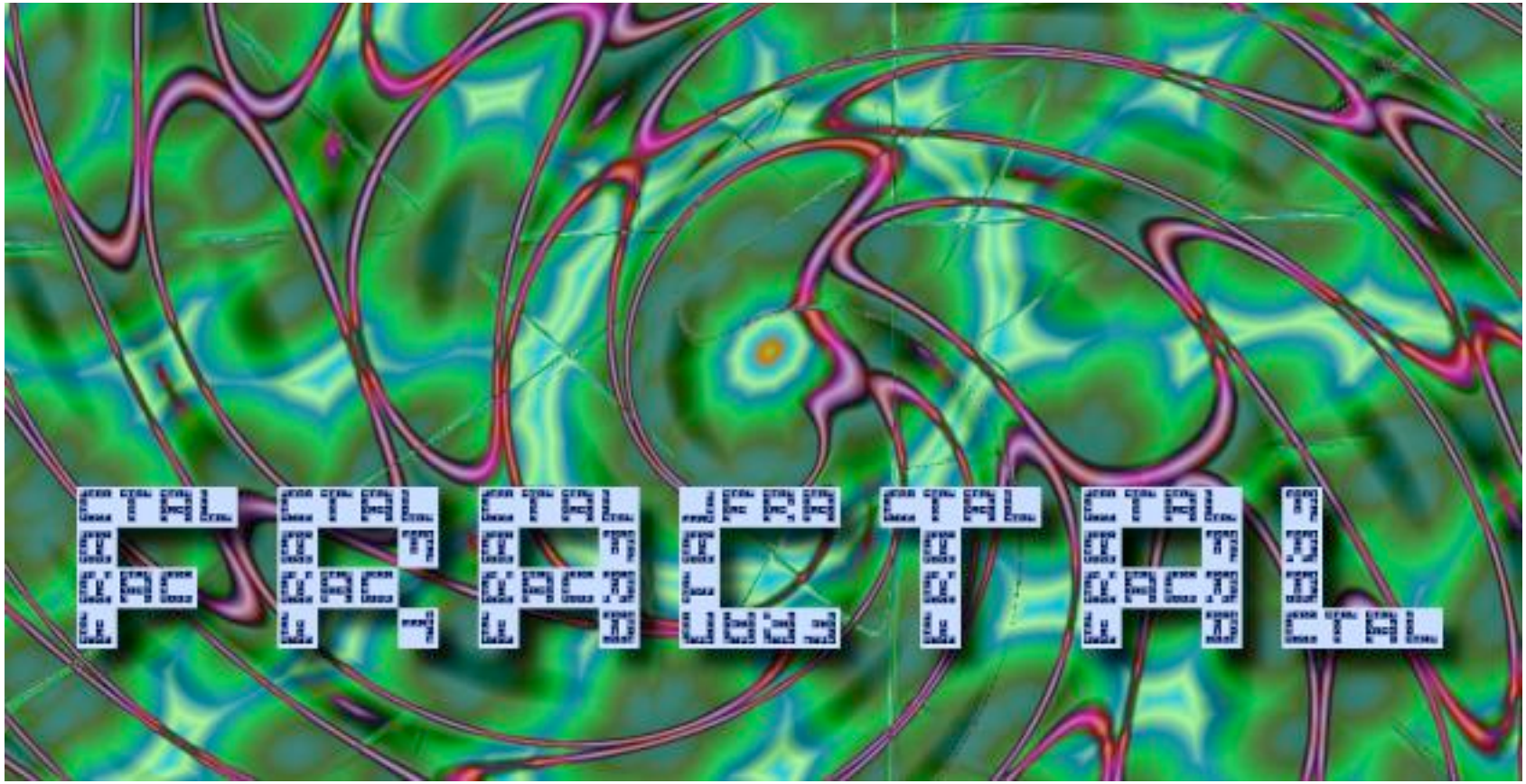
- Well know distributed component model:
  - DCOM, .NET
  - CORBA Component Model (CCM)
  - Enterprise Java Beans (EJB)

OpenEJB™

Microsoft .net



FRACTAL



http://fractal.ow2.org/index.html

- **Fractal is a modular, extensible and programming language agnostic component model** that can be used to design, implement, deploy and reconfigure systems and applications, from operating systems to middleware platforms and to graphical user interfaces.
- The goal of Fractal is to reduce the development, deployment and maintenance costs of software systems in general, and of ObjectWeb projects in particular.

ObjectWeb

The open source community for infrastructure software

http://fractal.ow2.org/index.html

- The Fractal component model has the following important features:
  - **recursivity**:
    - components can be nested in *composite* components (hence the "Fractal" name).
  - **reflectivity**:
    - components have full introspection and intercession capabilities.
  - **component sharing**:
    - a given component instance can be included (or shared) by more than one component.
    - This is useful to model shared resources such as memory manager or device drivers for instance.
  - **binding components**:
    - a single abstraction for components connections that is called *bindings*.
    - Bindings can embed any communication semantics from synchronous method calls to remote procedure calls
  - **execution model independence**:
    - no execution model is imposed. In that, components can be run within other execution models than the classical thread-based model such as event-based models and so on.
  - **open**:
    - extra-functional services associated to a component can be customized through the notion of a control membrane.