

Persistence
Layer
Generator

TLGEN

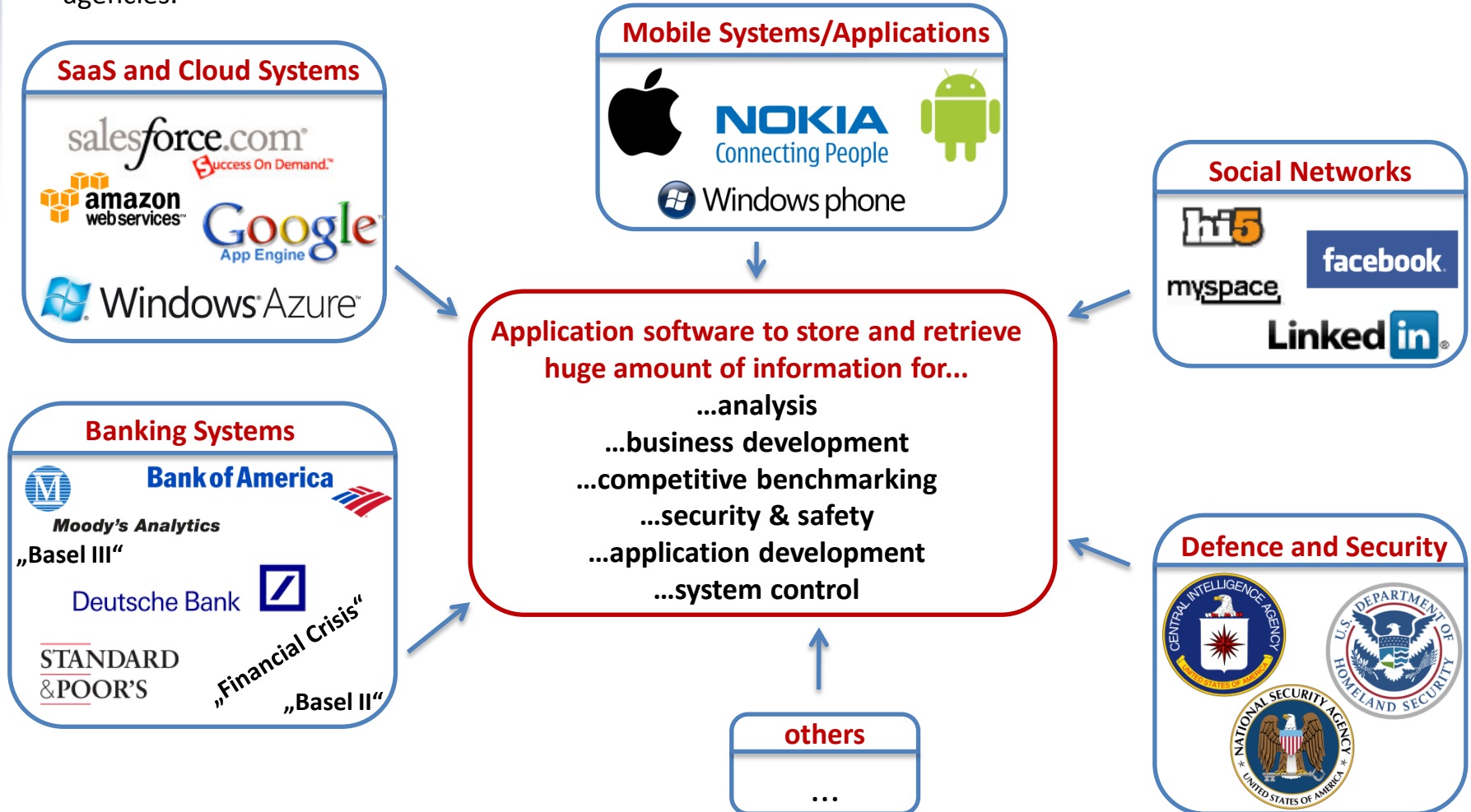
The 1st and only software solution

that automatically generates the persistence layer of
applications based on a domain model

