Software Architectures

From Design Patterns to Enterprise Architecture

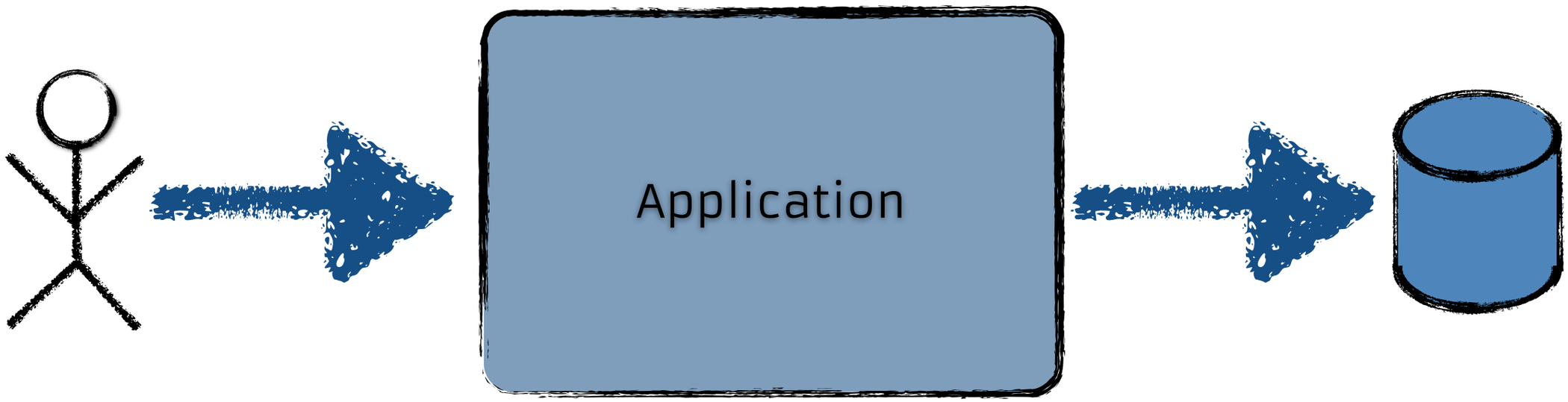
Enterprise Architectures II

Last week on "Software Architectures"

- Network architectures
  - Client-server model
  - Peer-to-peer model
- Multi-tier architectures
  - 1-tier, 2-tier, 3-tier, n-tier architectures
- Integration architectures
  - Extract, Transform, Load
  - Enterprise Application Integration
  - Service-oriented Architectures
- Rich Internet and Cloud Architecture
- … and of course… lot’s of thing you should not do.

Using relational databases in software

Overview



Data as arrays

- Represent the data you pull from the database in arrays
- Also organize data you write to the database in this fashion

```
Object[] user = db.getUser(id);
long id = (long) user[0];
String name = (String) user[1];
String fingerprint = (String) user[2];
```

id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548F3D02

Data as record sets

- Represent data using a class representing a relational table

```
DataTable users = db.getUsers();
for (DataRow row : users.getRows()) {
    print(“id: “ + row.get(“id”));
    print(“name: “ + row.get(“name”));
    print(“fingerprint: “ + row.get(“fingerprint”));
}
```

