

# Software Composition

Component-Based Software Engineering

[www.win.tue.nl/~mchaudro/cbse2007](http://www.win.tue.nl/~mchaudro/cbse2007)

M.R.V. Chaudron

TU Eindhoven

# Outline

- Recap composition last week
- Composition of Features, Systems
- Place (Locus) and Time of Composition

# How does a system know where to 'return' the result of a function call?

- Answer follows later ...

# Questions ?

Homework:

- Read about pipes and filters

[http://en.wikipedia.org/wiki/Pipes\\_and\\_filters](http://en.wikipedia.org/wiki/Pipes_and_filters)

- Think about:
  - Composing/Binding software?
    - What do architects/engineers need?
    - In what ways is software currently composed?
    - How does binding work in Pipes and Filters?

# What do you want to compose?

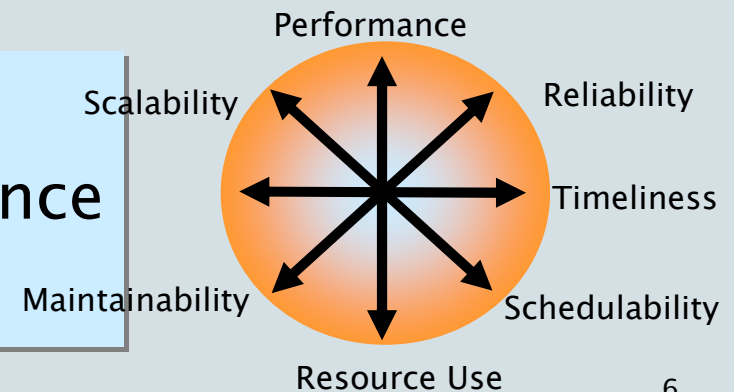
## Aspect of Scale / Granularity:

- Instructions
  - Methods
  - Modules / Classes
  - Frameworks / Libraries
  - Services
  - Components / Aspects
  - Features
  - Systems
- } intra-language
- } intra-architecture
- } across systems

## What should be the aim of composing software ?

- Extension?
- Interoperation?
- Functional composition?
- Synchronisation?
- Containment?
- ... Suggestions from the audience ...

**Composition = the combining of functions & features under invariance of quality properties**



# Composition: the programming language level

- Functional, Logic, Imperative Programming Languages
  - Composing functions, predicates, instructions
  - Assume underlying execution model
- Object Oriented Programming Languages
  - Composition of program text, partially in execution model
- Invariance of form – interfaces
- Introduction of state complicates matters significantly

# Composition in OO Prog. Lang's

- OO Prog. Lang's: Simula, C++, Java, ...
- Objects are essentially a structuring/abstraction mechanism
- Main composition mechanism:
  - Inheritance
  - Mixin's (Bracha)

```
Class P
{
  method Q
  method R
}
```

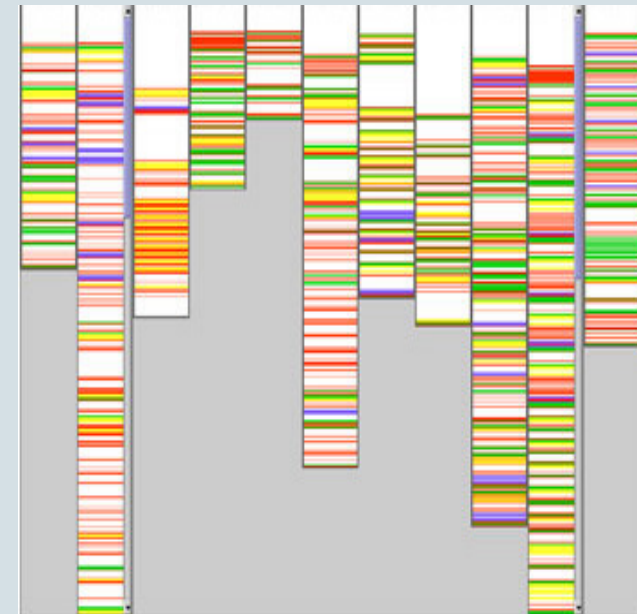
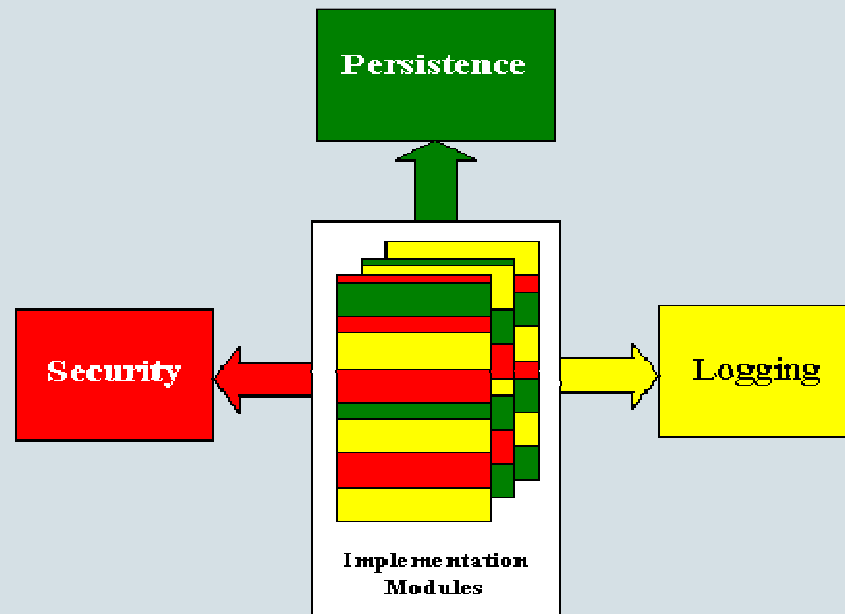
```
Class A
{
  method B
  method C
}
```

```
Class P: inherit A
{
  method B
  method C
  method Q
  method R
}
```

