Does Swiss IT Matter?

Perspektiven des Informatikstandorts Schweiz

Eine Fachtagung der Java User Group Schweiz
und der Credit Suisse
im Rahmen der informatica08

Montag, 29. September 2008

im Forum St. Peter der Credit Suisse in Zürich

Trägerschaft/Organisation

CREDIT SUISSE

JUG's
Java User Group Switzerland

netcetera Quality Software Engineering

Agenda

- Das Innovationsprofil einer Grossbank im Bereich Java Technologie am Beispiel der Credit Suisse
- Die Transformation vom Projektdienstleister zum Produktanbieter – Der Turnaround einer KMU-Unternehmung
- Die Konjunkturentwicklung der IT-Branche
- Gehaltsperspektiven und ihre Entwicklung auf dem IT Arbeitsmarkt Schweiz
- Integriertes Personalmanagement in der IT-Division eines globalen Finanzdienstleisters
- Der IT Arbeitsmarkt Schweiz - Koordinationsprobleme Angebot – Nachfrage
- Paneldiskussion: Was für Informatiker braucht es zukünftig am IT Standort Schweiz?

Agenda

- Das Innovationsprofil einer Grossbank im Bereich Java Technologie am Beispiel der Credit Suisse
- Die Transformation vom Projektdienstleister zum Produktanbieter – Der Turnaround einer KMU-Unternehmung
- Die Konjunkturentwicklung der IT-Branche
- Gehaltsperspektiven und ihre Entwicklung auf dem IT Arbeitsmarkt Schweiz
- Integriertes Personalmanagement in der IT-Division eines globalen Finanzdienstleisters
- Der IT Arbeitsmarkt Schweiz - Koordinationsprobleme Angebot – Nachfrage
- Paneldiskussion: Was für Informatiker braucht es zukünftig am IT Standort Schweiz?

Speaker Info – Stephan Hug

CTO Private Banking & CTO Switzerland, Credit Suisse

Stephan Hug ist der Chief Technology Officer (CTO) Private Banking und CTO Switzerland der Credit Suisse.

Stephan Hug trat 1995 in die Credit Suisse ein. In den 13 Jahren bei der Credit Suisse übte er zahlreiche Funktionen aus. Unter anderem war er Entwickler, Projektleiter und Architekt im Foreign Exchange und "Money Market IT"-Bereich der ehemaligen Credit Suisse First Boston in Zürich, New York und London.

Nach dem Wechsel in die Private Banking IT, wo er als Ressortleiter, Programmleiter OTEx (einem Grossprojekt im Bereich Client Trading) und zuletzt als IT-Architekt des Departments "Trading & Operations IT" tätig war, übernahm er 2007 die Verantwortung für die Applikations-Architektur der Private Banking IT.

Bevor Stephan Hug zur Credit Suisse stiess, war er im Bereich Research & Development eines Industriebetriebs tätig, und half einer Schweizer Grossbank den Trading Floor zu gestalten. Er besitzt einen Masters Degree in Computer Science der ETH Zürich sowie ein Nachdiplomstudium in Finance der Universität St. Gallen.

CREDIT SUISSE

Das Innovationsprofil einer Grossbank im Bereich JAVA Technologie am Beispiel der Credit Suisse

Stephan Hug
CTO Private Banking / Switzerland

September 2008

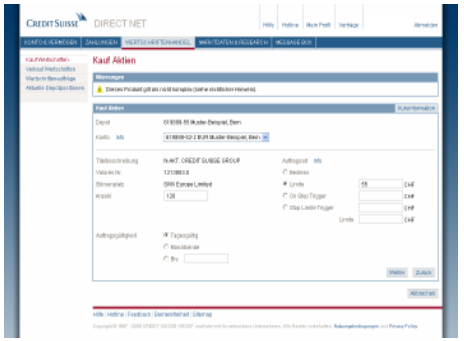
Agenda

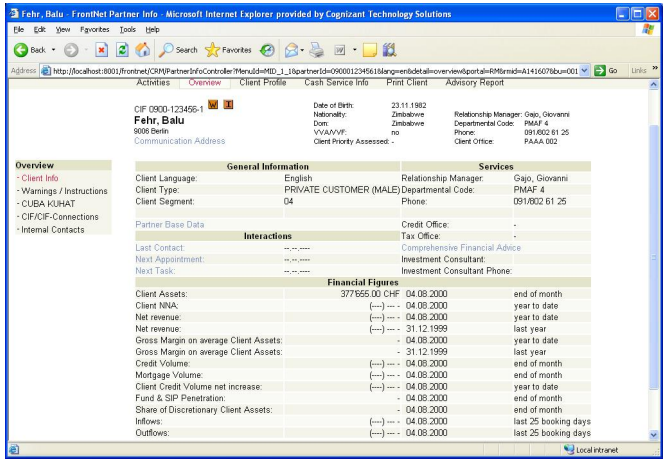
1	**DirectNet & FrontNet: Just the beginning (and an easy one)**
2	**Facts & Figures at the example of the SBIP**
3	**The challenge**
4	**How we address the challenge**
5	**Q&A**

DirectNet & FrontNet: Just the well-known beginning (and an easy one)

DirectNet

- 549'000 online customers (December 2007)
- 606'000 online customers (August 2008)
- Avg. 2.6 million logins per month





Description

- Intranet portal for relationship managers
- Combines existing functionality with a modern integrated user interface
- Results in improved and more flexible workplaces for relationship managers

Java has been very successful so far at CS, but most of the applications are no more than nice user interfaces to the mainframe!

