SOLID SERVICES

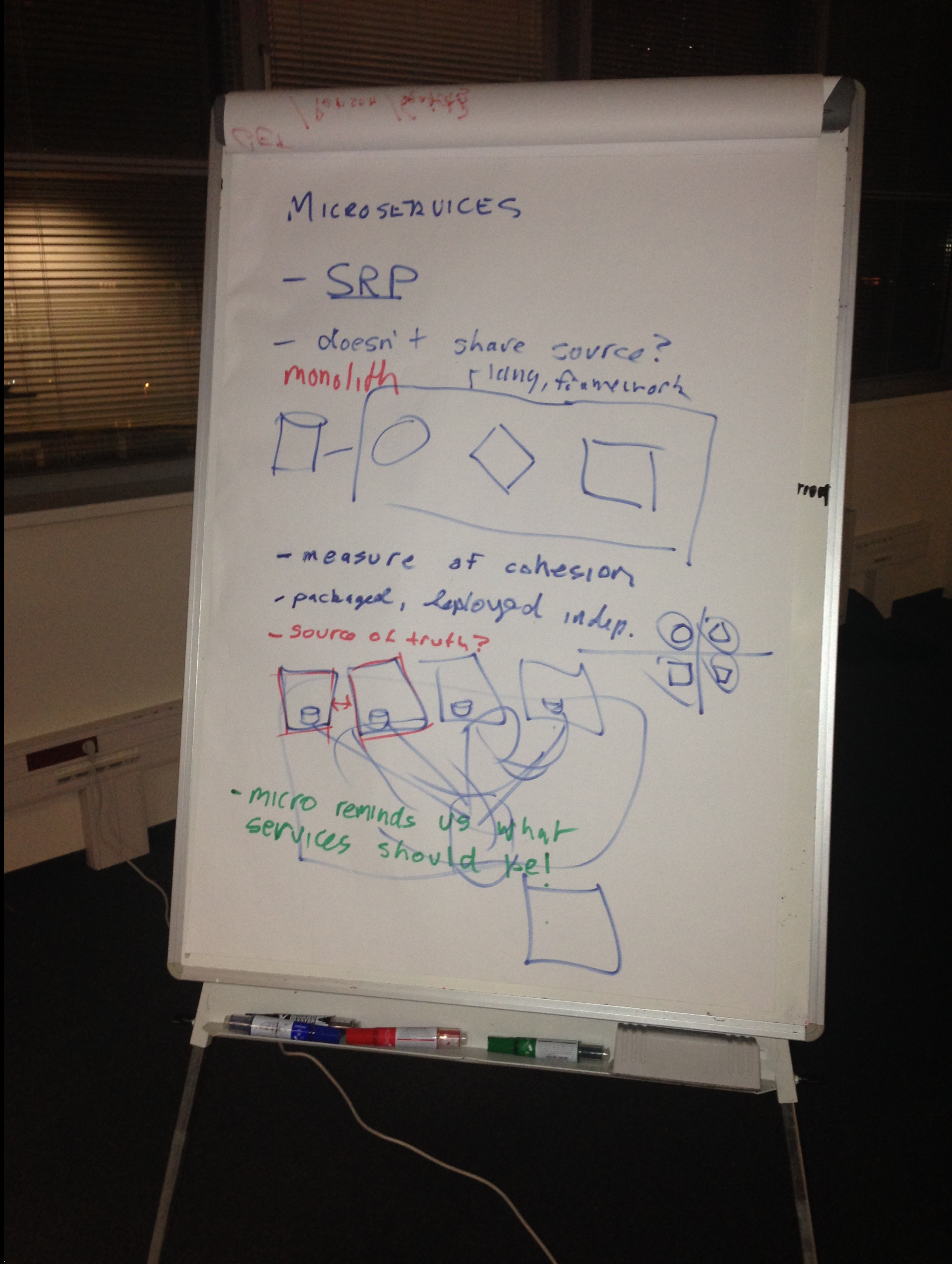
Ondřej Krajíček @hedragon @ysoftdevs

A Little Disclaimer

- Today, I am trying to give you a perspective or - say - a point of view.
- What I am presenting today are my own opinions.
- But I am not presenting a new invention. And if I have learnt anything, it was from my cooperation with my **colleagues at Y Soft**.
  Hey guys... :-).
- I am going to sometimes refer to buzzwords. I will try to make this explicit.
- No services were harmed during preparation of this talk.

In a Galaxy Far Far Away...

Geecon Prague 2014
Open Spaces with Bruce Eckel



Architecture (Latin architectura, after the Greek ἀρχιτέκτων – arkhitekton – from ἀρχι- "chief" and τέκτων "builder, carpenter, mason") is both the process and the product of planning, designing, and constructing buildings and other physical structures.

Why do we have **software** architecture?

Software Architecture
gives answers to the most expensive questions.

Software Architecture
is a decomposition of software products
into systems,
based on their responsibilities
and interactions.

Architecture envelops **software design.**

Software Architecture
is the servant of high-priority stakeholder values.
Is as **simple** as possible, but **not simpler**.
Is designed to be **replaceable**.
(Tom Gilb)

Functional vs. Non-Functional Requirements

~~Functional vs. Non-Functional Requirements~~

Functions (Features) **and** Qualities (Quality Requirements)