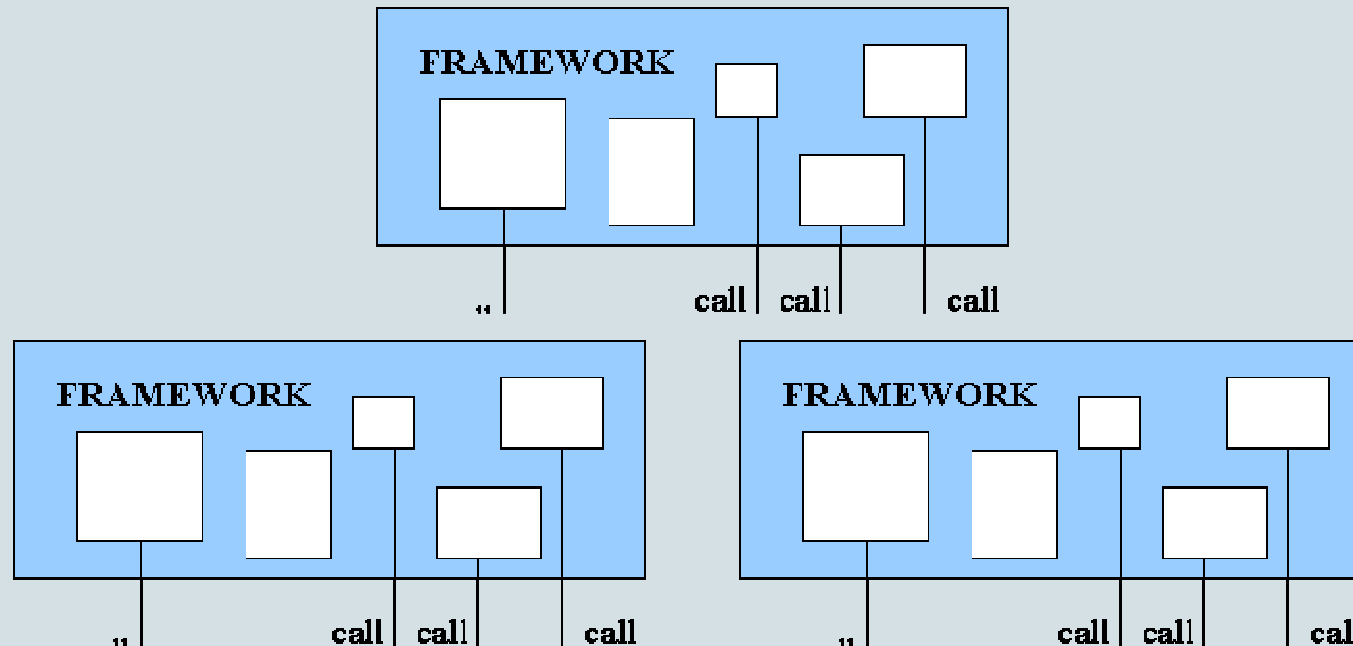
# Aspect Oriented Composition

Design & maintain concerns in isolation

Automatically construct implementation  
by 'weaving' concerns

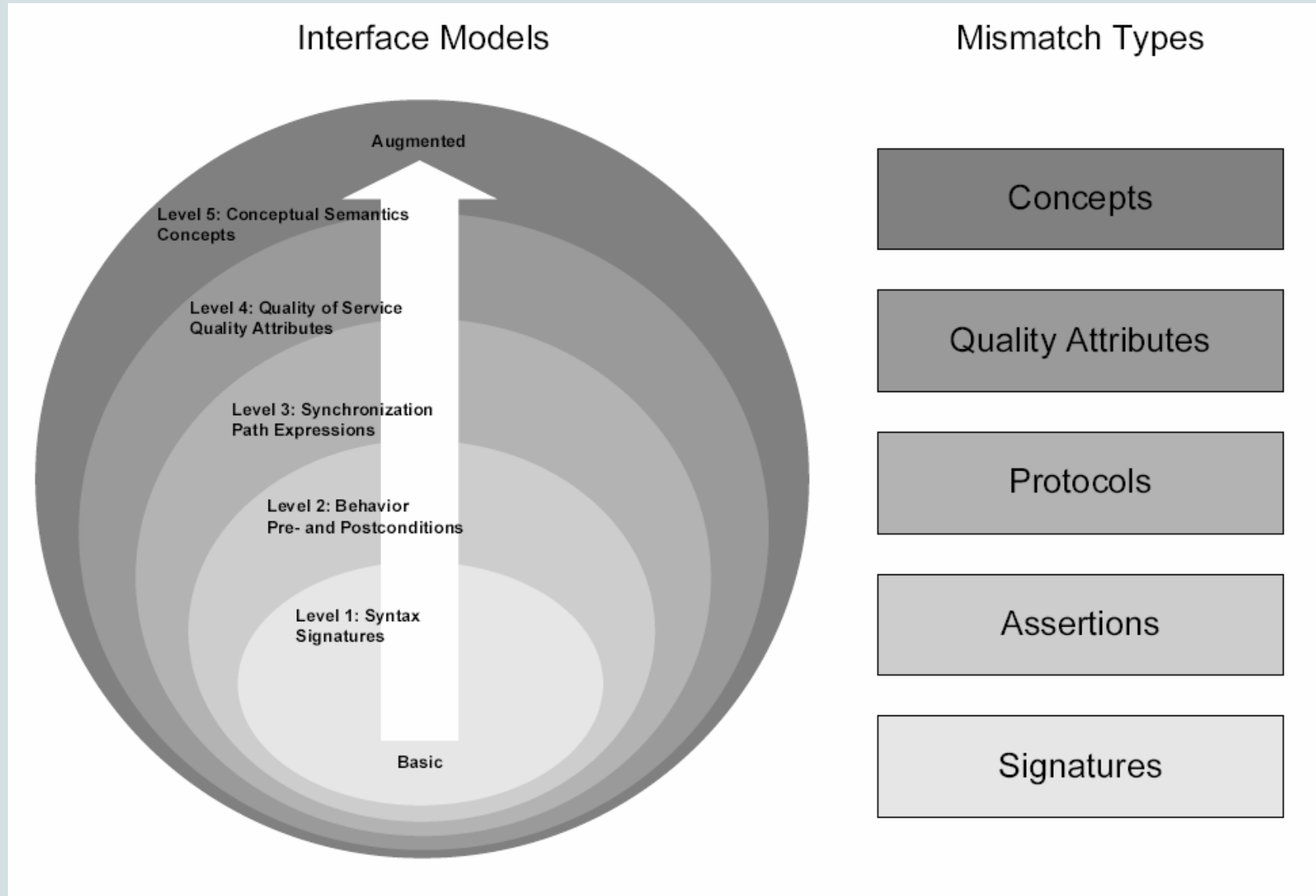


# Composing Frameworks



Scope of a framework is broader than component  
Typically not all functionality is used...

# Levels of (Mis)Match



# Composition of Features

# Features

According to

FODA: A prominent and user-visible aspect, quality or characteristic of a system.

ODM: A distinguishable characteristic of a system that is relevant to a stakeholder of the system

In mobile telephones:

- polyphonic ringtones
- SMS, MMS
- dual, tri-band,
- ...

In cars:

- airco
- power-steering
- remote key-lock
- ...

# Feature models

## Types of features

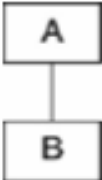

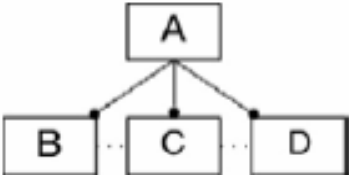
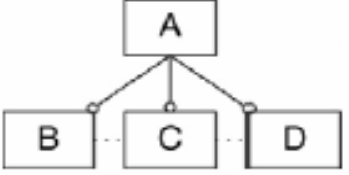
**Mandatory:** All systems must have it  
e.g. A car must have an engine

**Alternative:**

A system must have one out of multiple options  
e.g. Transmission may be manual or automatic

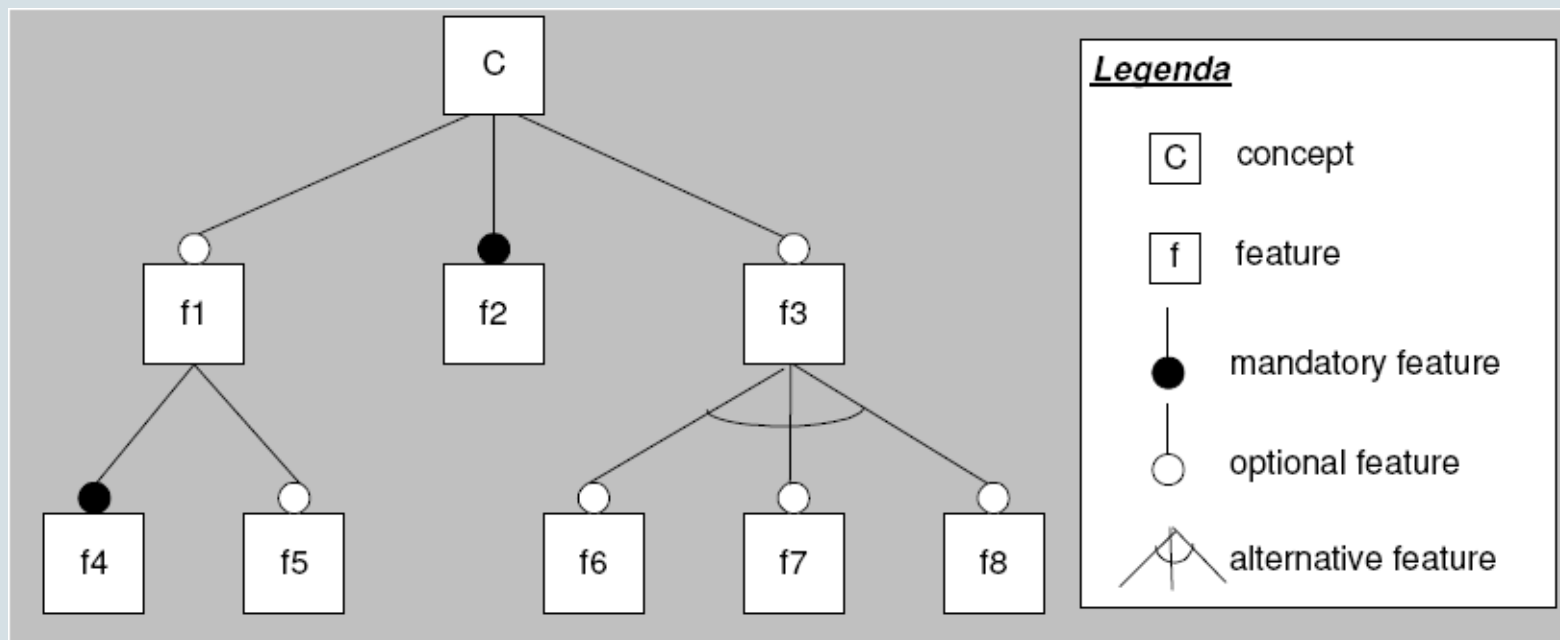
**Optional:** A system may have a feature  
e.g. A car may have air-conditioning