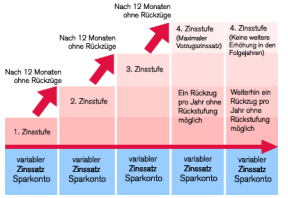
Facts & Figures: Product Portfolio

Payment transactions

- Payment order inland
- Payment order FC
- Direct debit
- Credit cards
- Checks
- Notes
- Foreign exchange
- Safe-deposit box
- Documentary credits
- ....

Accounts

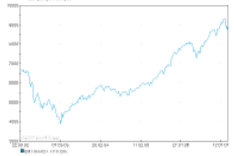
- Savings accounts
- Private accounts
- Foreign curr. accounts
- Flex Inv. account
- ...



Universal Bank

Investments/Provisions

- Trading
- Investment funds
- OTC financial products
- Life insurances
- Securities Lending & Borrowing
- Fiduciary deposit
- Pension accounts
- Safekeeping accounts
- ...

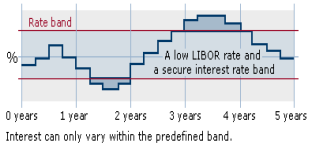


Credits

- Mortgages
- Lombard loans
- Personal loans
- Leasing
- Security deposits
- Fixed advance
- Commerce loans
- Plain loans
- ...

Treasury

- Foreign exchange trans.
- Forward transactions
- Foreign currency options
- Precious metals
- ...



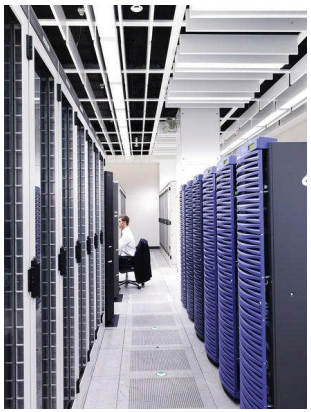
Which systems process all those products?

Facts & Figures: Myth & truth

Credit Suisse

Headquarter:	Zurich
Year of Foundation:	1856
Business Areas:	Private Banking, Investment Banking and Asset Management
Locations in:	Switzerland, Americas, EMEA, APAC
Net Income 2007:	7 760 CHF million
Return on Equity 2007:	18.0 %
Employees*:	~49'000

Credit Suisse IT Switzerland

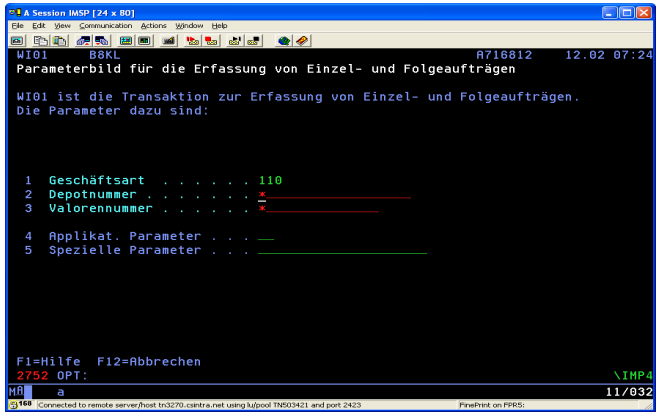


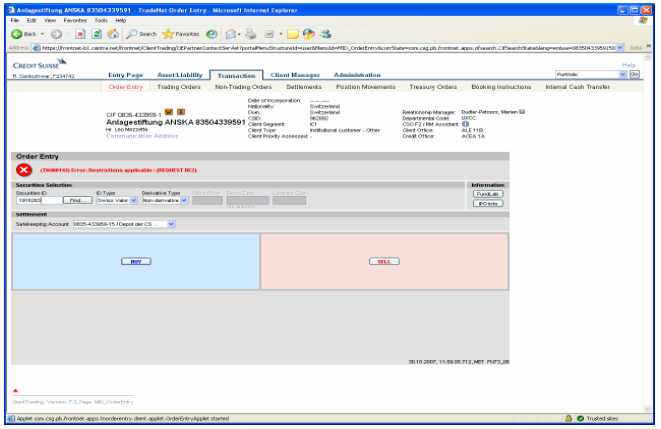
Number of servers:	6,750 Servers (Windows, UNIX, z/OS)
Number of Applications:	~ 800 applications
Lines of Code in PL/1:	~32 Mio. lines of code
Lines of Code in Java:	~11 Mio. lines of code
Payment transactions:	~ 250 Mio. / year
Straight through processing rate:	> 92 %
Printed pages:	~ 224 Mio. / year
Email:	~ 339 Mio. / year
Employees:	~4'000
Employees hired p.a.:	~400-500

Mainly this (very large) system!

The Challenge

The majority of the business critical logic is in PL/1 on the mainframe





Challenge: How do you get from black screens (and a lot of business logic behind it!) to modern GUI's with business logic in Java (rather than calling hundreds of CORBA services on the mainframe)

Why do we need to do this?