id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548F3D02

Domain models

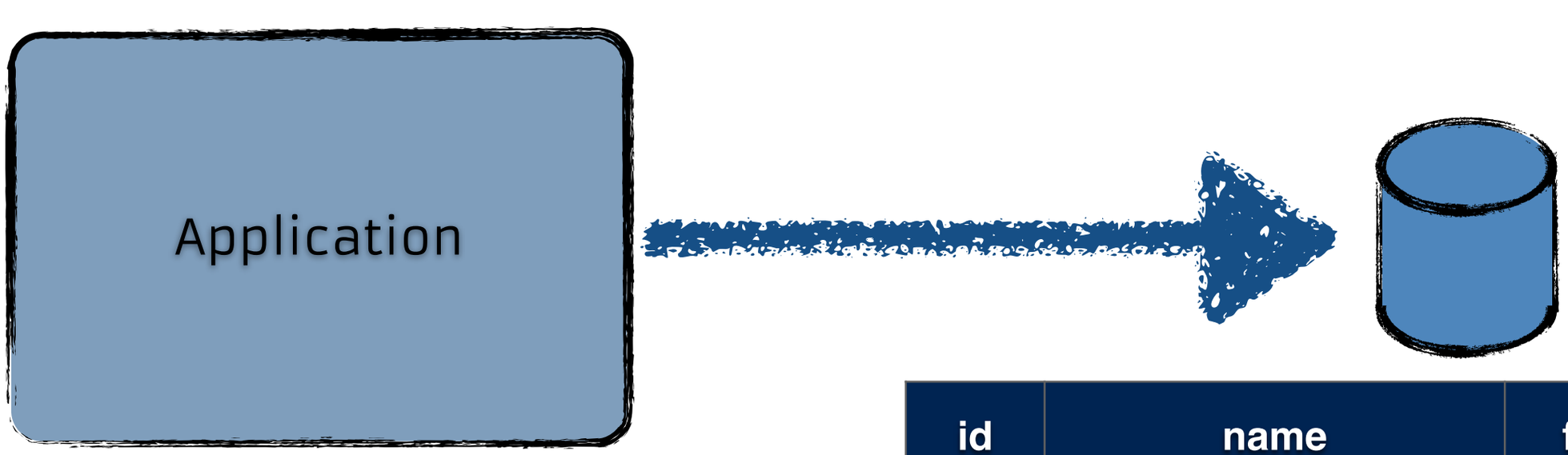
- Represent data from a database in an object that incorporates the behavior and the data

```
User userA = db.getUser(id);
print(“id:” + userA.id);
print(“name:” + userA.name);
print(“fingerprint:” + userA.fingerprint);
```

id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548F3D02

Populating your model

- So we have a data representation, but how is the data coming into our representation




```
Object[] user = db.getUser(id);
DataTable users = db.getUsers();
User userA = db.getUser(id);
```

id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548F3D02

Data accessor

- Write a class for each table you have that provides access to the data

```
public class UserAccessor {
    private DbConnection db;
    …
    public Object[] getUser(long id) {
        return db.ExecuteQuery(“SELECT * FROM User WHERE id = “ + id);
    }
    …
}
```

Centralized data access

- Well we could do this better, now could we?

```
public class ObjectAccessor {
    private DbConnection db;
    private String table;

    public ObjectAccess(String table) {
        this.table = table;
    }
    …
    public Object[] getObject(long id) {
        return db.ExecuteQuery(“SELECT * FROM “ + table +
                               ” WHERE id = “ + id);
    }
    …
}
```