Table 1  
Explanation of feature diagram elements

Feature type	Graphical representation
<p><b>Mandatory</b> Mandatory feature B has to be included if its parent feature A is selected</p>	
<p><b>Optional</b> Optional feature B may be included if its parent feature A is selected.</p>	
<p><b>Alternative</b> Alternative features are organized in <i>alternative groups</i>. Exactly one feature of such a group B, C, D has to be selected if the group's parent feature A is selected.</p>	
<p><b>Or</b> Or features are organized in <i>or groups</i>. At least one feature of such a group B, C, D has to be selected if the group's parent feature A is selected.</p>	

# Feature-oriented composition

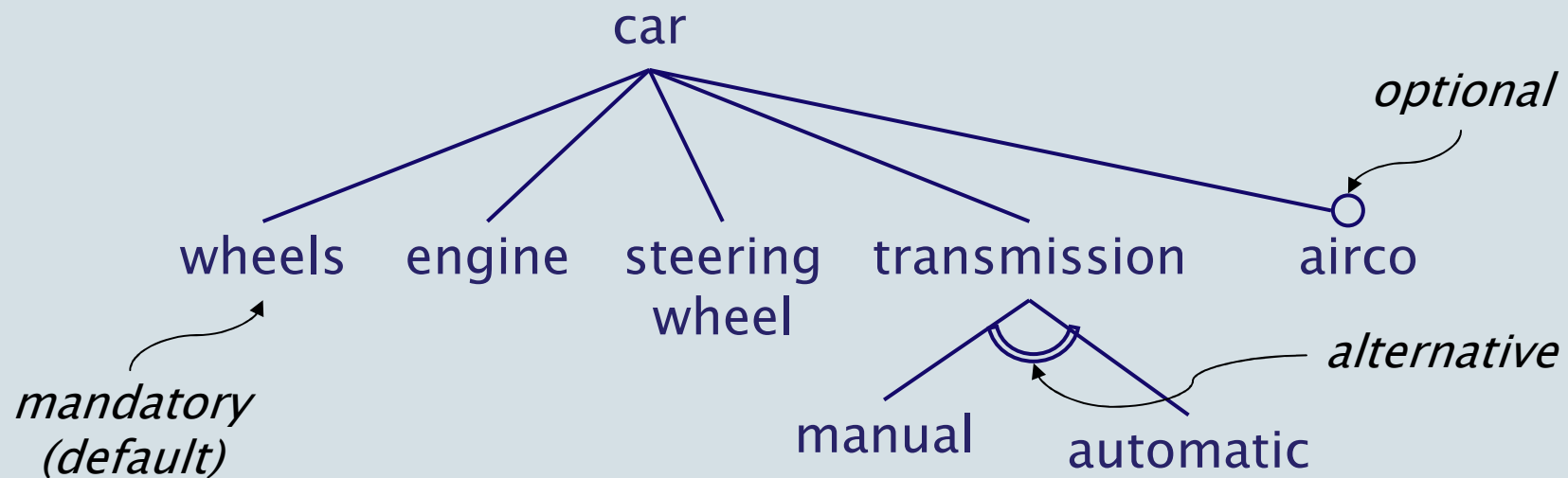
- Strongly driven by feature design and architecture



# Feature Diagram

A hierarchical decomposition of features.

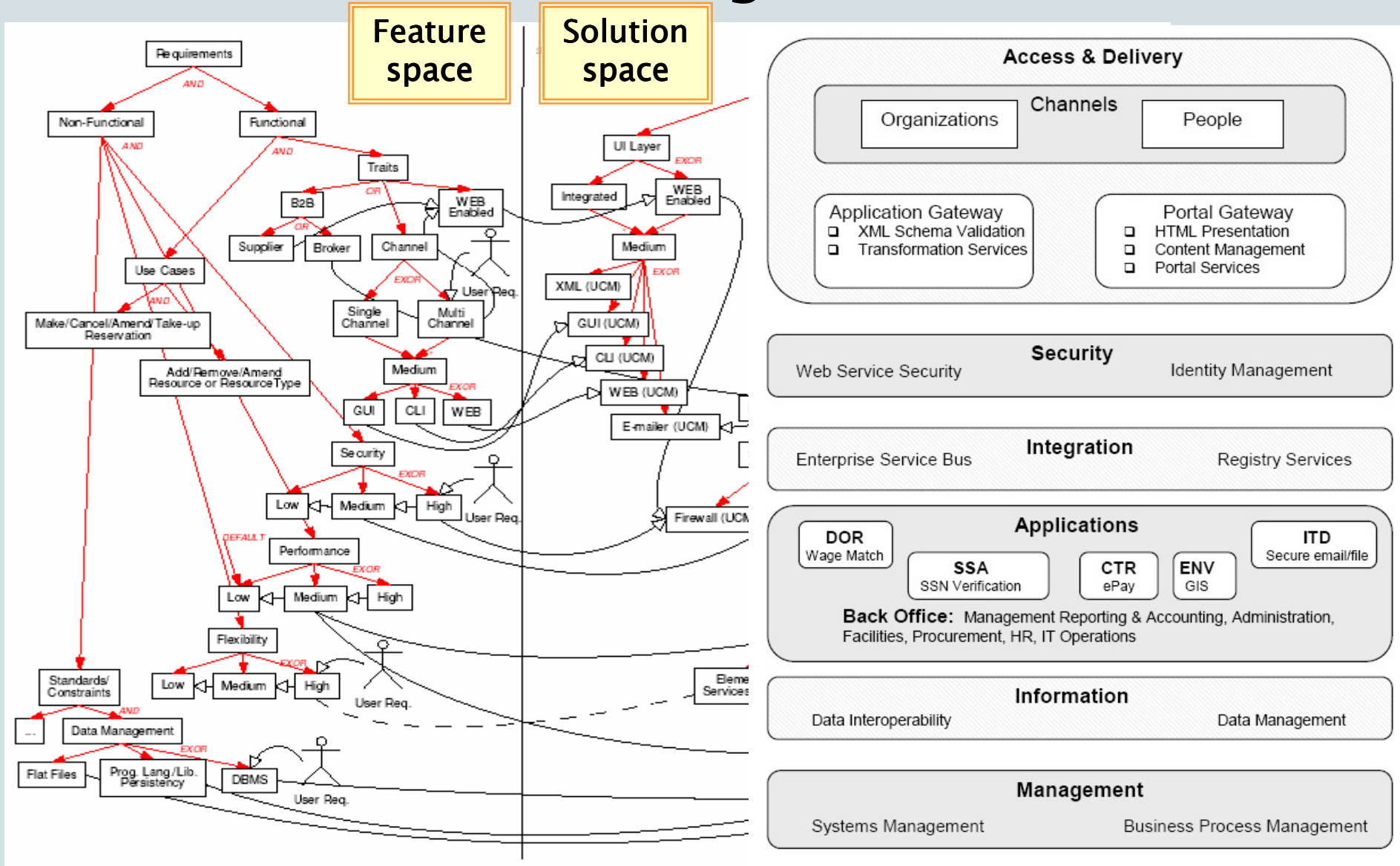
A concept higher in the tree *consists of* its children



Additional annotations that may be used in the feature diagram:

- mutually exclusive features
- rationale for choosing between alternatives
- composition rules: airco may be used if horsepower > 100

# Feature Solution Diagrams



# Oefenen

- Create a feature model for mobile telephones
- Zie bijvoorbeeld [newmobile.nl](http://newmobile.nl) voor inspiration