- Lifecycle issues with applications, some are 30+ years old
- Difficulties to deal with some of the modern products (e.g. structured products)
- Difficulties to cope with increasing demand for flexibility
- Difficulties to find PL/1 skills in the marketplace

TIOBE Index

TIOBE Programming Community Index December 2007

www.tiobe.com/tpci.htm

Position		Trend	Programming Language	Ratings		Delta
Dec 06	Dec 07			Dec 06	Dec 07	
1	1	=	Java	19.91%	20.05%	0.14%
2	2	=	C	16.61%	13.17%	-3.44%
4	3	↑	(Visual) Basic	8.91%	10.22%	1.31%
5	4	↑	PHP	8.53%	8.39%	-0.14%
3	5	↓↓	C++	10.41%	7.87%	-2.54%
7	6	↑	Python	3.77%	4.70%	0.93%
6	7	↓	Perl	6.39%	4.38%	-2.01%
8	8	=	C#	3.17%	3.99%	0.82%
11	9	↑↑	Ruby	2.33%	3.09%	0.76%
10	10	=	JavaScript	2.56%	2.73%	0.17%
..	..					
18	15	↑↑↑	COBOL	0.60%	0.89%	0.29%
>50	>50	=	PL/1	<0.10%	<0.10%	0.00%

- Gives an indication of the popularity of programming languages
- The attractiveness is reflected in available resources, tools and methods
- PL/1 seems not to be a long-term option

Ratings are based on world-wide availability of: skilled engineers; courses; third party vendors
Ratings are calculated by counting hits of the most popular search engines, such as Google, MSN, Yahoo!
TIOBE considers programming languages with a rating > 0.7% for more than three consecutive months as mainstream languages

How we address the challenge

Improve our existing Java Application Platform (JAP) to become a true alternative to the mainframe

- Elements required:
  - Distributed: Includes host and non-host applications (that interact during batch and online)
  - Logically layered: apply clear SOA layer architecture including tracability to business components and business processes
  - State-of-the art software engineering: Will use MDA, rule engines, process orchestration engines
  - Failure resistant: Will consist of multiple runtime systems which are failure-independent (replication of shared information)
  - Scalability: Since volumes are growing constantly, scalability has to be a key element of the future platform

Assets we already have

- We do not have to start from scratch, we can look back on more than 10 years of experience with SOA and have multiple assets already

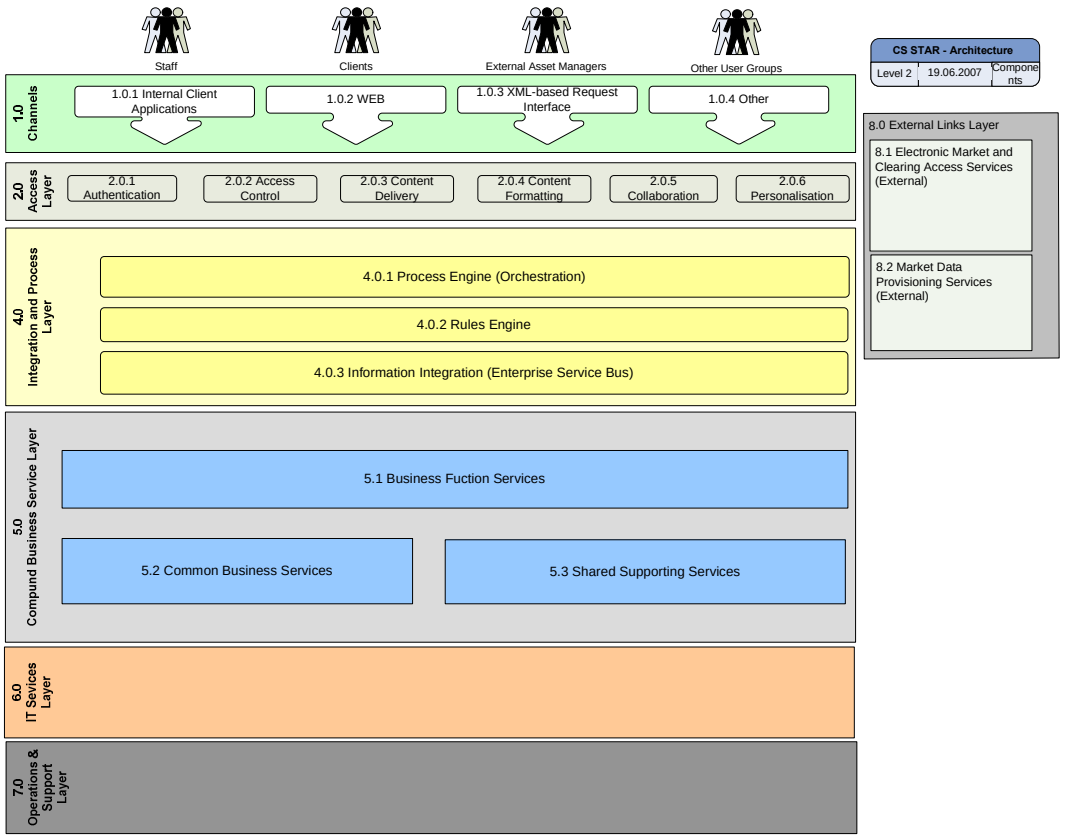
  - Integration-Architecture/Middleware (Credit Suisse Information Bus, Event Bus Infrastructure)
    - CORBA and Websphere MQ is in use already and can help with the co-existence

  - Application Platforms (Java Application Platform)
    - Good experience with JAP
    - Can and will position JAP as platform of the future

  - Governance & Processes (Quality Check Process, Project Rewview Borad Process, IT-Projekt Vorgehen)
    - Leverage existing governance and processes
    - Leverage CMMI