Stakeholders Values

Product Qualities

System Qualities

Microservices *Architectural Style*

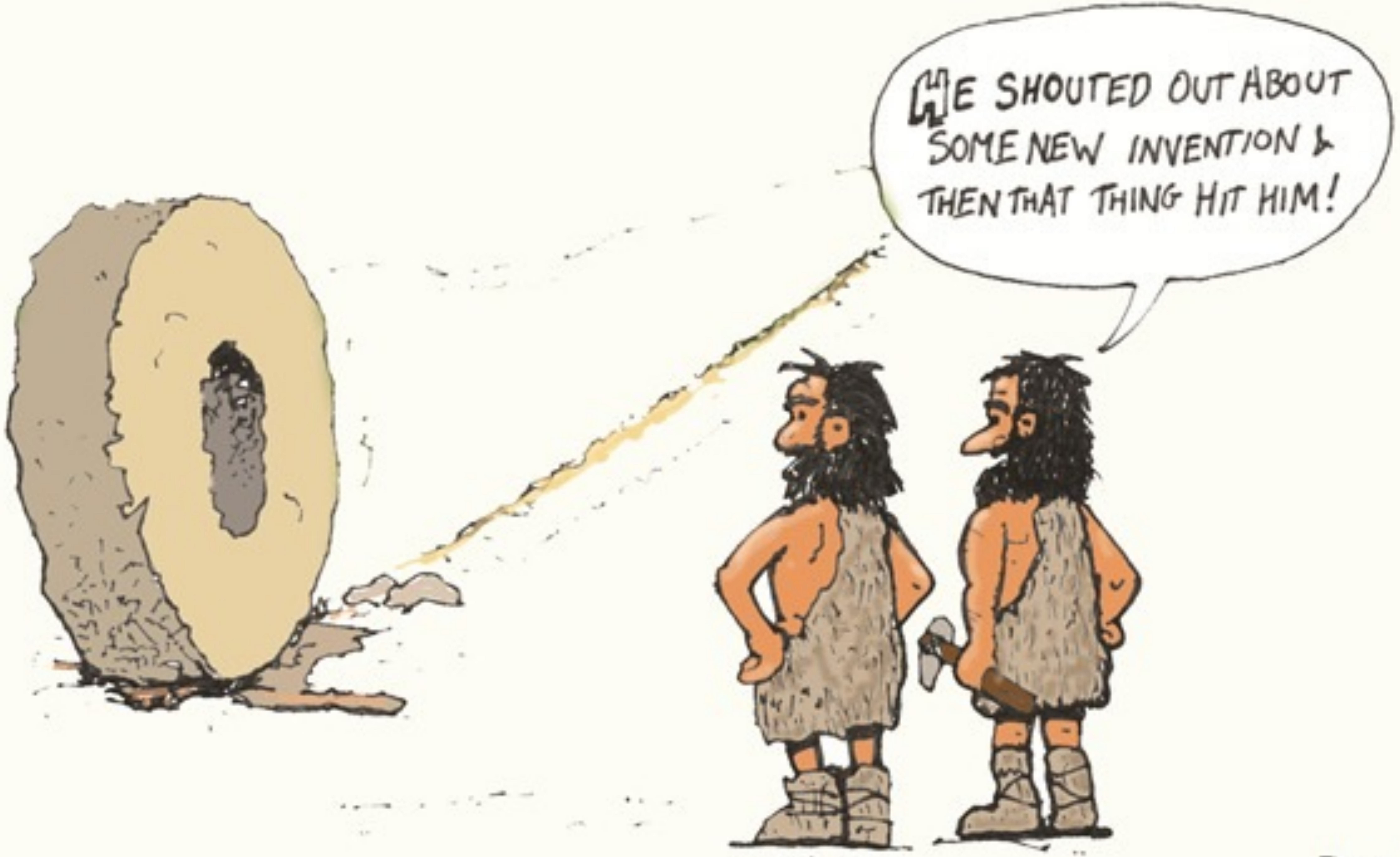
What is a SERVICE?

Microservices (2014)

*"In short, the microservice architectural style is an approach to developing a **single application** as a **suite of small services**, each running in its **own process** and **communicating** with lightweight mechanisms, often an HTTP resource API. These services are built around **business capabilities** and independently deployable by fully **automated deployment** machinery. There is a bare mininum of centralized management of these services, which may be written in **different programming languages** and use different data storage technologies."*

Martin Fowler

*Microservices (**2014**)*	*RPC (**1991**)*
• Single Service per Process	• Single Service per Process
• Lightweight Network Communication and Interoperability	• Encapsulated Network Communication and Interoperability
• Based on Business Concepts	• Based on Business Concepts
• Programming Language Agnostic	• Programming Language Agnostic
• Synchronous / Asynchronous	• Synchronous / Asynchronous
• Independent from others	• Independent from others


HE SHOUTED OUT ABOUT SOME NEW INVENTION & THEN THAT THING HIT HIM!

DeepFat '09

Microservices vs. Monoliths

By popular demand, the new technology is replacing the old, obsolete one.

Deja-Vu

Tight vs. Loose Coupling

- Local vs. Remote Procedure Call

Pooling Resources

- Database Connection Pools reused by several services

Memory and Latency Conservation

- Micro-Kernel (Mach) vs. Modular Kernel (WinNT) vs. Monolith (Linux)

It all has been invented already (compare microservices to CSPs, WSRF, Globus Toolkit, OGSI, Agents, Actors, Data Driven Systems, Message Passing Systems, P2P Networks, etc.)

Let's Apply some Engineering...

SOLID

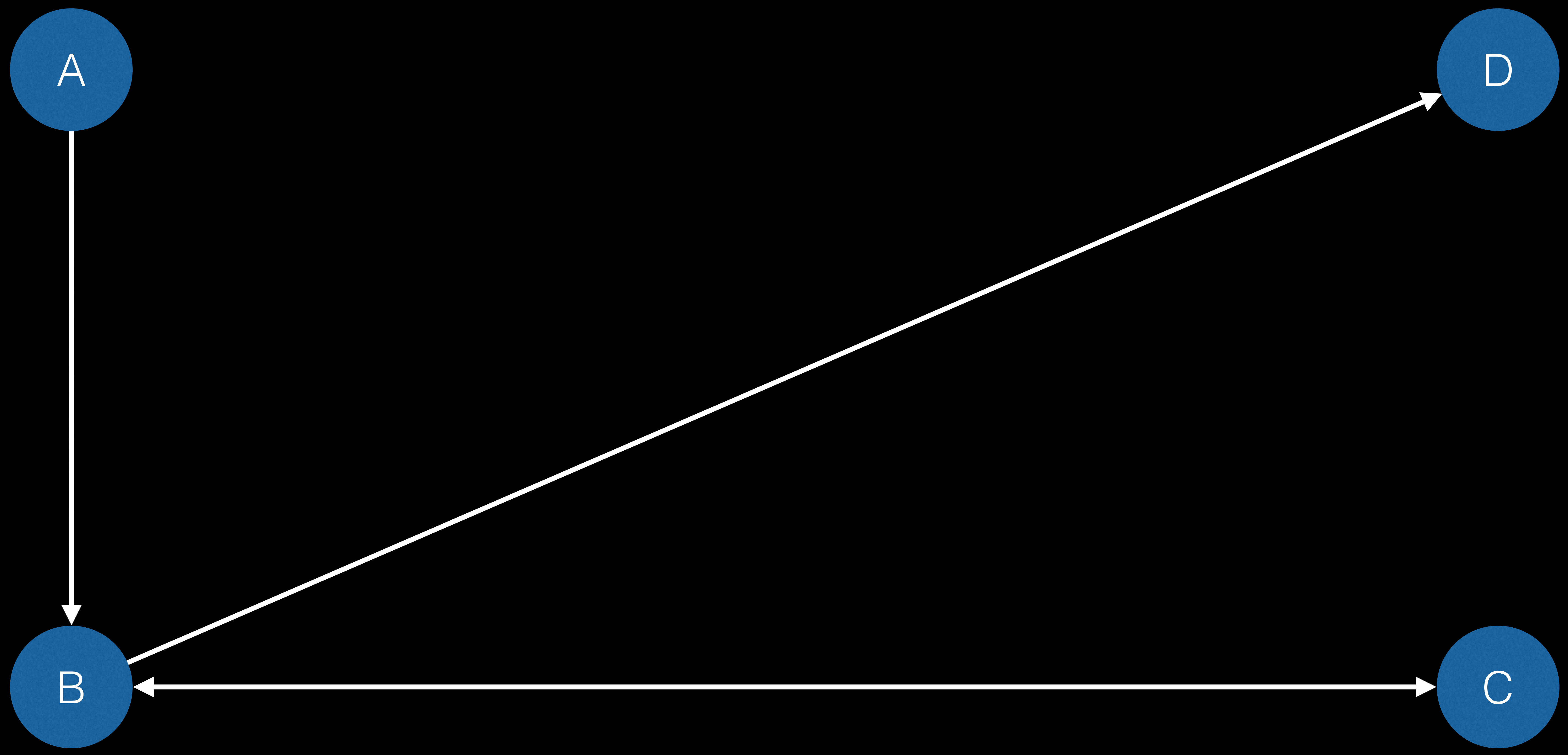
- Single Responsibility Principle
- Open for Extension / Closed for Modification Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion

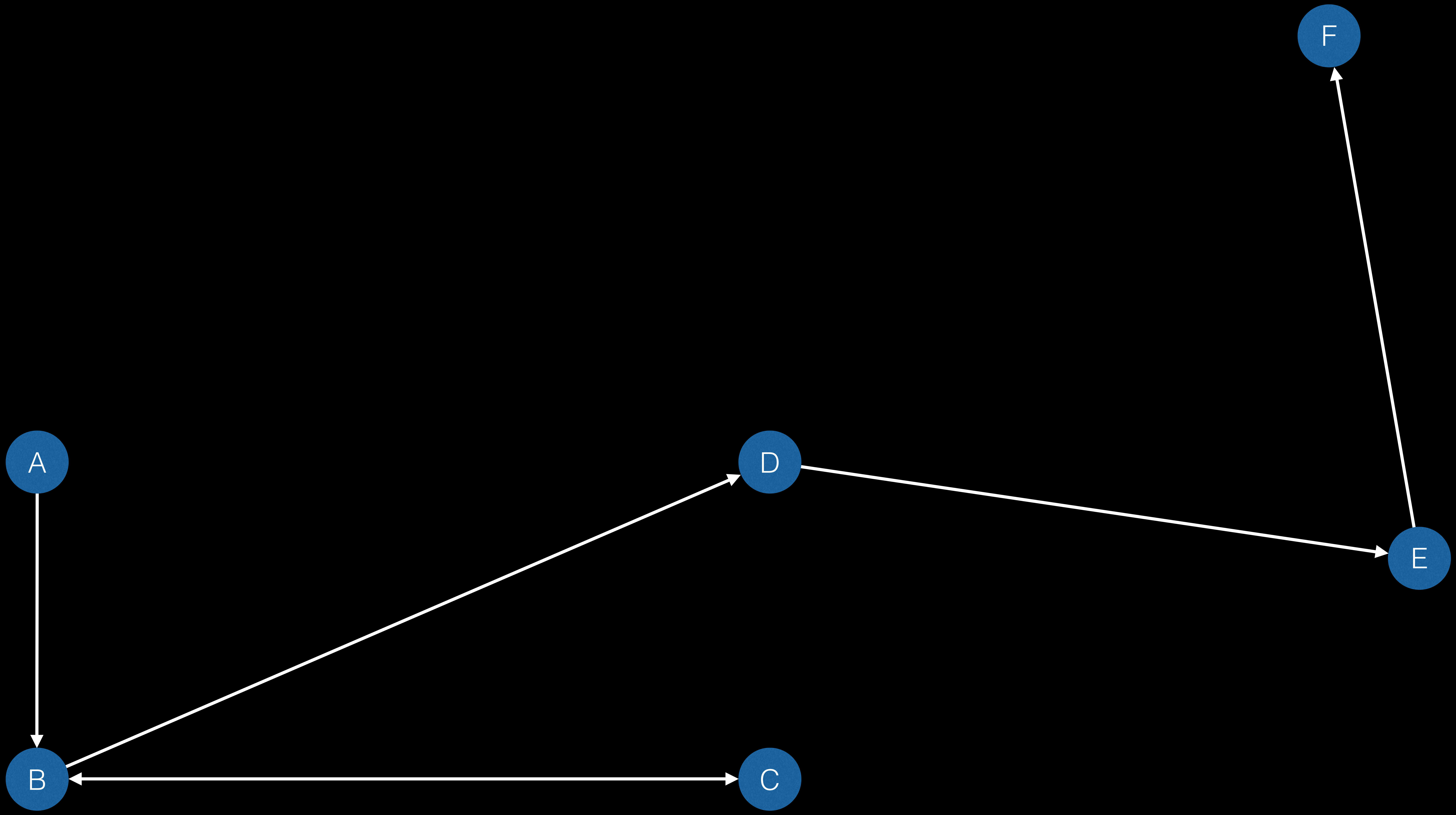
How can we apply principles from structured / object oriented design to (micro)services?