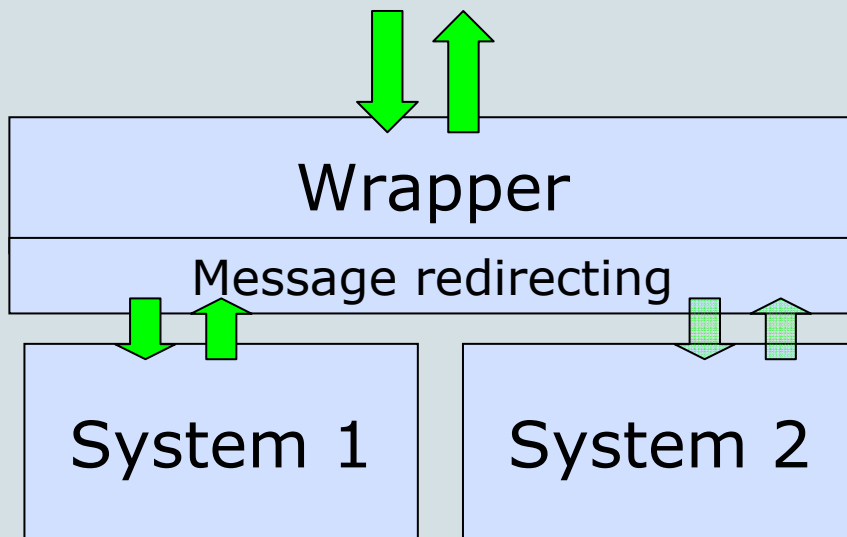
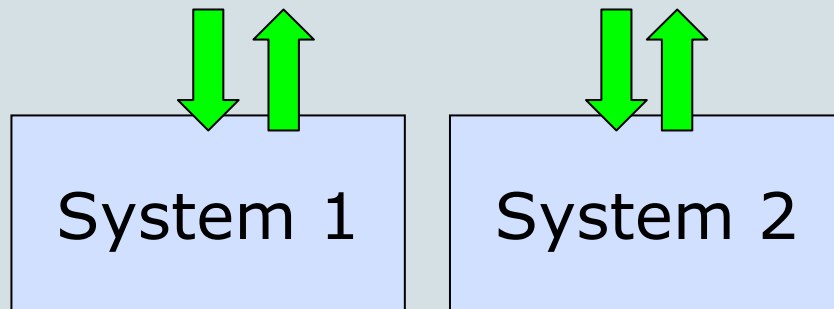
# Features dd. 24 oct 2007

- **Vormgeving**
- Clamshell
- Flip model
- Horloge
- Monoblock
- Slider
- Swivel
- **Network**
- Dual band
- Quad band
- Single band
- Tri band
- **Data network**
- EDGE
- GPRS
- HSCSD
- HSDPA
- HSUPA
- UMTS
- WCDMA
- **Beeldscherm**
- Extern beeldscherm
- Kleuren beeldscherm
- Landscape modus
- Monochroom beeldscherm
- Roterend beeldscherm
- Scherm met spiegeleffect
- Touchscreen
- **Externe vormgeving**
- Automatisch openen
- Bedieningstoets
- Camera
- Externe bediening MP3 speler
- Geïntegreerde antenne
- Haptic feedback
- Internet sneltoets
- Lichtgevoelige toetsenbordverlichting
- Secundaire camera
- Stof-/spatwaterdicht
- Trilfunctie
- Verwisselbare covers
- Volledig toetsenbord
- Zaklantaarn
- **Interne vormgeving**
- Dynamisch lettertypeformaat
- Individualiseerbare profielen
- Screensavers
- Thema's
- Wallpapers
- **Geheugen**
- Extern meegeleverd geheugen
- Geheugenkaart slot
- Intern geheugen
- Interne harddisk
- **Audio**
- Beltoon composer
- FM-radio
- Handsfree functie
- Monofone beltonen
- Polyfone beltonen
- Stereo luidspreker
- Text-to-speech
- Voice command
- Voice dialing
- Voice memo
- **Messaging**
- Chat functie
- E-mail
- EMS
- Instant messaging
- MMS
- Nokia Smart Messaging
- Predictive Text Input (T9/Ezi/iTap)
- Quickshare
- RSS lezer
- SMS
- Video MMS
- **Connectivity**
- 3G ondersteuning
- A-GPS
- Audio jack plug
- Bluetooth
- DVB-H Mobile TV
- GPS ontvanger
- Infrarood
- Internet browser
- Mini-USB poort
- Modem
- Push-to-Talk
- Push-to-View
- Synchronisatie
- T-DMB Mobile TV
- TCP/IP ondersteuning
- UPnP ondersteuning
- Video-out
- Wallet & WIM
- WAP
- WAP browser
- Wireless Lan (WLAN/Wifi)
- Aanwezige programma's
- Agenda
- Alarmklok
- Automatisch toestel in-/uitschakelen
- Automatische toetsenbordvergrendeling
- Automatische volumeregeling
- Bioritme meter
- Businesskaart scanner
- Calculator
- Calorieën meter
- Chronometer
- Contactpersonen
- Digitaal kompas
- Fotoalbum
- Fototelefoonboek
- Geluidsmeter
- Handschrift herkenning
- Herinneringen
- Image editor
- Ingebouwd navigatiesysteem
- Ingebouwde spellen (Games)
- Java ondersteuning
- Kalendernotities
- Movie editor
- MP3 speler
- Remote printing
- Stopwatch & timer
- Takenlijst
- Telefoonboek met beller groepen
- Thermometer
- Tijdsnotatie of Klok
- Valuta omzetter
- Video afspeelfunctie
- Windows Media player
- Overige ondersteuning
- C++ ondersteuning
- Office ondersteuning
- Ondersteuning PDF bestanden
- YouTube ondersteuning
- Besturingssysteem
- Hiptop OS

# Composition of Systems

- In IS this means: interoperability & collaboration
- In ES this means: integration into a larger system

# Composition of Systems



IS type of integration

What is needed?

- Interoperability of Networks, Operating Systems & Programming Languages
- Integration of data models, data and queries
- Business Processes

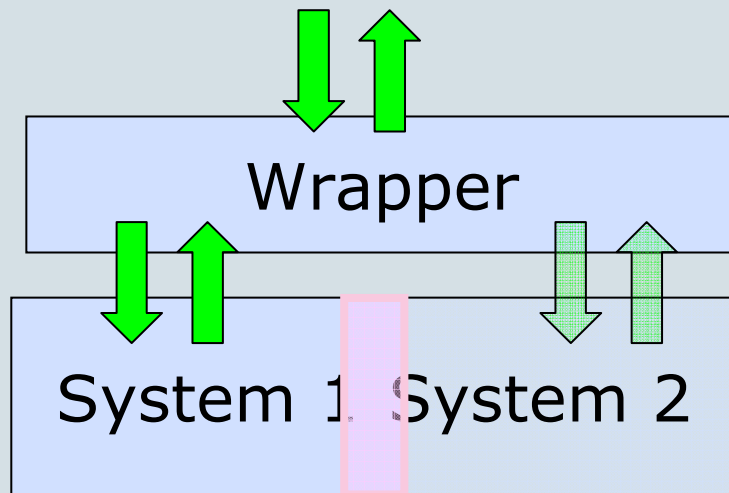
Works well if:

- Architecture is layered:  
UI-, application-, business- and data-management layers

Common Solution:

- Bus with Asynchronous Messaging

# Composition of Systems



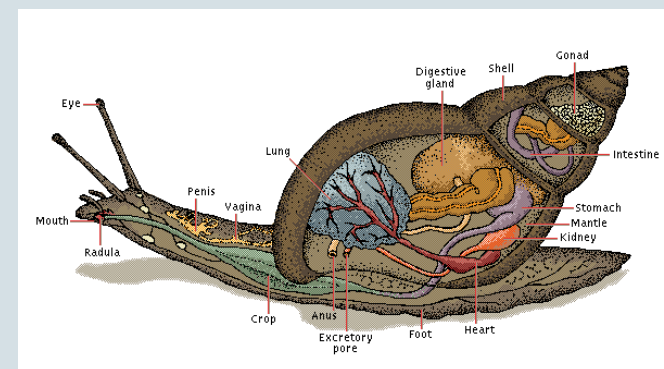
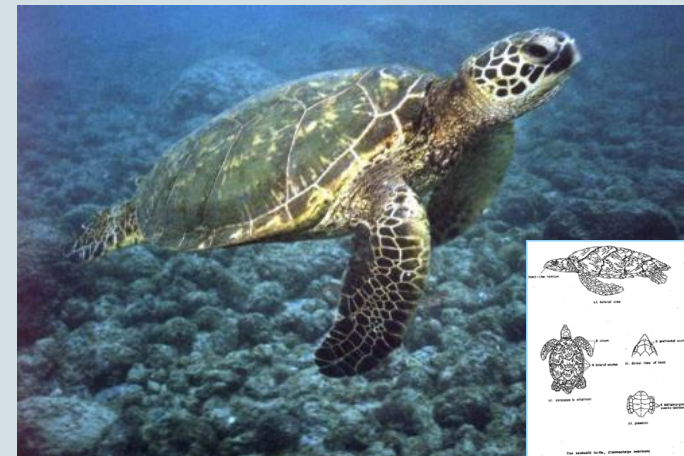
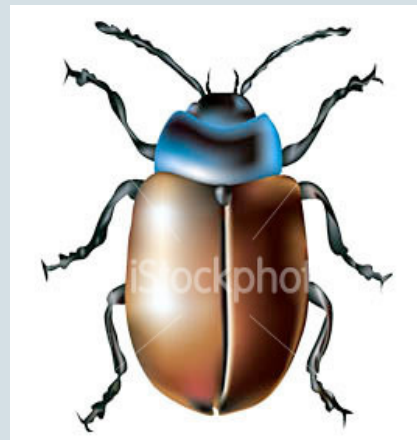
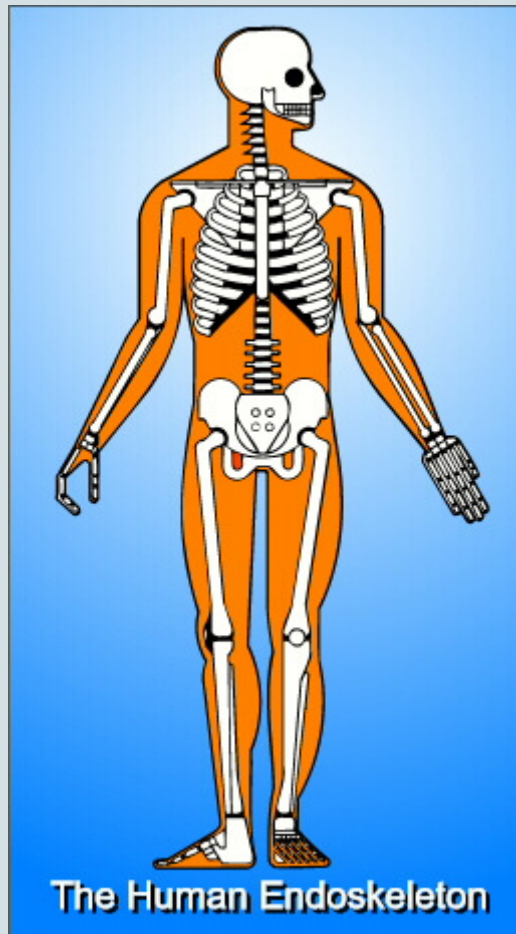
Not always ideal complement