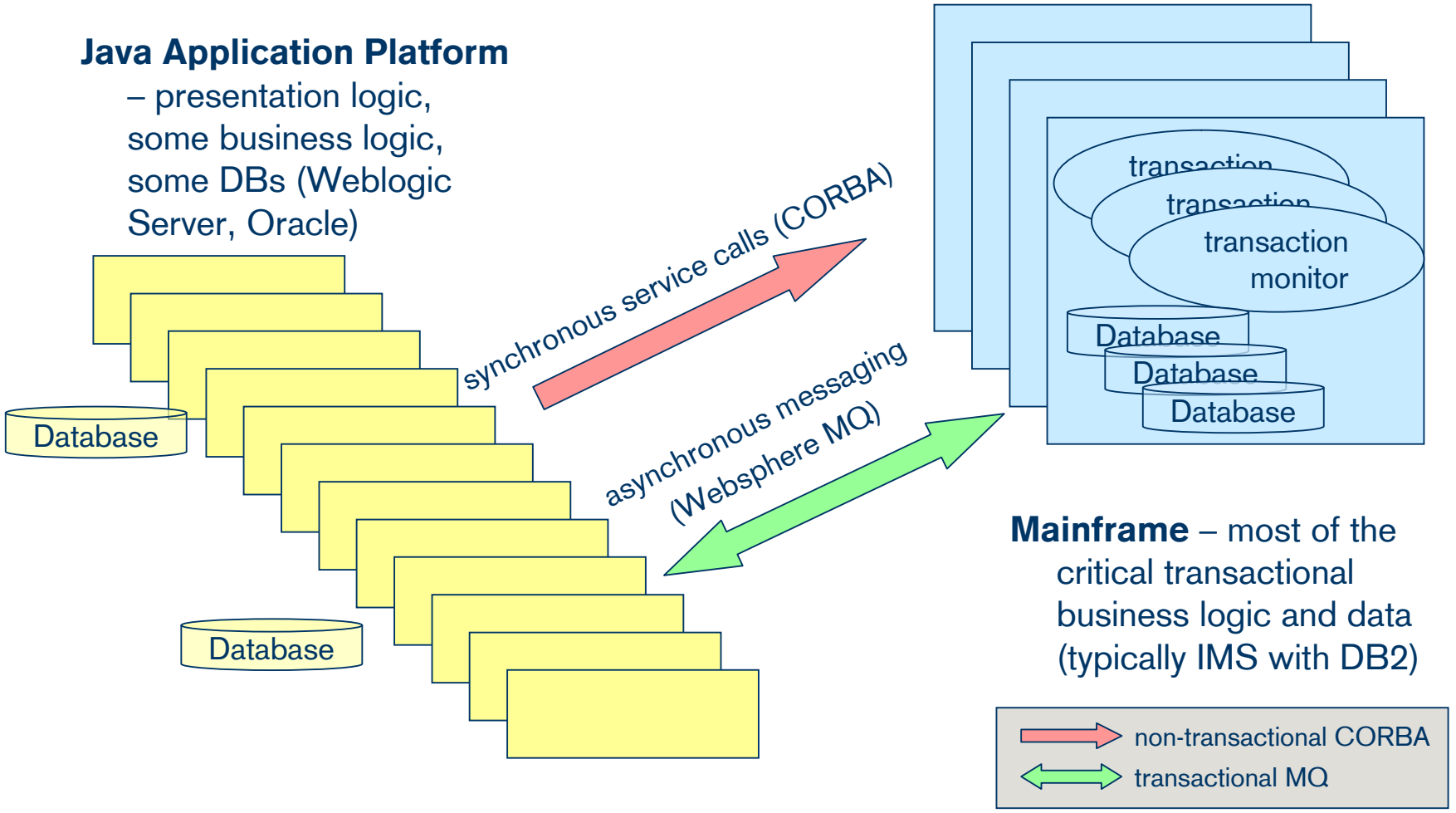
  - People (not many, distributed across the organization)
    - Bring the right people together