1. Starting Point: World of Information
2. What is a Persistence Layer?
3. Actual Solutions for Persistence Layer Development
4. TLGen: The New World of Persistence Layer Development
5. TLGen: How does it work? A comparison with Hibernate
6. TLGen in the Field
7. TLGen: Unique Selling Points (USP)

1. Starting Point: World of Information

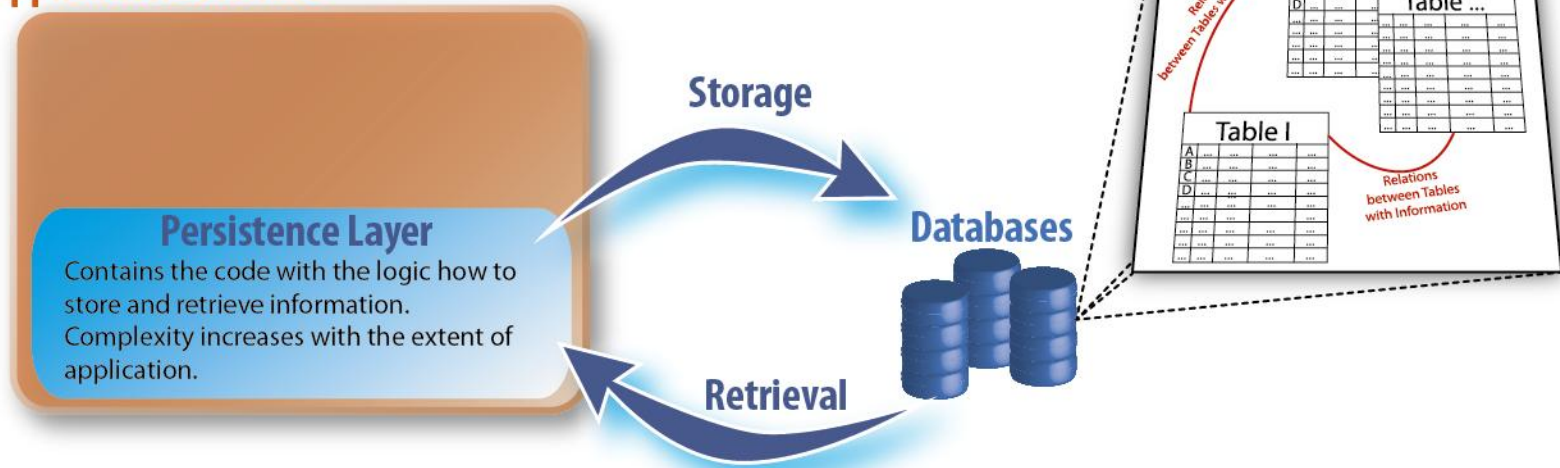
The development of database information systems increased benefited by the development of internet, web 2.0, application server and new information businesses (e.g. Google). Every application software stores and retrieves information vary from bank systems and social networks to mobile apps and defence/security agencies.



2. What is a Persistence Layer?

- “The persistence layer is a group of classes and components responsible for storing data to, and retrieving it from, one or more databases. This layer necessarily includes a model of the business domain entities (even if it’s only a metadata model). (Bauer, 2007)”
- Persistence layer contains the logic structures for the data management.
- The quality of a persistence layer is the determining factor for the rate and possibility to retrieve data from databases.