overlapping  
functionality

# Locus of Composition

- Who knows what to compose?
- Where does this information reside in the system?

# Endoskeletons and Exoskeletons



Strength, Protection, Muscle attachment

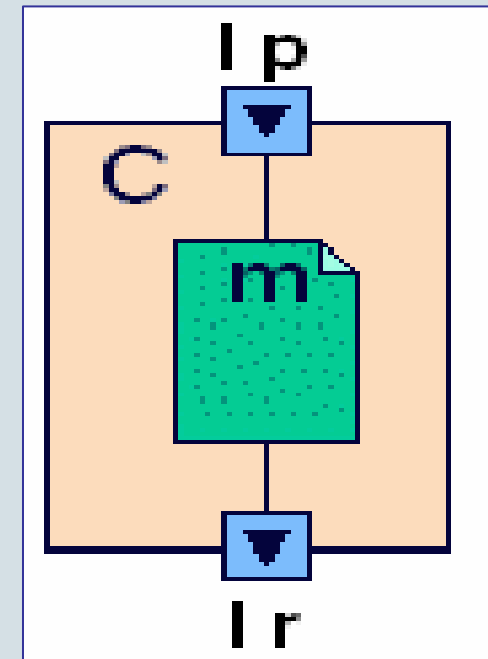
# KOALA Example

- File i.id (hand written)

```
interface I{  
    void f(int x);  
}
```

- File c.cd (hand written)

```
component C {  
    provides I p;  
    requires I r;  
    contains module m;  
    connects p = m;  
             m = r;  
}
```



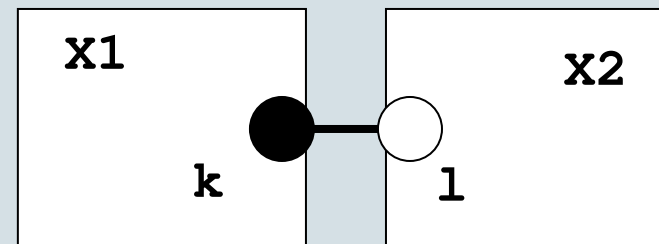
## Binding of Components

```
Interface Iv  
{ ... }
```

```
Component P1  
{ provide Iv k }
```

```
Component P2  
{ require Iv l }
```

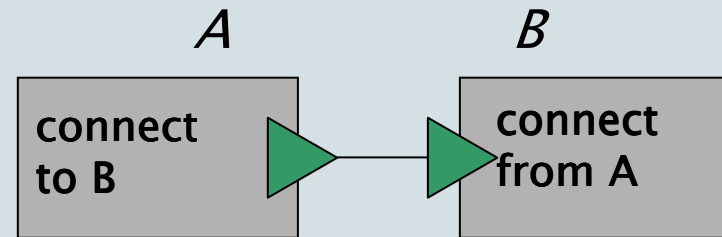
```
Inst X1:P1, X2:P2  
Bind X1.k - X2.l
```



- Exogenous or Endogenous?
- What is the relation between composition and containment? Where is it in the text?

# Endogenous versus Exogenous composition

*Endogenous composition:*  
composition is built into the items that are composed



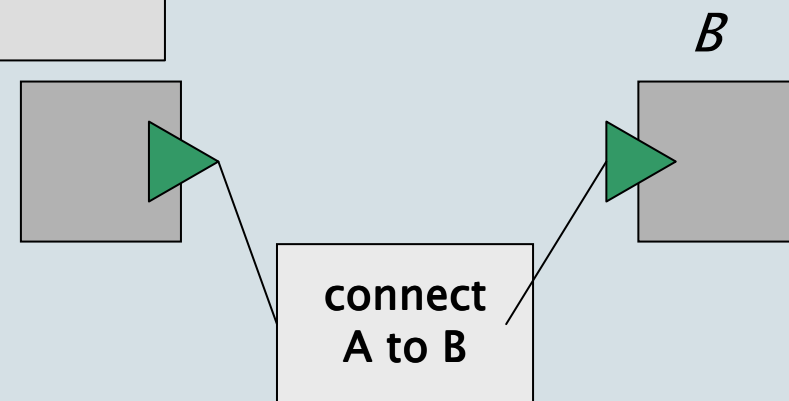
e.g. method calls in OO languages

```

Class A
{
  method Q
  {
    B.m()
  }
}
    
```

*Exogenous composition:*  
composition is defined outside of the items that are composed.

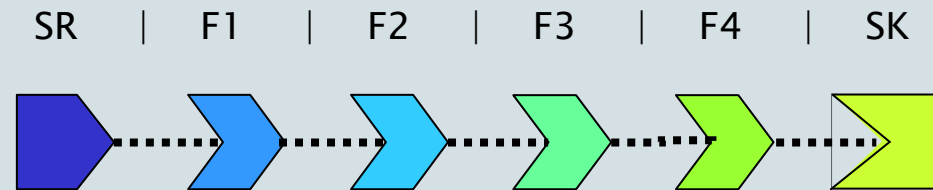
- localizes changes in binding
- reduces dependencies between application components



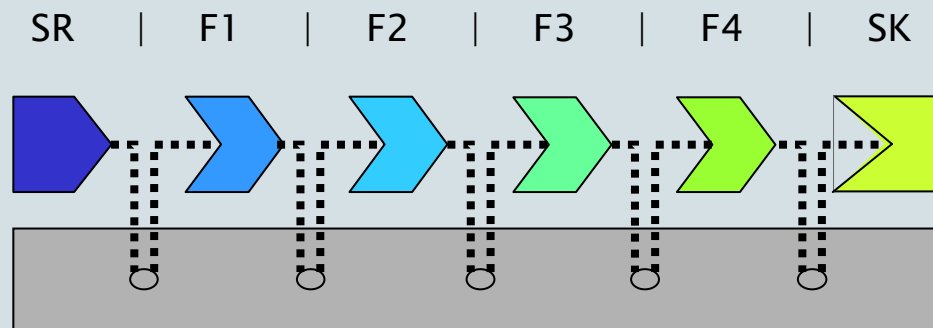
Benefits: consider changes in A or B or in the connections.

# Pipe and Filter

Binding looks like:



But is actually like:



The infrastructure does the plumbing:  
it directs outputs to the right inputs.  
Neighbours do not know of each other.

# Moment of Composition

- Binding time

## Component Binding (noun)

When a component X uses/knows information about another component Y, this is a dependency of X on Y. We say that X is bound to Y.

### Binding (verb)

The act of constructing a binding

# Composition/Binding time

Binding time: the stage in development that a component X obtains a binding to another component Y.