SOA Layer Architecture

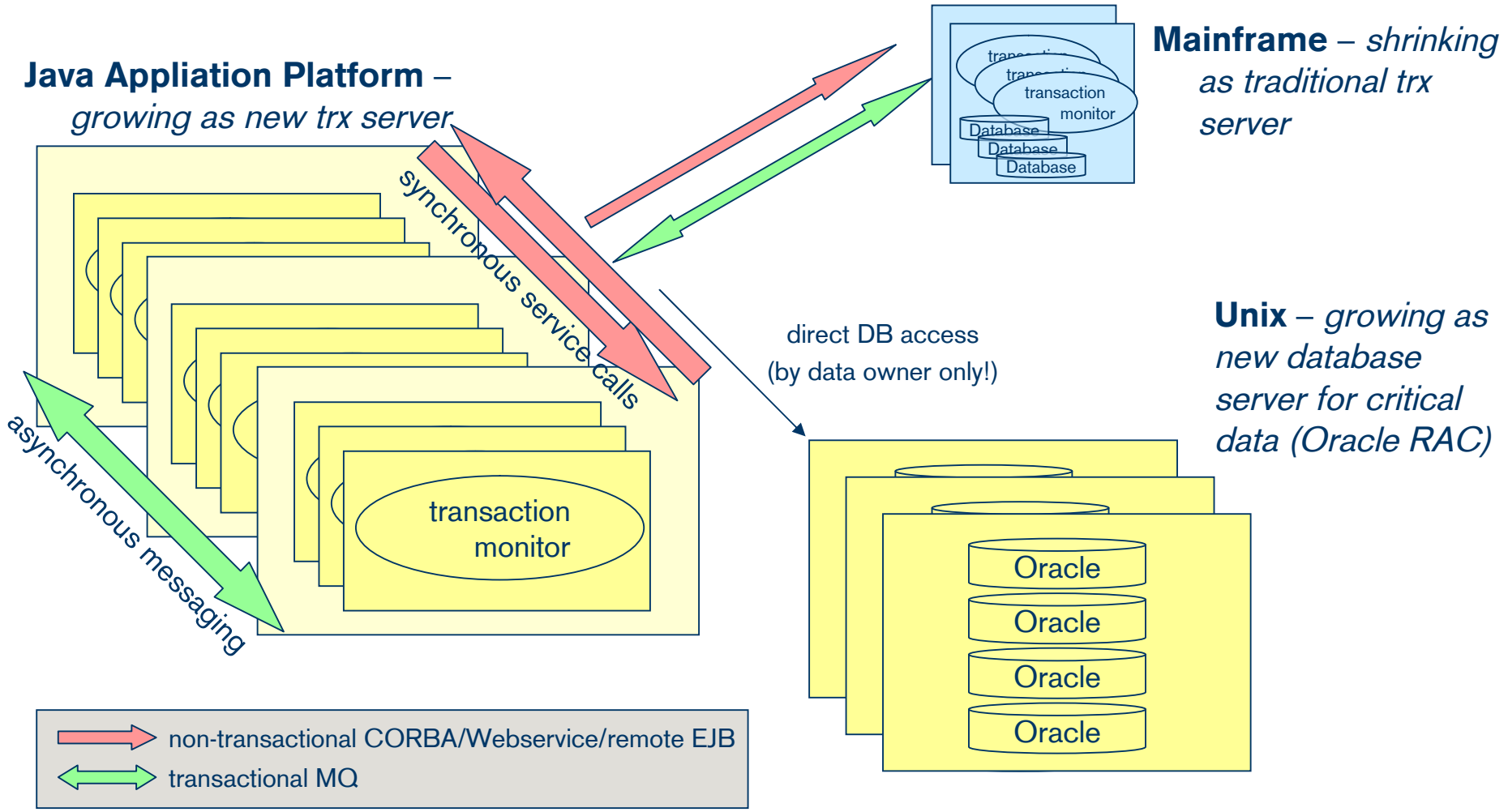


- Classic layer architecture
- Includes more sophisticated layers such as Process Engine anfd Rules Engine
- Transaction Processing is the key to success when trying to position this architecture as the Java-based alternative to the mainframe
- Database is measured with DB/2 on the mainframe

Database: Current Architecture with critical business data mainly in DB/2 and IMS on the mainframe



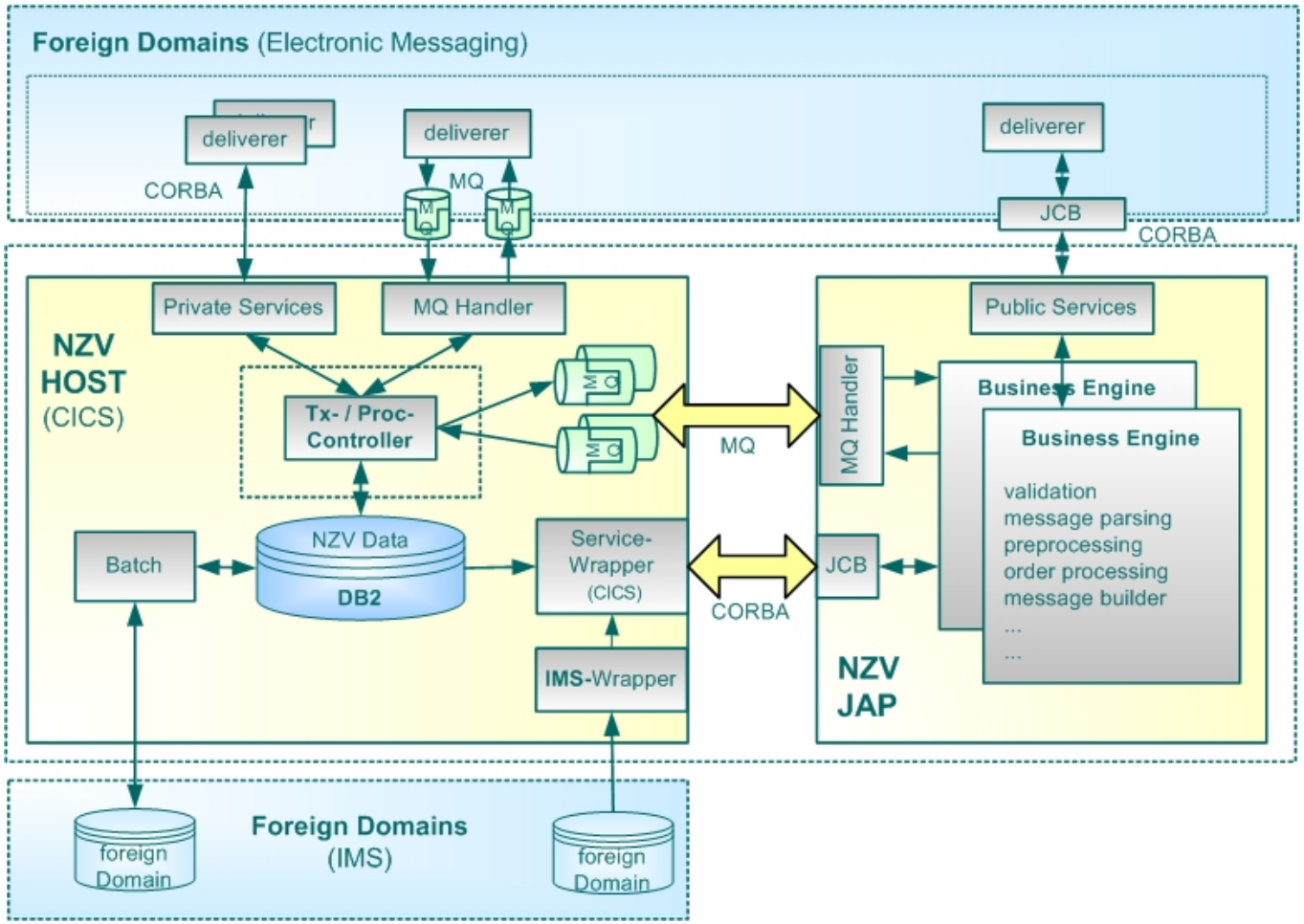
Database: New Architecture with business critical data in Oracle RAC on UNIX