Application Software



3. Actual Solutions for Persistence Layer Development

Today there are several possibilities to develop a persistence layer:

- **Hand-coding with SQL/JDBC:** A solution which is rare and doesn't happen in medium and big projects:

Pro:

Contra:

- Slow and complicated
- Bugs afflicted (human errors)
- Team coordination problems

- **Hibernate with EJB 3 and JBoss Application Server:** One of the most chosen solution in the Java world. Other solutions like Toplink (IBM), Ibatis (Apache) or NHibernate (for the C# world) are similar to Hibernate's logic and Hibernate's project flow. Features for the Hibernate Framework:

Pro:

- Resolves some mismatches of hand-coding
- Uses domain models (philosophy of Model Driven Architecture)
- Works in every Java EE/J2EE application server (prefers JBoss Application Server)
- Open source project

Contra:

- Hibernate Core has to be always implemented in applications
- Implementation of Hibernate means implementation of Hibernate bugs
- Fixed terms of Hibernate have to be adopted to project logic (no flexible development)
- Changes in old persistence layer projects (with Hibernate) need a Hibernate Core update because of downward compatibility problems (dependence on Hibernate version)
- Own query language HQL (almost like SQL)
- Mapping metadata has to be written in native Hibernate XML files
- Java code has to be programmed (some parts will be generated)
- Bugs afflicted (because of hand-coding)
- Team coordination problems

4. TLGen: The New World of Persistence Layer Development

TLGen is a persistence layer code generator which generates the persistence layer code and test classes automatically using a domain or database model as input.

TLGen Features:

- **Generates** Java code in latest **J2EE** standard (**EJB 3**)
- TLGen combines the advantages of Model Driven Architecture (**MDA**) and Computer Aided Software Engineering (**CASE**)
- Works in every **application server**
- Uses a **domain model** as **information source** to generate the persistence layer code
- Optional: uses a **database model** to generate code for a **refactoring project** or to generate a new domain model for a new application development
- Tool independence - **no core** has to be implemented, **pure code** will be generated
- For customizing a **configuration file** will be created and be used as additional information source
- The generated code is „ready to go “ code - **no hand-coding** is needed
- Every time the application will be changed the new persistence layer can be easily generated
- Changes in logic can be immediately tested because of fast generated code
- A new level of quality in the structure of code will be achieved (database queries **up to 20 times faster**)
- Developed within 2 1/2 years
- Inspired by **30 years** of database expert knowledge
- TLGen is a technologically advanced software: 320,000 lines of code (Hibernate: 80,000 lines of code (2007))

5. TLGen: How does it work? A comparison with Hibernate

Project-Flow with TLGen:

- 1. TLGen uses the domain model to generate the persistence layer code (and some test classes) automatically
 - 2. A developer can add further information for the code generation with the TLGen configuration file (e.g. annotations)
- ➔ Persistence layer is written and can be tested

Logic optimization (green arrows):

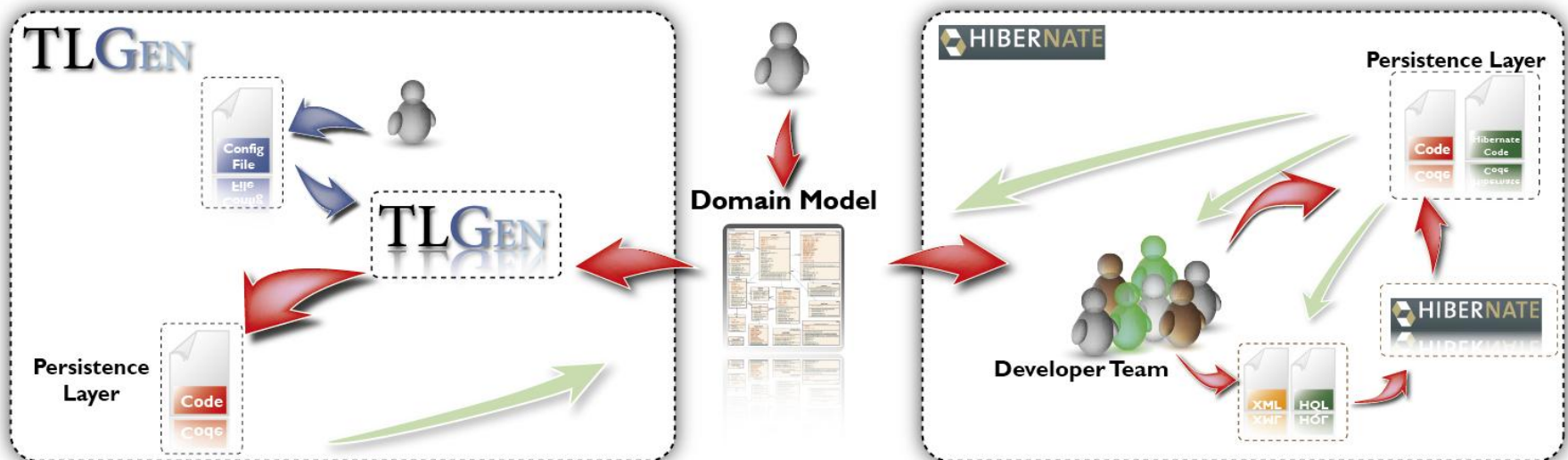
- 1. If changes are necessary after tests, the domain model (or TLGen-configuration file) will be adjusted
 - 2. New code will be generated automatically with the TLGen software
- ➔ New tests can be performed right after domain model changes