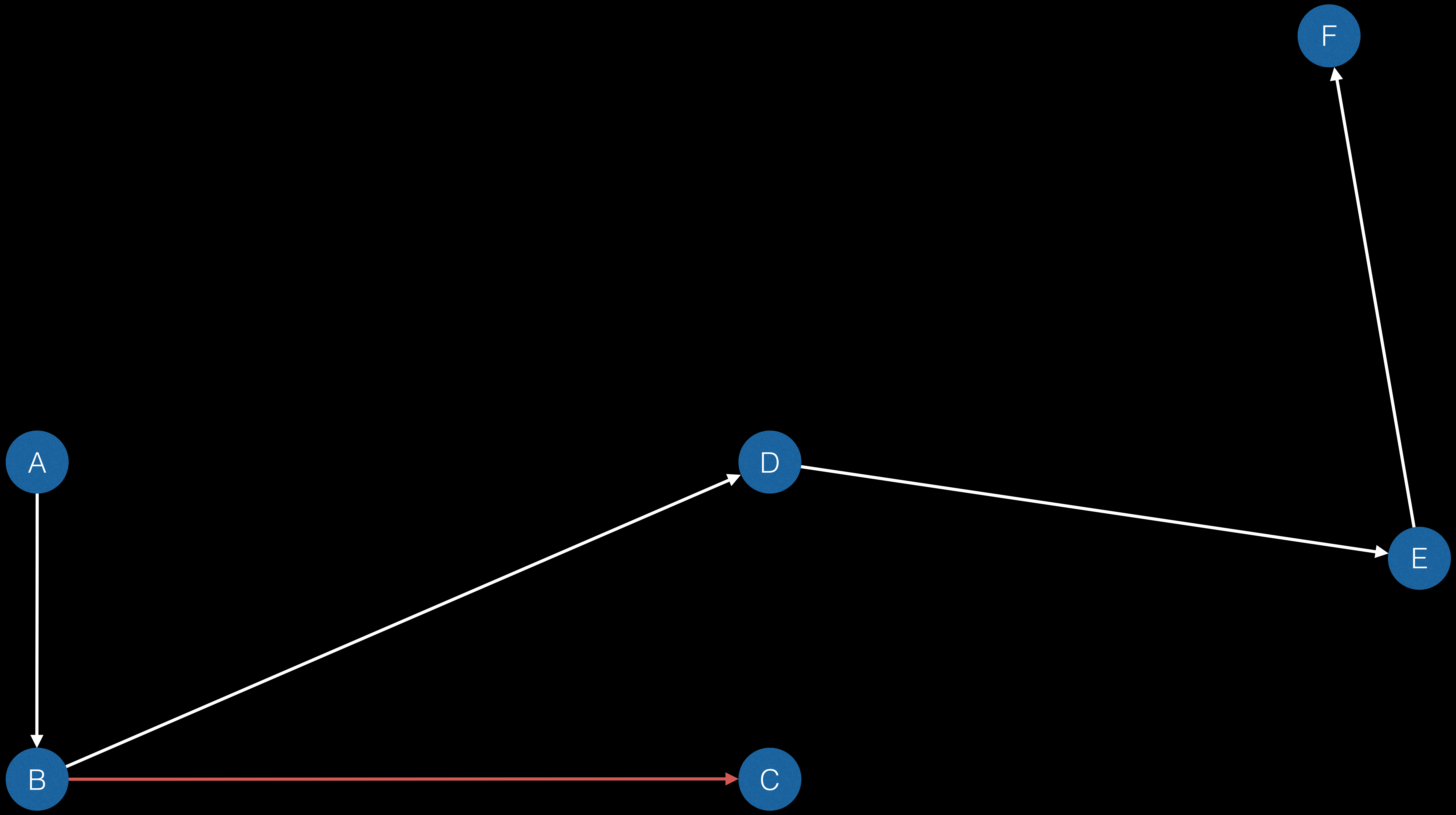
A case for **cohesion** and **coupling**

Single Responsibility Principle

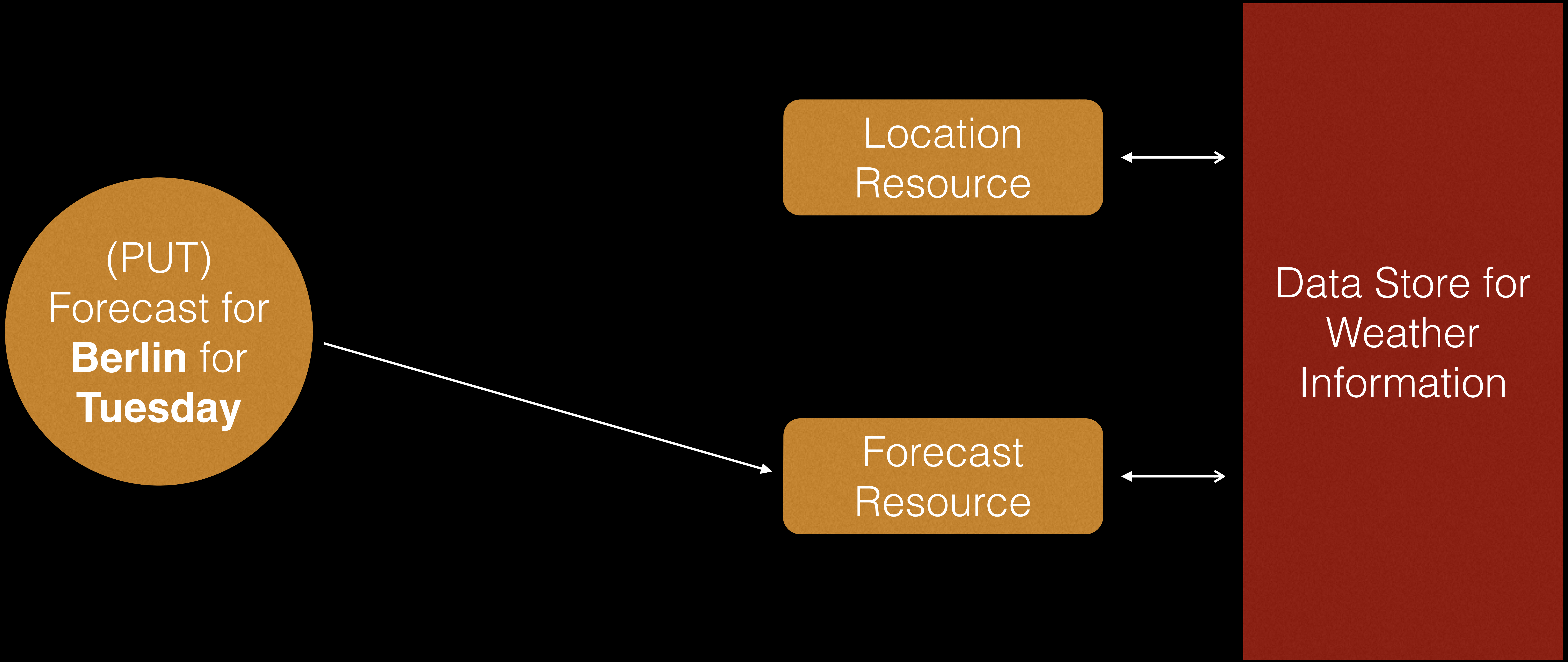
- Single Responsibility = A Case for Coupling and Cohesion
- How to **measure** and **evaluate** cohesion?
- REACHABILITY
    - http://www.cs.colostate.edu/~bieman/Pubs/tse98.pdf
    - Or a **simple concept** of semi-connected graphs might suffice (cohesion is the inverse of semi-connected components).