## Development-time

- Integrate source text files and build instructions

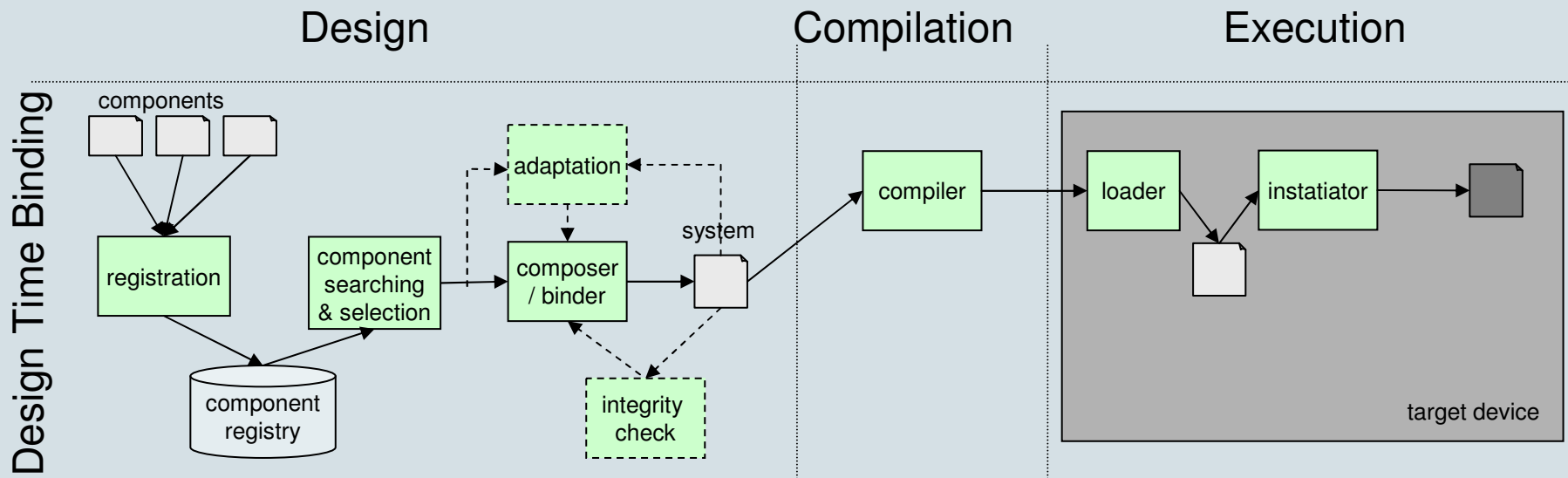
## Compile / Link-time

- Integrate source or object-code into executable

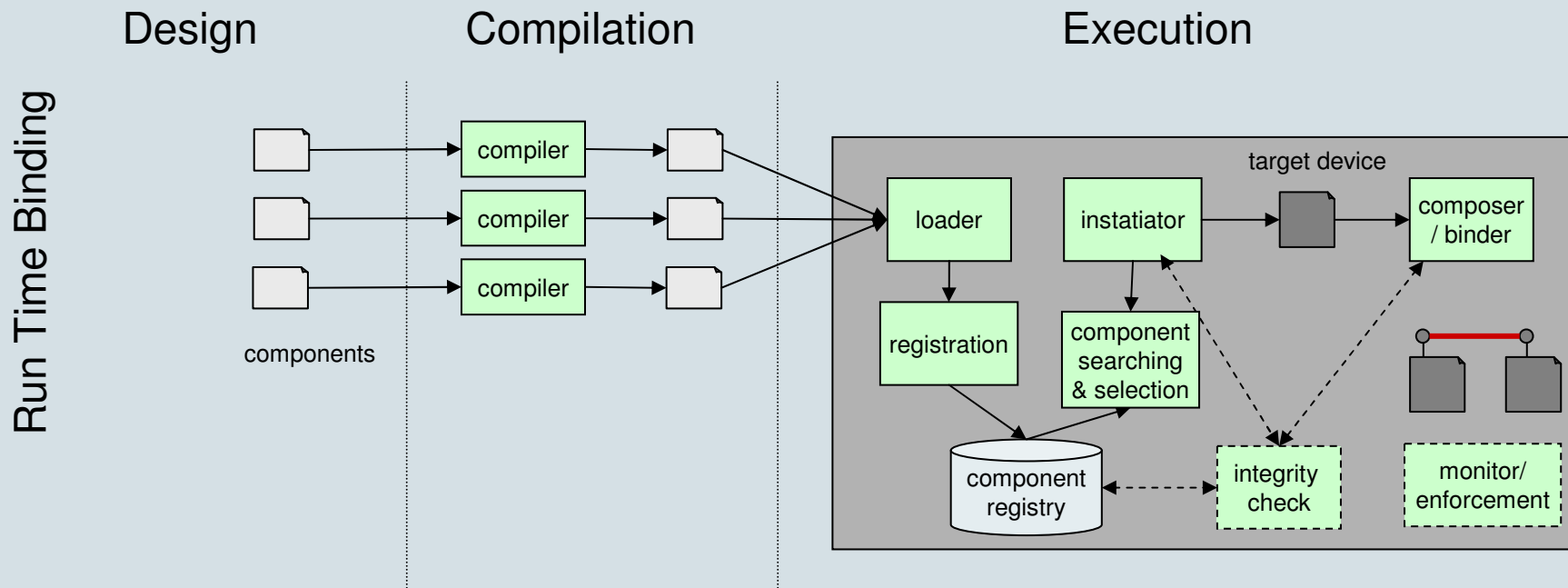
## Run-time

- Integrate executable components in a running system

# Design Time Binding



# Load-Time Binding



Boodschap van deze diagrammen:

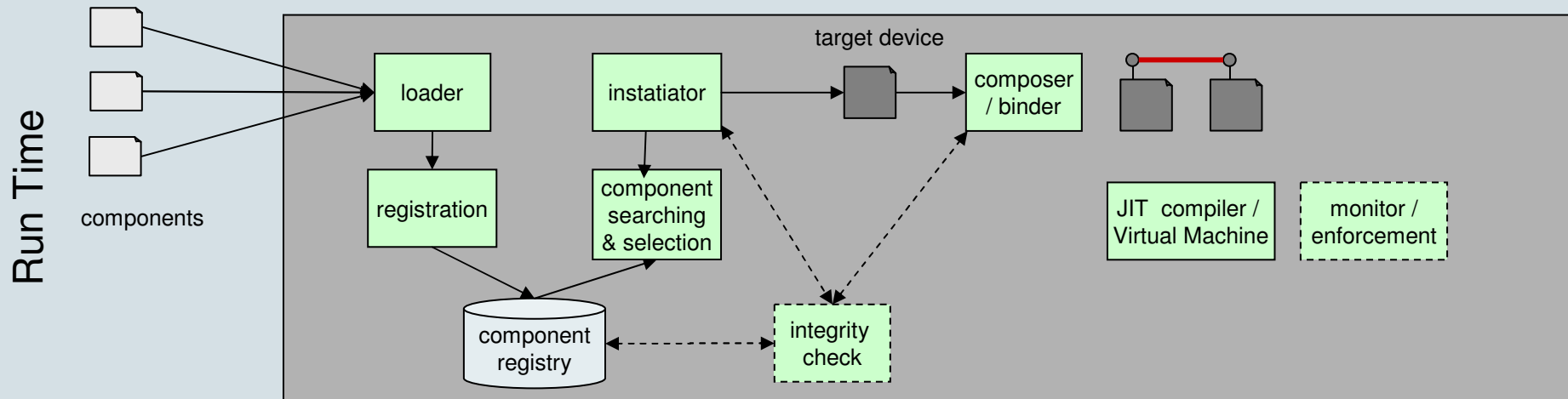
Dezelfde taken worden uitgevoerd, maar in een andere volgorde.