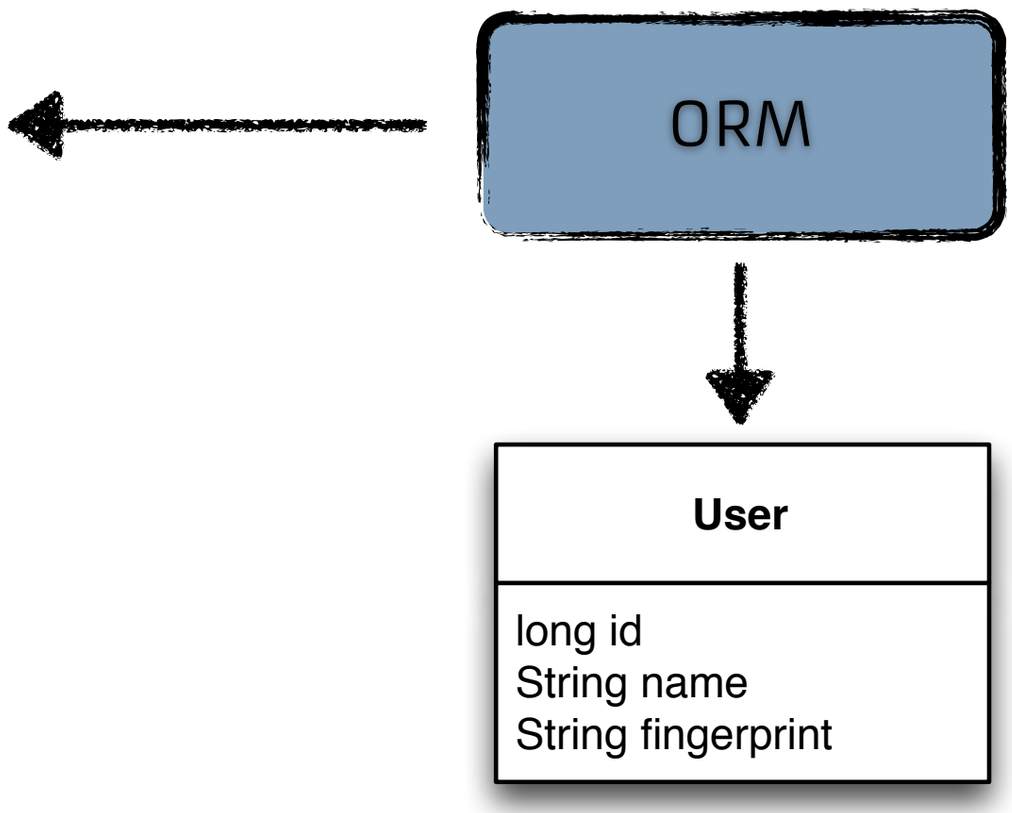
The Impedance Mismatch

- Well, now try to do this for your entire application
- Especially when using a domain model
- Hard, isn’t it?
- Well, that is because OO concepts and relational concepts don’t match that well.
- Think of Encapsulation, Visibility, Subclassing, Polymorphism…
- Are there alternatives?

Object-Relational Mapping

- A technique to transfer data between incompatible type systems
- See: Hibernate, Entity Framework, Active Record

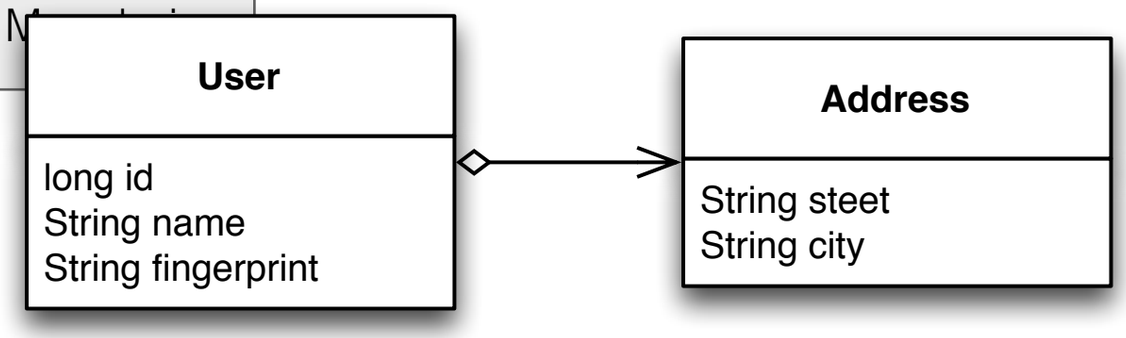
id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548F3D02



ORM: Component Mapping

- Represent parts of the table in separate objects

id	name	fingerprint	street	city
1	Max Mustermann	0FAB3171	Hochschulstraße 10	Darmstadt
2	Erika Musterfrau	28A01FC3	Zeil 111	Frankfurt
3	Sabine Test	548F3D02	Coblitzallee 8	M

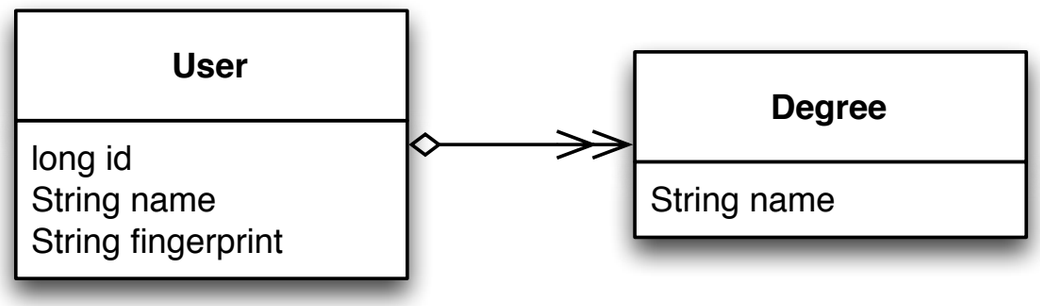


ORM: Collection Mapping

- Represent related tables in collections

id	name	fingerprint
1	Max Mustermann	0FAB3171
2	Erika Musterfrau	28A01FC3
3	Sabine Test	548E3D02