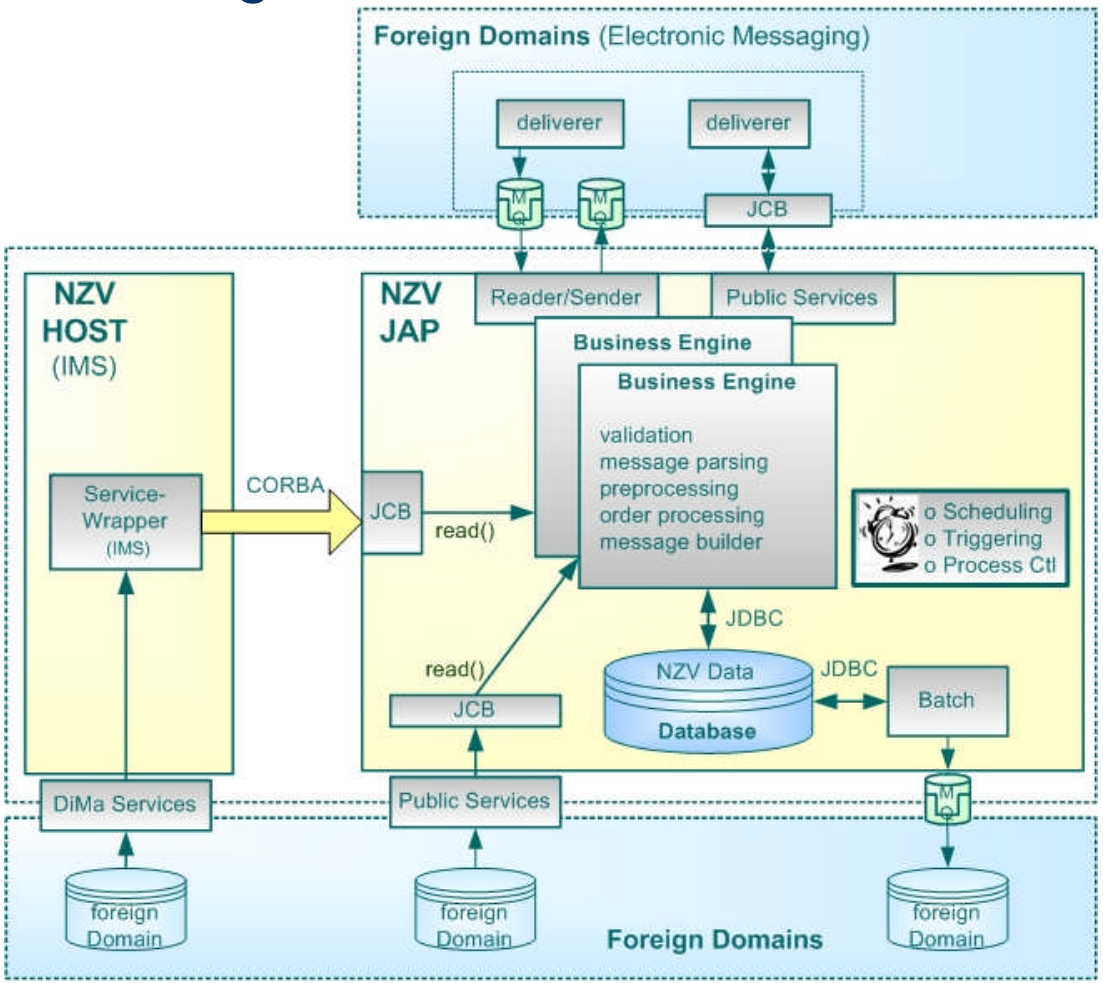
Transaction Processing

- We will use built in BEA WebLogic Transaction Monitor (based on BEA Tuxedo technology)
- Has been proven in a prototype in payments area (see next two slides)
- Many questions remain open, but those questions are more application than technology related:
  - What is the LUW (Logical Unit of Work) concept on the new platform. Concepts used on the mainframe no longer work in a distributed environment
  - How can we avoid sychronicity and use asynchronous communication instead?