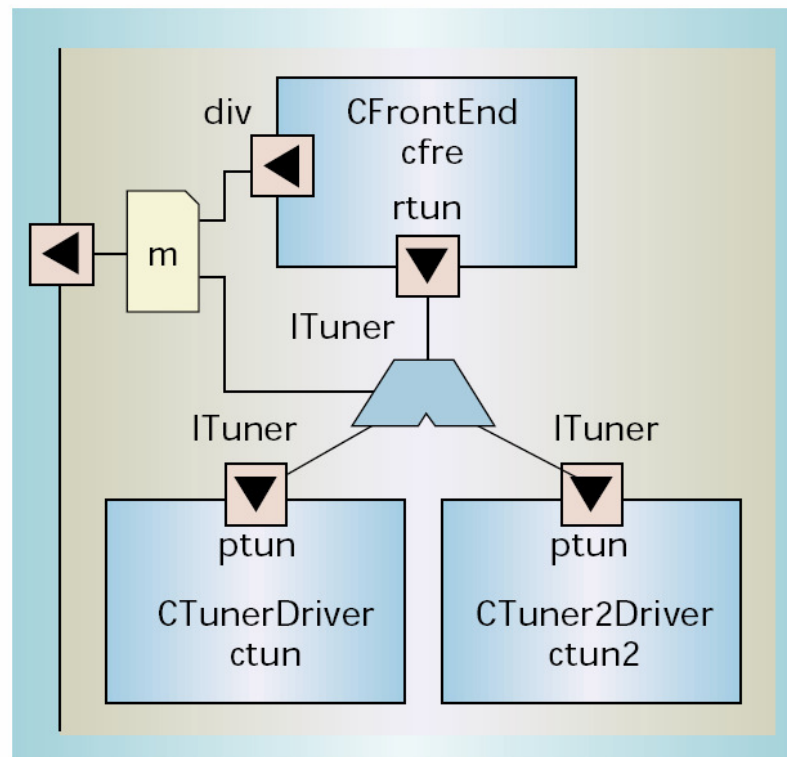
# Run-time decomposition

- Not so long ago, the windows operating system crashed when you unplugged the mouse
- (un)pluggability needs to be designed into a system esp. the interaction protocols



# Run Time met virtual machine / JIT compiler

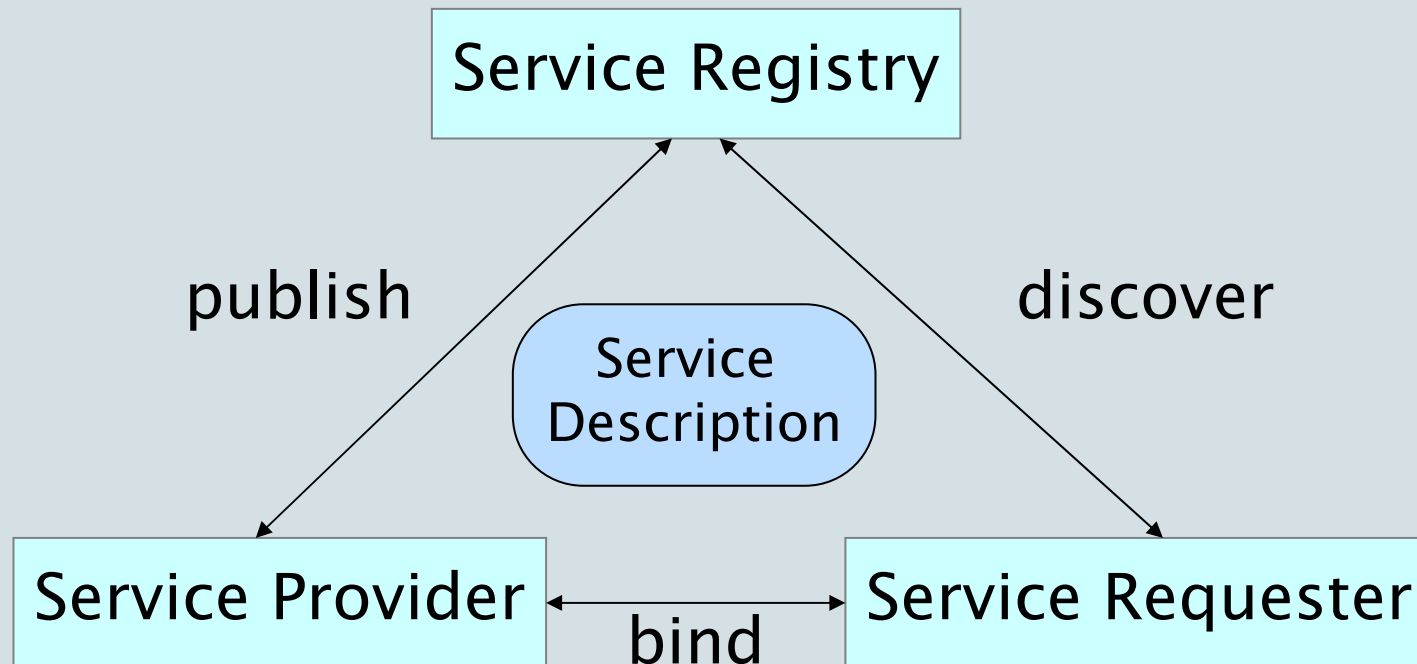




*Figure 5. Diversity interfaces and switches. CFrontEnd has a diversity interface that is defined in module m. The switch selects between two drivers and is controlled by module m. Module m can define both drivers in terms of the diversity of the compound component.*

- Bindings that are created at design time, may still cater for run-time selection of collaborations ...