$$\begin{tikzcd}
A \arrow[d] & & D \\
B \arrow[urr] \arrow[rr, leftrightarrow] & & C
\end{tikzcd}$$


F
A
D
E
B
C


F
A
D
E
B
C

Services abstraction: processes owning and exposing **resources** responding to **messages**

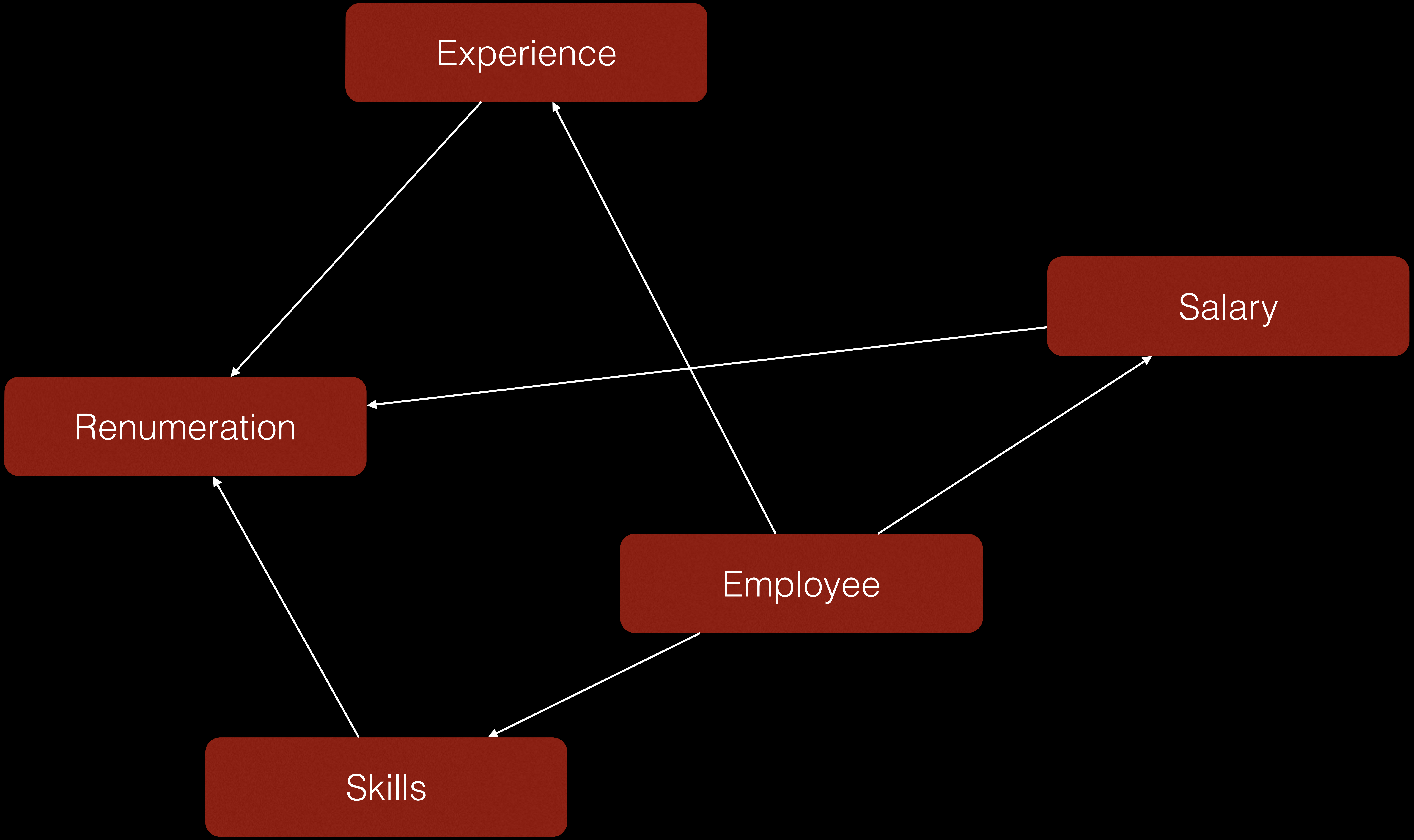

(PUT) Forecast for **Berlin** for **Tuesday**
Location Resource
Forecast Resource
Data Store for Weather Information