Payments Platform Consolidation – Status Quo with main transaction processing on mainframe



NZV Platform Consolidation – Target Architecture with main transaction processing on JAP



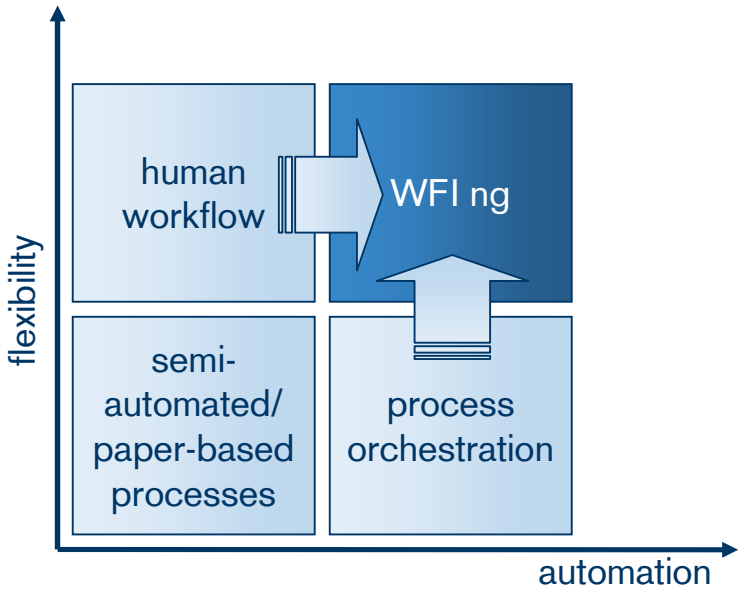
Flexibility and Reuse
WFI (Workflow Infrastructure) Next Generation

Challenges

- **Better integration** into JAP applications and portals (e.g., common worklist component)
- Need for high-throughput **process orchestration** in addition to human workflow
- Improved functionality for workflow modeling, monitoring, and analysis; **support for standards (BPEL)**
- Current workflow engine behind the Workflow Infrastructure (Websphere MQ Workflow) reaches end of life-cycle

Solution

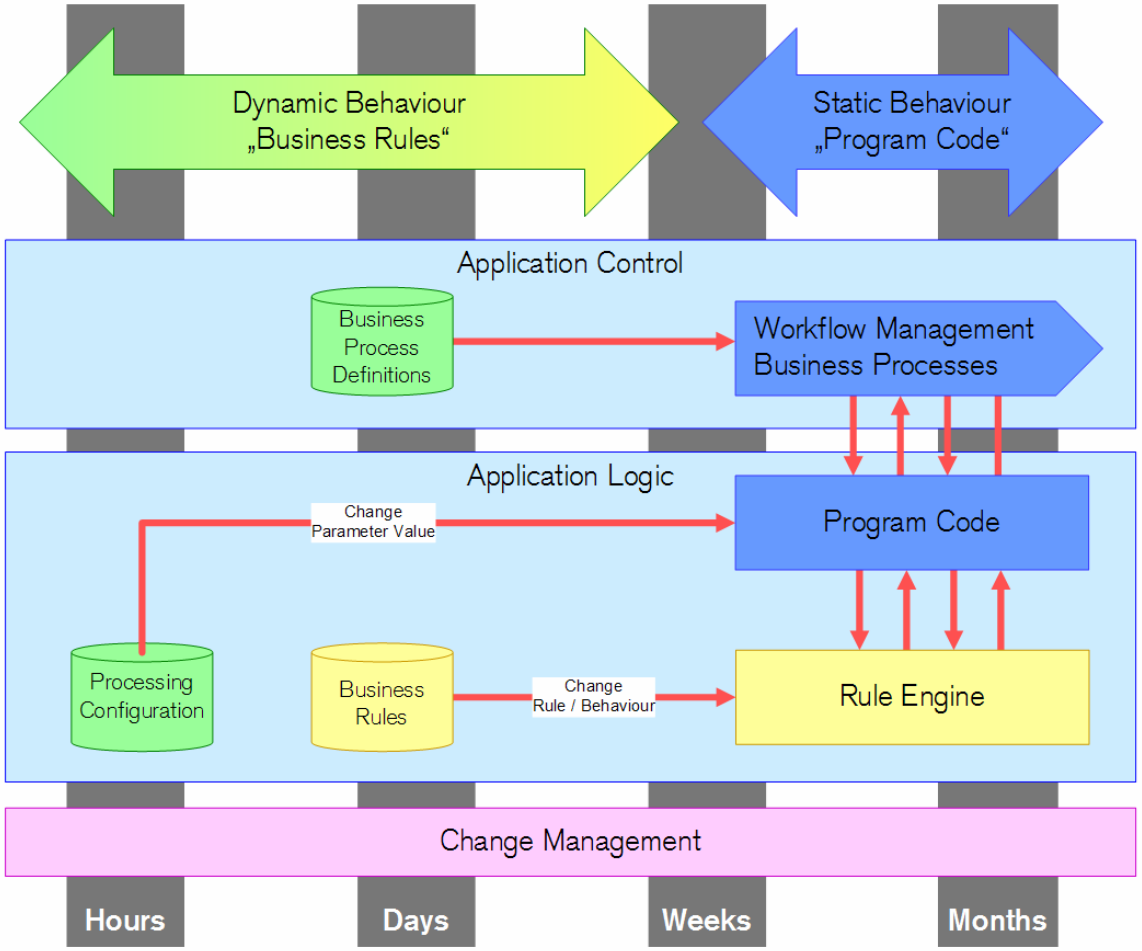
- State-of-the art BPM tool as a fundamental building block for
  - the future backend platform for our core applications
  - front components
- Process automation environment tightly integrated with JAP