id	userid	degree
401	1	Prof.
402	1	Dr.
403	2	MBA



Storing non-persistent data

Think about storage times and guarantees

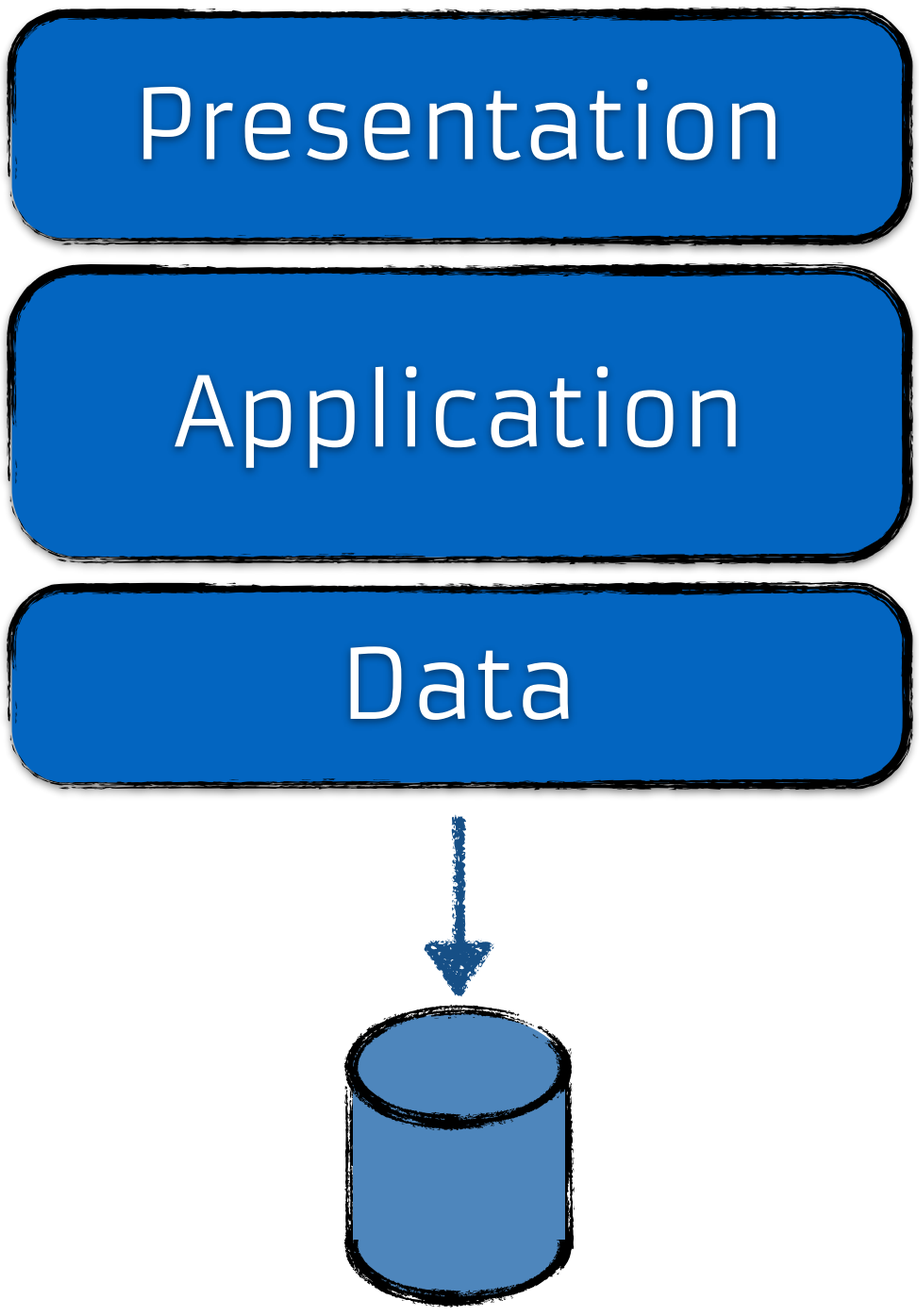
- How long is your data retention?
  - Forever?
  - One month?
  - One day?
  - Ten seconds?
- Should everything that you only keep for a certain amount of time go into the relational database?

Caching (Memcached and the others...)