Project-Flow with Hibernate:

- 1. After domain model development the team will be put together
 - 2. Team writes the XML and HQL code
 - 3. Some parts of persistence layer will be generated by Hibernate
 - 4. Rest of Java code will be developed by team members
 - 5. Hibernate framework will be implemented in the application and therefore becomes a part of it
- ➔ Persistence layer is written and can be tested

Logic and bug optimization (green arrows):

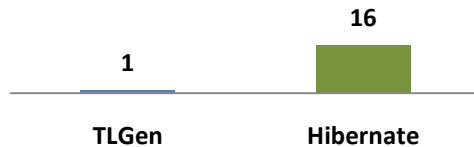
- 1. Hand-written code has to be changed (hand coding is the biggest source for bugs within IT projects and increases with project size)
 - 2. Logic problems will be changed in the Java, XML and HQL code
 - 3. The domain model will be adjusted
- ➔ New tests can be performed only after simultaneous changes of some criteria (e.g. programmed code, domain model, Hibernate adjustment)



6. TLGen in the Field

In 2010 for the first time TLGen was applied in a real project for a worldwide operating ICT provider located in Germany. This project was implemented parallel as a test against the usual Hibernate project. The results are:

Expenditure of Time (number of weeks)



The Hibernate project took 16 weeks till the first test could be performed. The persistence layer development by TLGen took 1 week. The main part was the writing of the configuration file, the generation of the whole persistence layer code took only some seconds.

Team Size (number of persons)



The Hibernate project needed 10 developers to write the code and implement the framework while TLGen needed only one person to write the configuration file and to generate the whole persistence layer code.

Number of Java Classes



At the end of the project, Hibernate code comprised almost 700 java classes while the TLGen generated code comprised 330 perfect structured java classes.

Mainly this gap happened because of team organization problems, a well known problem in the IT world.

One effect was that a complex database query with Hibernate code took almost 10 seconds while the query with TLGen code took only 0.5 seconds.

7. TLGen: Unique Selling Points (USP)

What will be changed for the IT world using TLGen? What are the advantages?

Significant Cost Savings → **Up to 20 Times Cheaper**

- No big developer teams needed
- No complex problems correction (control only with domain model and no hand-coding)

Significant Time Savings → **Up to 15 Times Faster**

- Fastest project implementation ever with automatic code generation

Impressive Flexibility for Implementation of Project Logic → **Fast Idea Implementation**

- 100 % control with the domain model as input source

New kind of solution for refactoring projects → **Big World Market**

- TLGen is a cheap, fast and new solution for refactoring projects because old databases can be used for the new persistence layer in a new application (TLGen generates persistence layer code with a database model as input)

Tremendous Quality Improvement → **Database Queries up to 20 times faster**

- No human hand coding implies no bugs (code will be automatically generated)
- Logic structured code