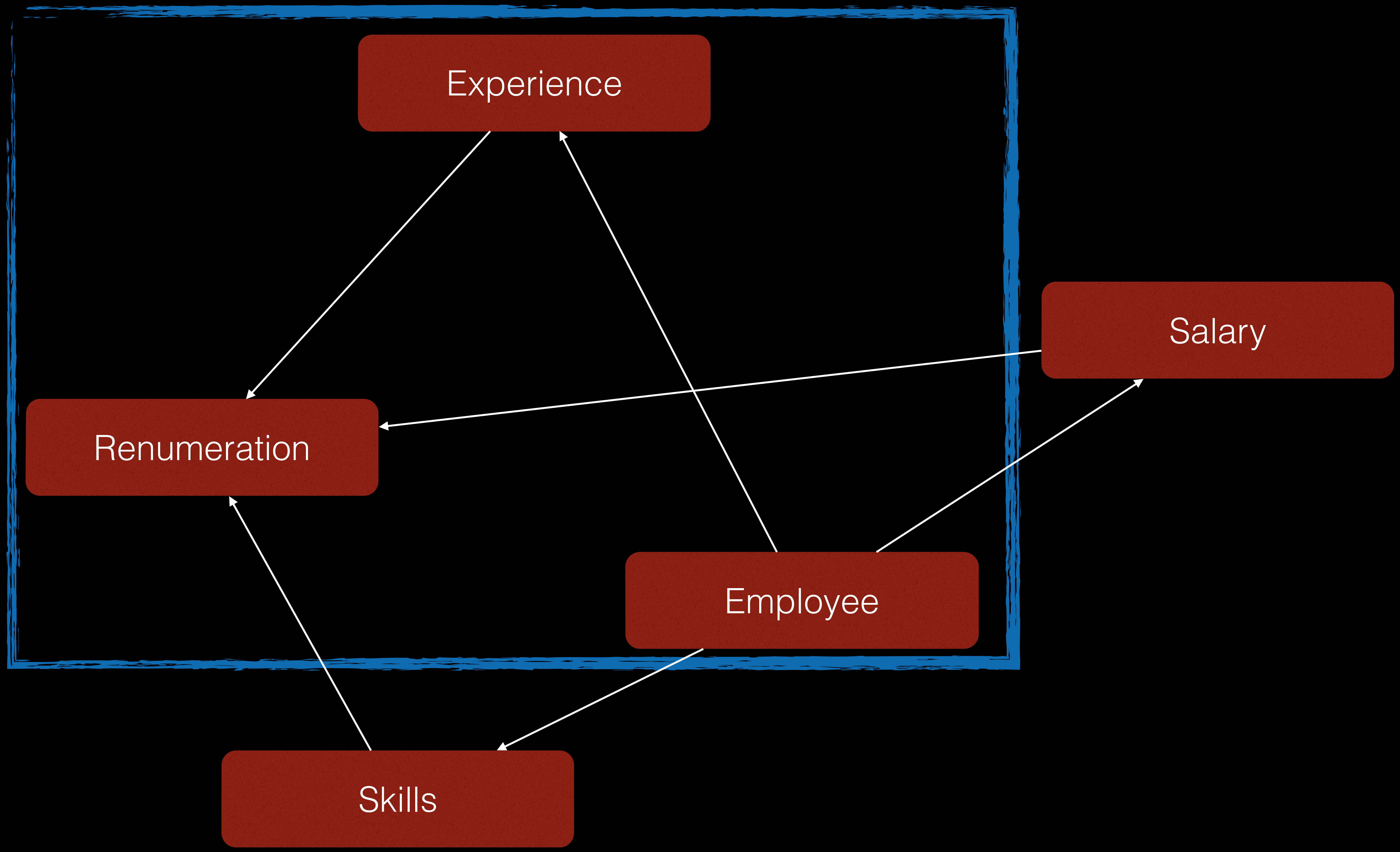
RESOURCE COHESION

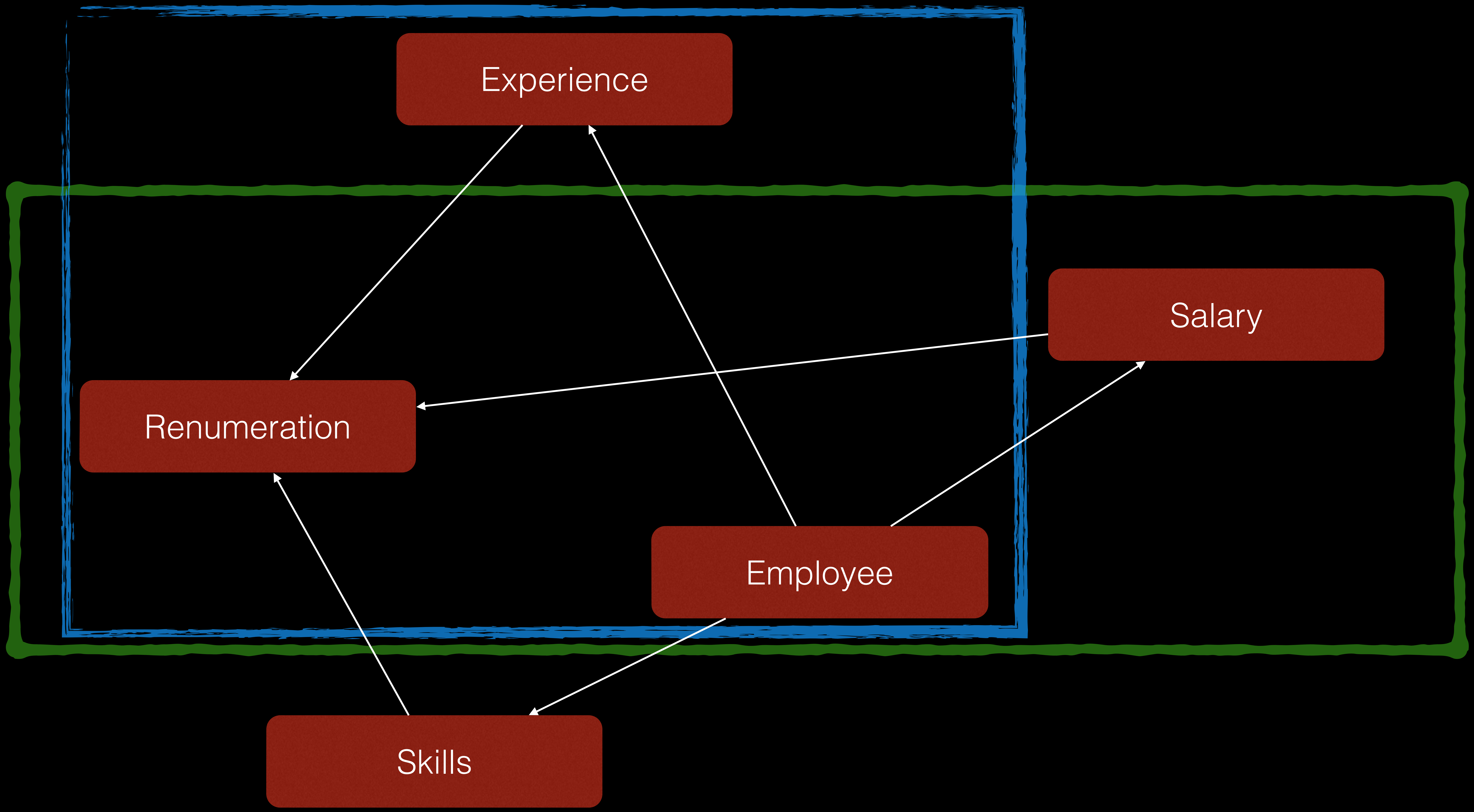
- There is a connection between resources iff…
  - There is a reference in a message directed at a resource (forecast refers to location).
- References induces edges in resource graph, then…
  - Cohesion is inverse to the number of semi-connected **components**.