- Think about memory caching
- Basically this is a big key-value store in memory
- Think of it as a hash table
- Fast lookups
- Can be distributed to many hosts

Breaking the linear architecture

Classic implementation of the 3-tier architecture

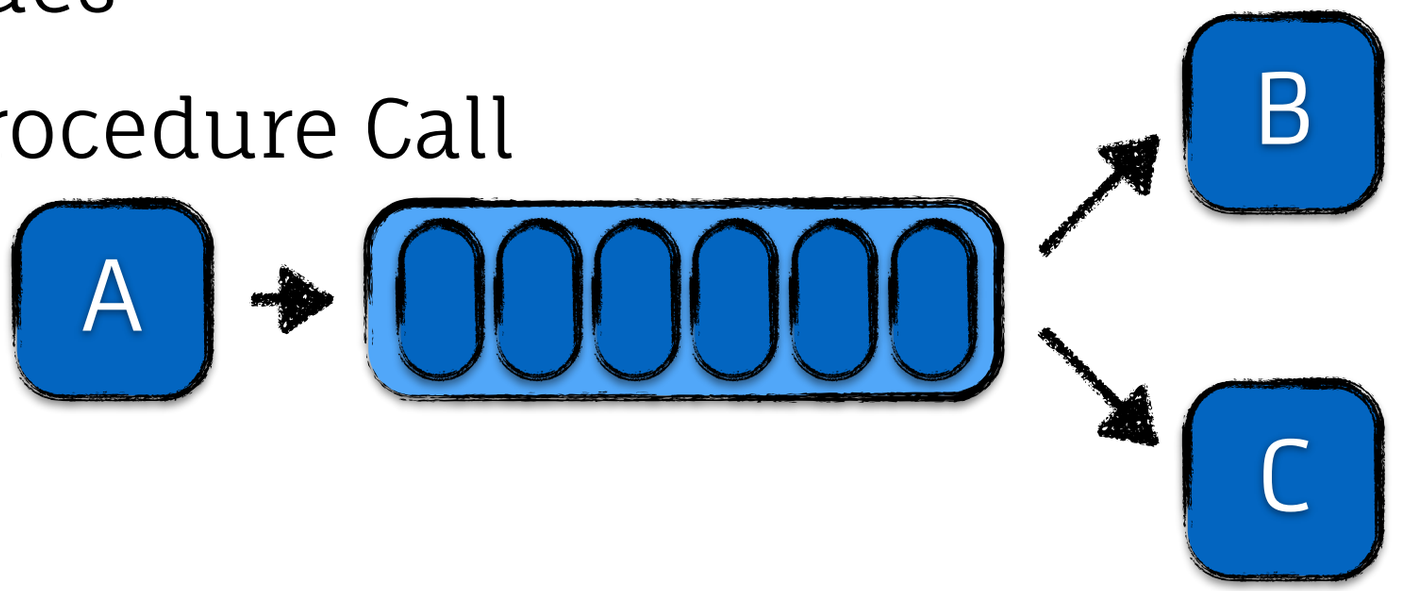


Message-oriented architectures

- Detach logical and physical components
- You don’t need to know where it runs anymore
- Basic concept: sending and receiving messages
- Reduces complexity of applications spanning computers or networks
- Standards & Implementations:
  - AMQP, DDS, XMPP, JMS, MSMQ, ActiveMQ, RabbitMQ
- In a way… SOAP and REST are also message-oriented

Message Queues

- Queue: Something where messages go
- (Some) possible uses:
  - Publish-Subscribe
  - Work queues
  - Remote Procedure Call



Object Request Brokers

- A somewhat older middleware technology
- ORBs form a distributed object system
- Interfaces have to be described an IDL
- See:
  - CORBA
  - DCOM
  - .NET Remoting
  - RMI

Enterprise Service Bus

- Part of a service-oriented architecture (SOA)
- ESBs level out differences between different interface formats (SOAP, REST, JNI, WCF, …)
- They may route requests to the proper services
- Controlled deployment and versioning of services
- Also offer some commodity services: event handling, message queueing, security, quality of service

Security

It is the assumptions where the vulnerabilities are.
- Fred B. Schneider

Major Web Security Flaws

OWASP Top 10 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

Coping with security issues

- Program defensively
- Do offensive testing
- Use security toolkits
- ALWAYS: be informed

Wrap up

- Handling data access in applications
- Handling of non-persistent data
- Breaking the linear architecture with middleware technologies
- Some thoughts about security

FRANKFURTER ENTWICKLERTAG

Frankfurter Entwicklertag 2014

19. Februar 2014
Uni Campus West, Frankfurt

Die Software Engineering Konferenz für die Rhein-Main-Region: Agilität, Qualität, Innovation.
Keynote Speaker: Bob Martin

Jetzt anmelden!
http://www.entwicklertag.de/frankfurt/2014/

Günstige Konditionen für Studenten