⇨ **Improved support for business process automation and increased flexibility for the business**

⇨ **Processes are no longer hardcoded**

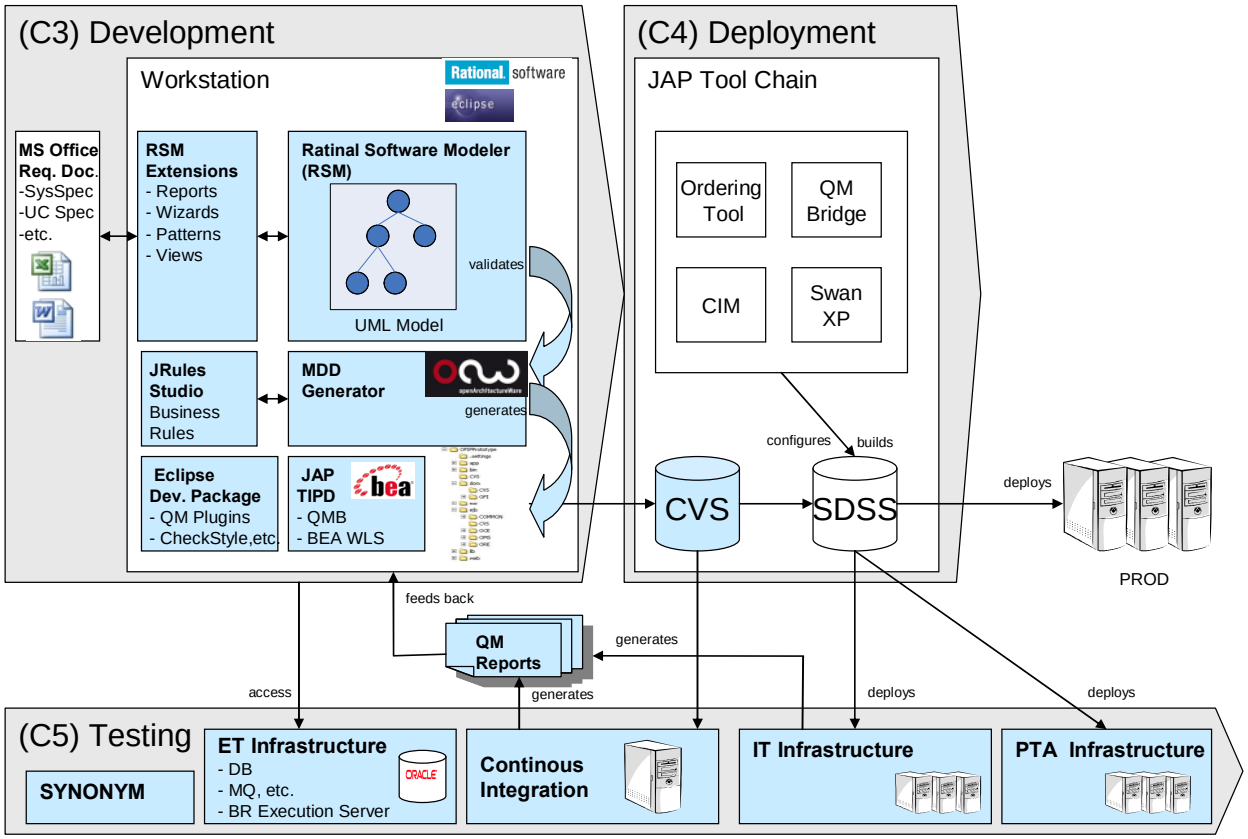
Business Rules vs. Code



- Business Rule Engine is part of the new platform
- Business Rule Engine can increase flexibility but is not the answer to all problems
- Move business rules, where applicable into specialized engine with eco system
- Longterm: Make business responsible for maintenance of business rules
- Shortterm: Rules are maintained by IT

Development Toolchain: Enhance efficiency

Development Tool Chain: Overview



Vision

- Be 5 times more efficient in development using model driven approaches
- This resolves potential resource problems and moves the effort (even more) from development into analysis and desgin

Challenges ahead (only some examples)

Defining and implementing the future platform requires to tackle a large number of challenges, for instance

- Application Architecture
  - Principles for application design with reduced LUW (Logical Unit of Work) scopes
  - Re-thinking of Batch net design in a distributed environment
  - Additional required checks and balances because of distribution
- Platform architecture
  - Provide all the necessary building blocks for transaction processing on JAP
  - TX manager, batch management, HA databases etc
- Integration architecture
  - Provide the concepts and means for reliable transaction processing in a distributed fashion
  - Rules for component design, service design, transaction control etc
  - Exception handling
- Security architecture
  - Efficient authorization and authentication in a component based world
  - Re-eavalute the current security mechanisms for the new environment
- Systems management architecture
  - Failure detection and analysis in a distributed component environment
  - Do we need more Business activity monitoring?
- Co-Existence
  - Since the old and the new platform will co-exist for many years, a concept on how to do this is key

Questions?