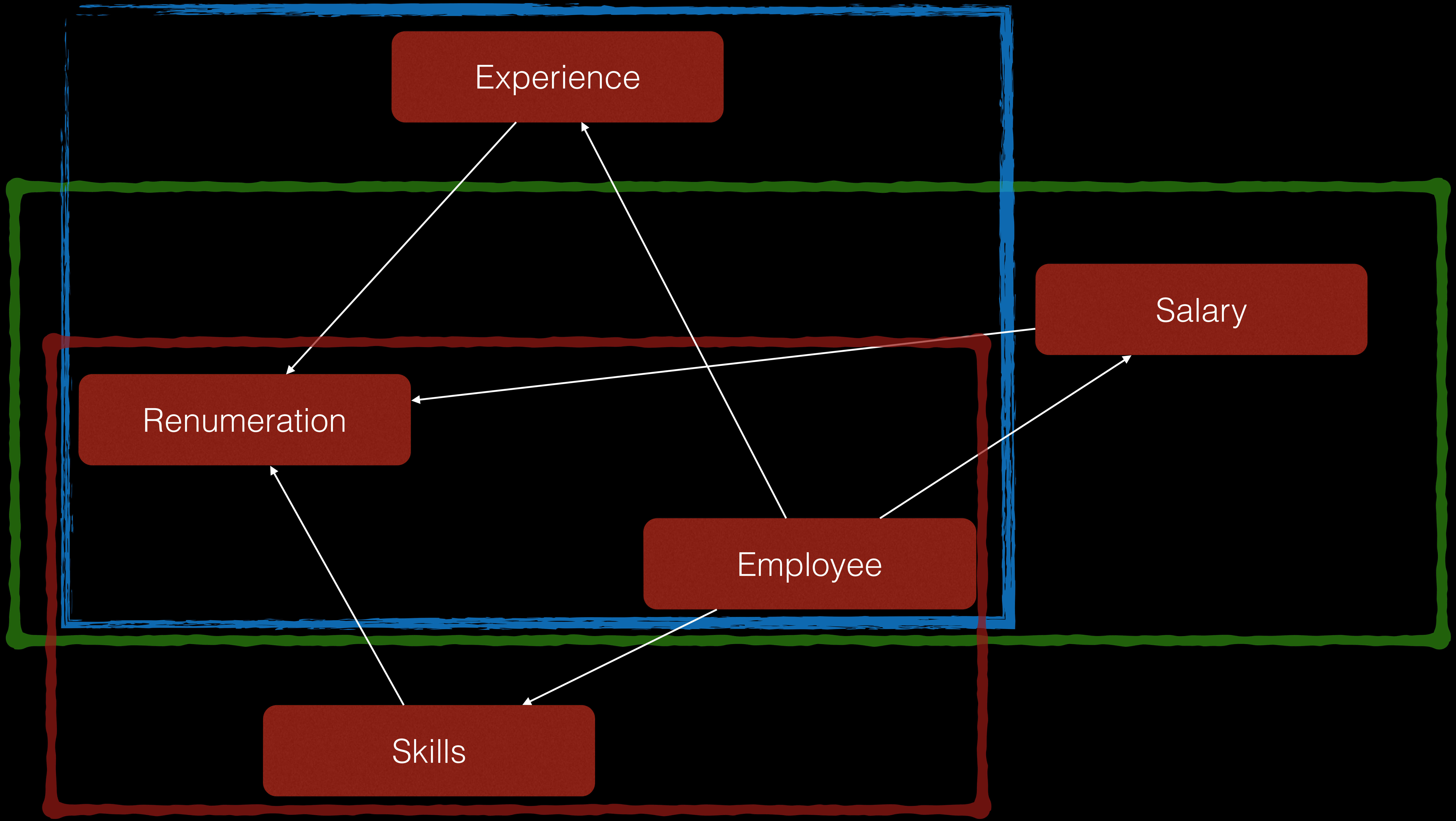
HR Module in ERP System

a trivial example


Experience
Salary
Renumeration
Employee
Skills

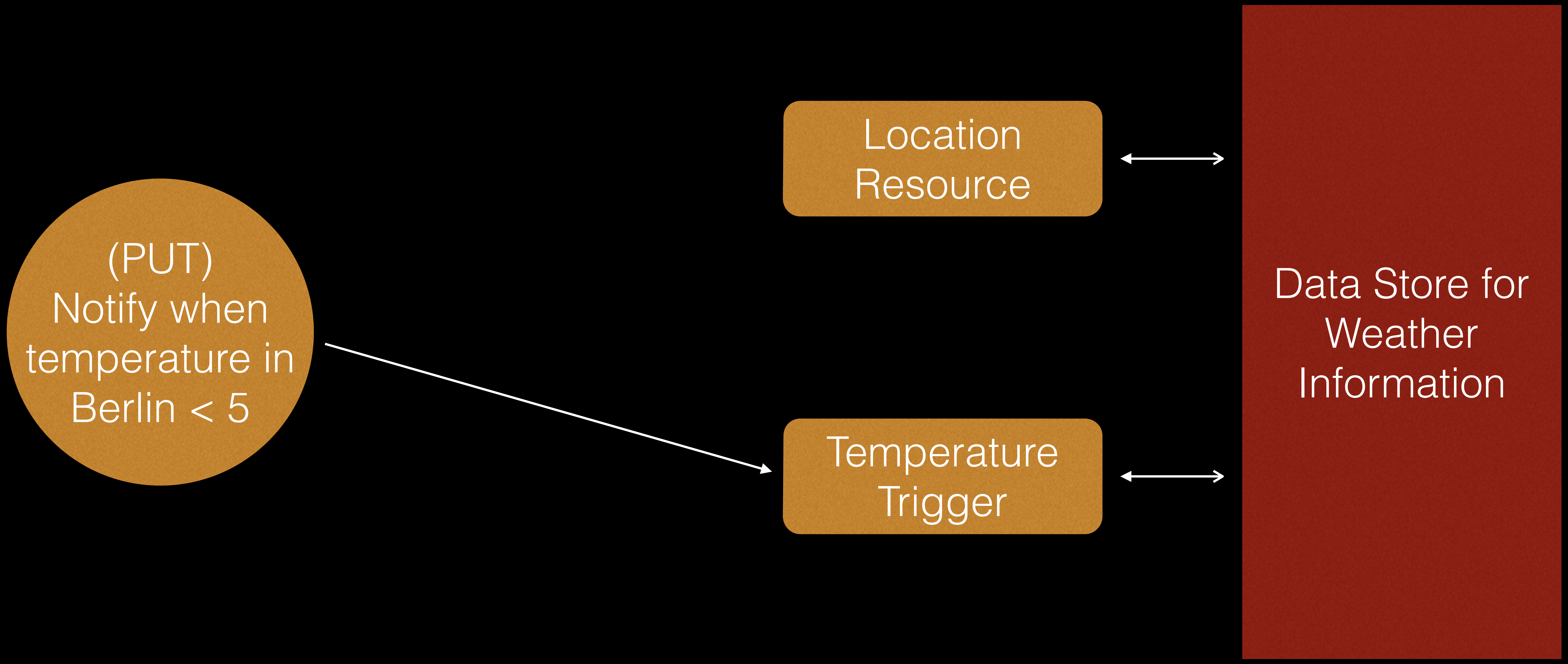

Experience
Salary
Renumeration
Employee
Skills


Experience
Salary
Renumeration
Employee
Skills

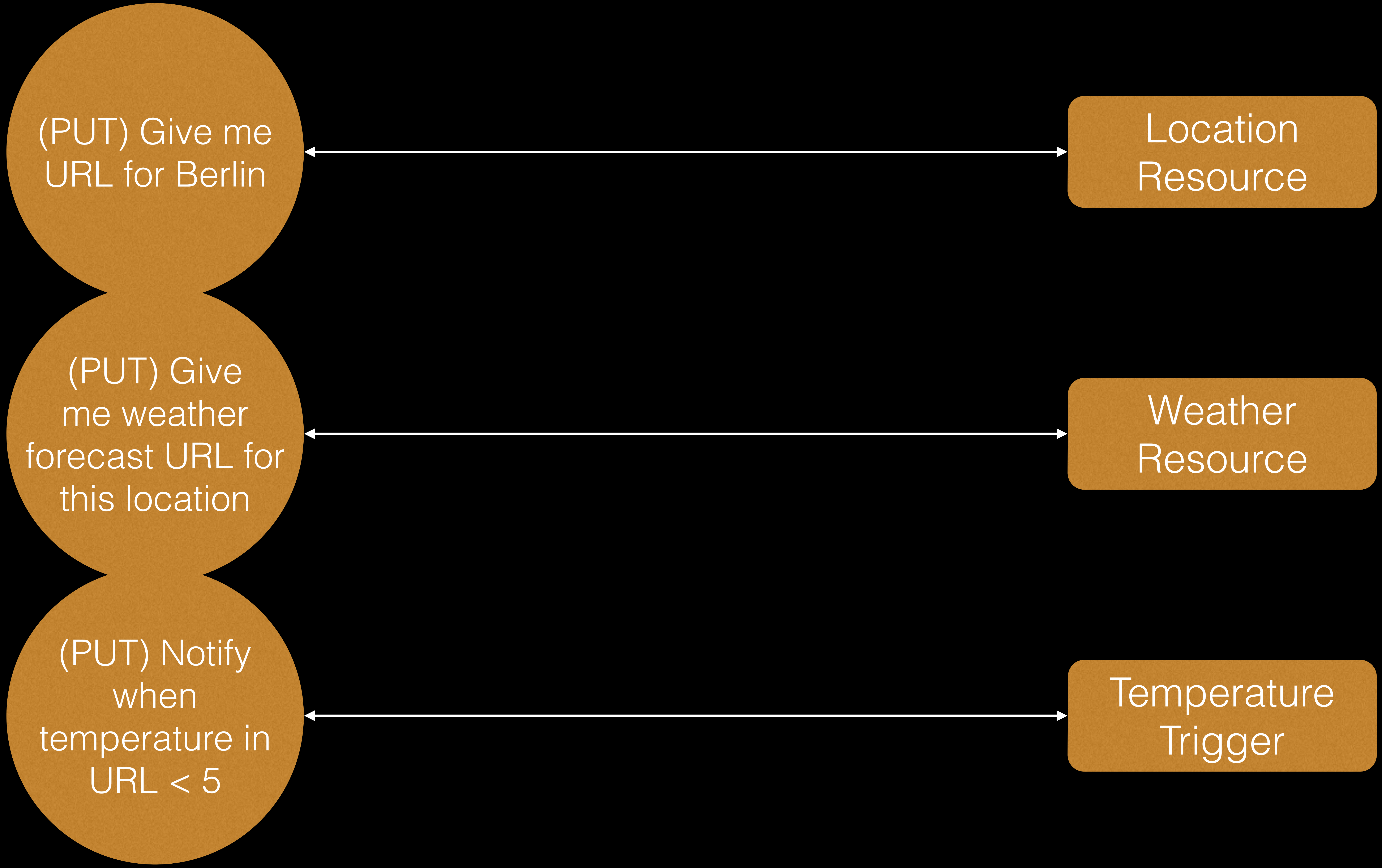

Experience
Salary
Renumeration
Employee
Skills

A Case for **Dependency Inversion**

A problem of **resource** dependencies can be resolved on resource level.


(PUT)
Notify when temperature in Berlin < 5
Location Resource
Temperature Trigger
Data Store for Weather Information

A problem of **resource** dependencies can be resolved on resource level.


(PUT) Give me URL for Berlin
Location Resource
(PUT) Give me weather forecast URL for this location
Weather Resource
(PUT) Notify when temperature in URL < 5
Temperature Trigger

- The issue of **trust**…
    - The fact that I trust the client does not mean that I trust whatever resource URL the client throws at me.
- The issue of **availability**…
    - What happens if the resource identified by the client is no longer available?
- The issue of **interoperability**…
    - Does the service identified by the client speaks my language? A need to for standard / conventional contracts arises (again).

Food for Thought

- Brown, Simon. Distributed big balls of mud (http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html)

- Meilir Page-Jones. Practical Guide to Structured Systems Design. Prentice-Hall. ISBN: 007-6092032779

- Yourdon, Edward; Constantine, Larry L. (1979) [1975]. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Yourdon Press. ISBN 0-13-854471-9

- Gilb, Tom (2005). Competitive Engineering. Elsevier Butterworth-Heinemann. ISBN 0-7506-6507-6.

- Brown, Simon. Software Architecture for Developers. https://leanpub.com/software-architecture-for-developers

Principles over Patterns

(and definitely much less buzzwords)

THANK YOU!