# Binding Pattern

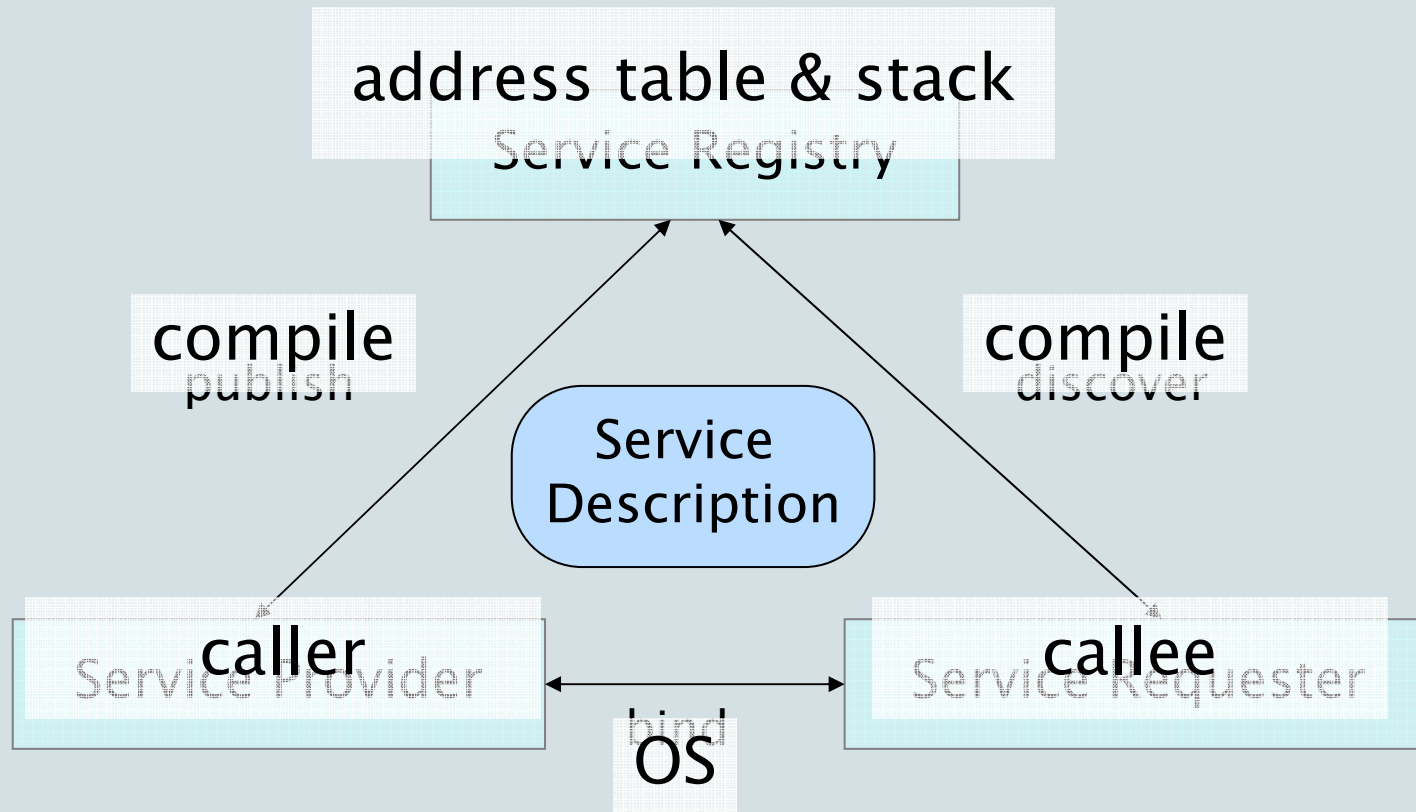


Often the framework (e.g. run-time infrastructure) plays role of registry

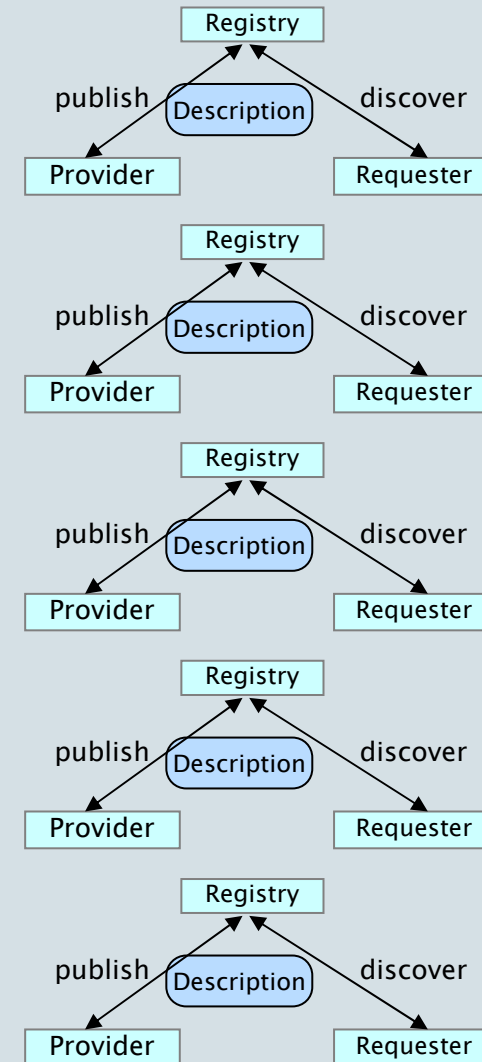
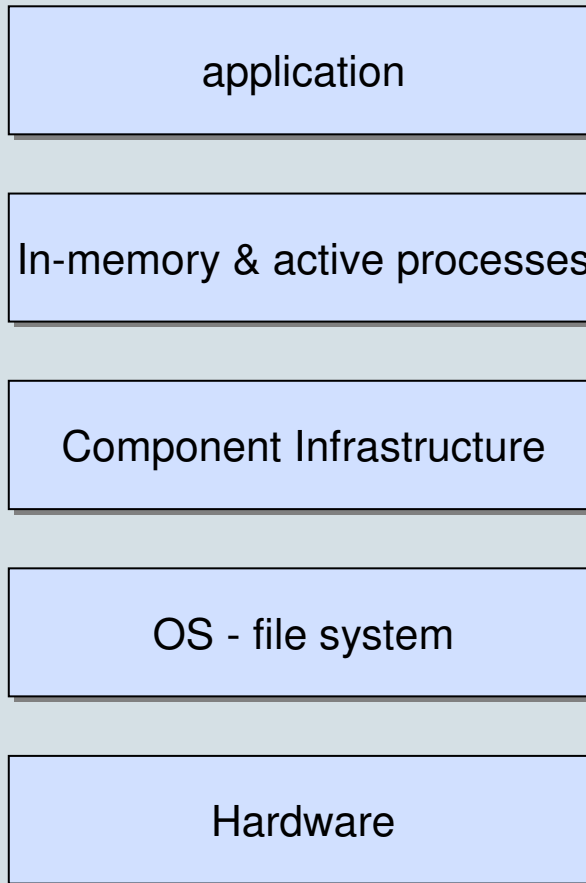
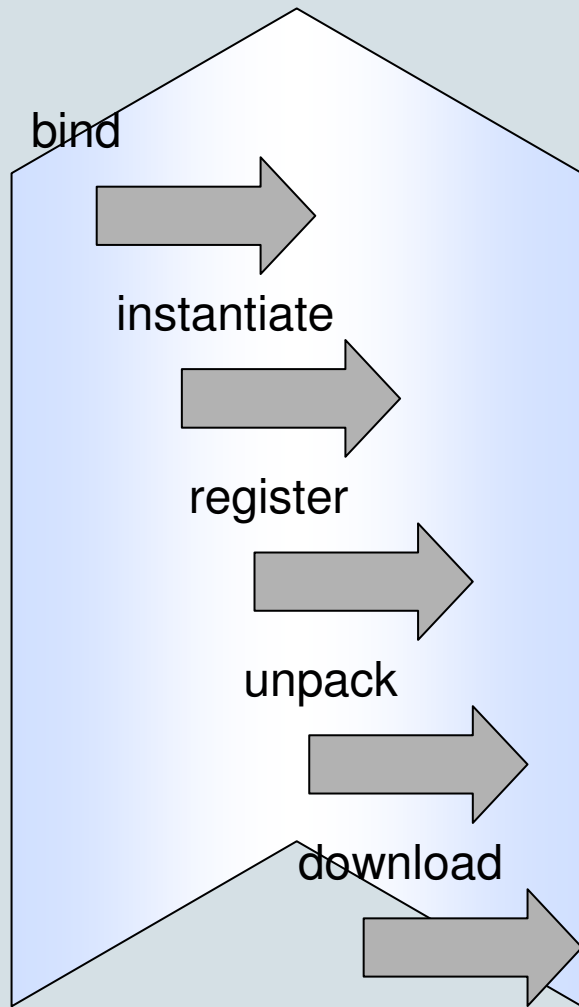
# How does a system know where to 'return' the result of a function call?

- Answer follows later ...

# Binding Pattern

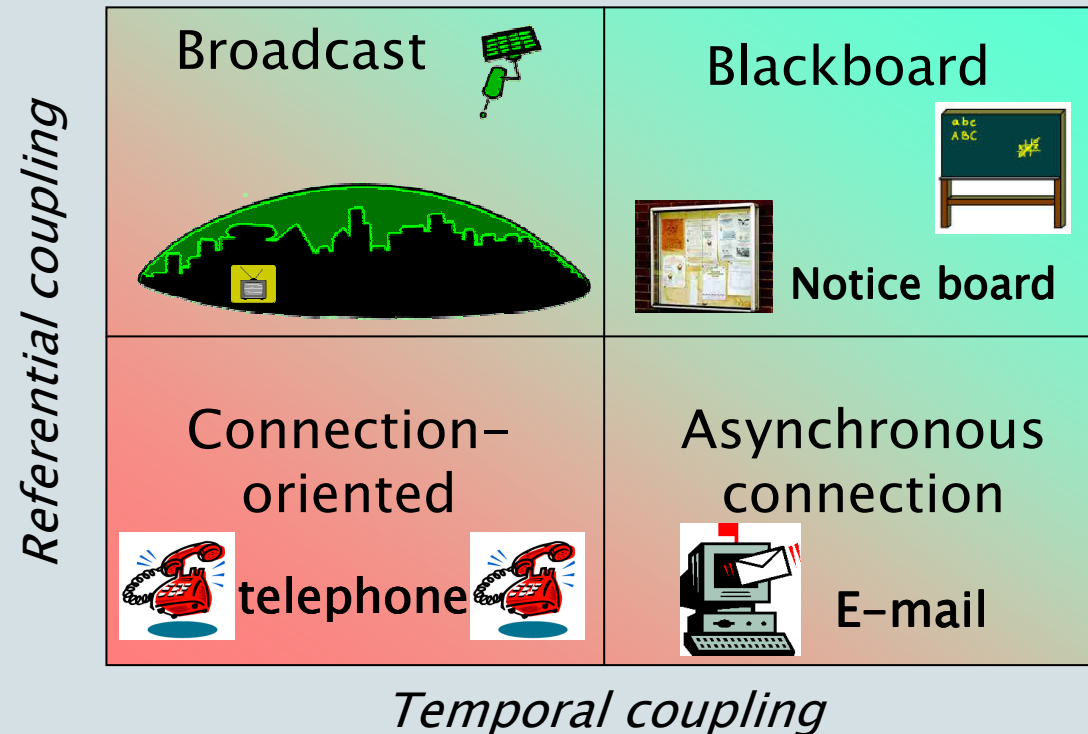


# Binding at different levels in the system



# Interaction Style and Binding

Interaction styles bring inherent type of binding



Referential coupling : sender has reference to receiver name/address

Temporal coupling: sender and receiver synchronize in time

# Conclusions

- Different types of composition exist:
  - At different abstraction levels (programming, module)
  - Endogenous and exogenous
  - Depending on the interaction style
  - Aspects of composition: binding & encapsulating
- Flexibility (late binding) does not come for free.
  - Facilities are needed in the infrastructure

# References

- G. Bracha and W. Cook. Mixin-based inheritance. In Proc. of the Joint ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications and the European Conference on Object-Oriented Programming, October 1990.