14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Renovation statt Abrissbirne

Sanfte Migration spart Kosten

Bruno Schäffer

Canoo Engineering AG

Agenda

- Motivation
- Business Case
- Technology
- Experience

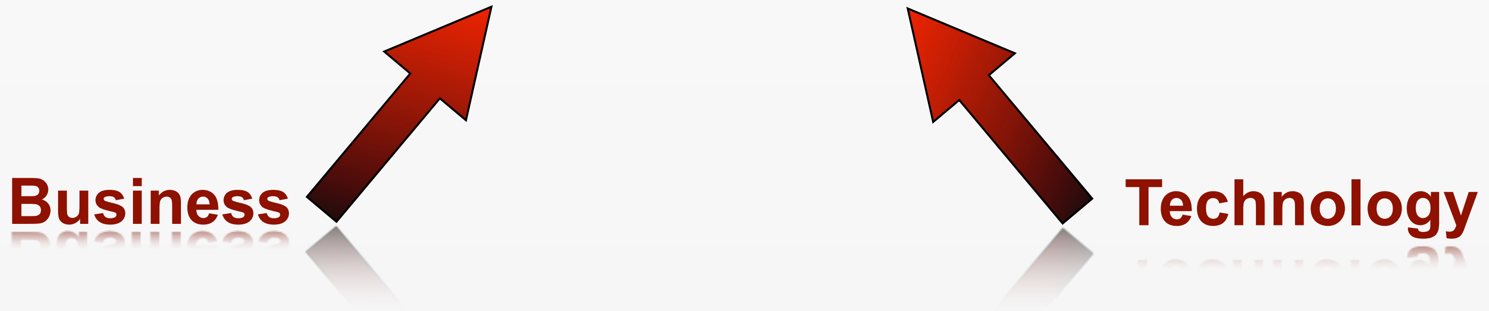
Motivation

- Migration of 4GL or 3270 applications
  - Applications for expert workers
  - What should the new application be based on?

- Web applications are too slow for expert workers (even with AJAX)

- Swing (or any other widget toolkit) is too low-level

- Developers should focus on what and not how



Affichage, APG, IT Dept.

Business: Outdoor advertising with posters in streets, train stations, parking lots, shopping centers, tourist resorts, airports; posters in and on vehicles

Affichage Holding (by Dec 2007)

Staff:	819
Turnover:	396’900’000 CHF

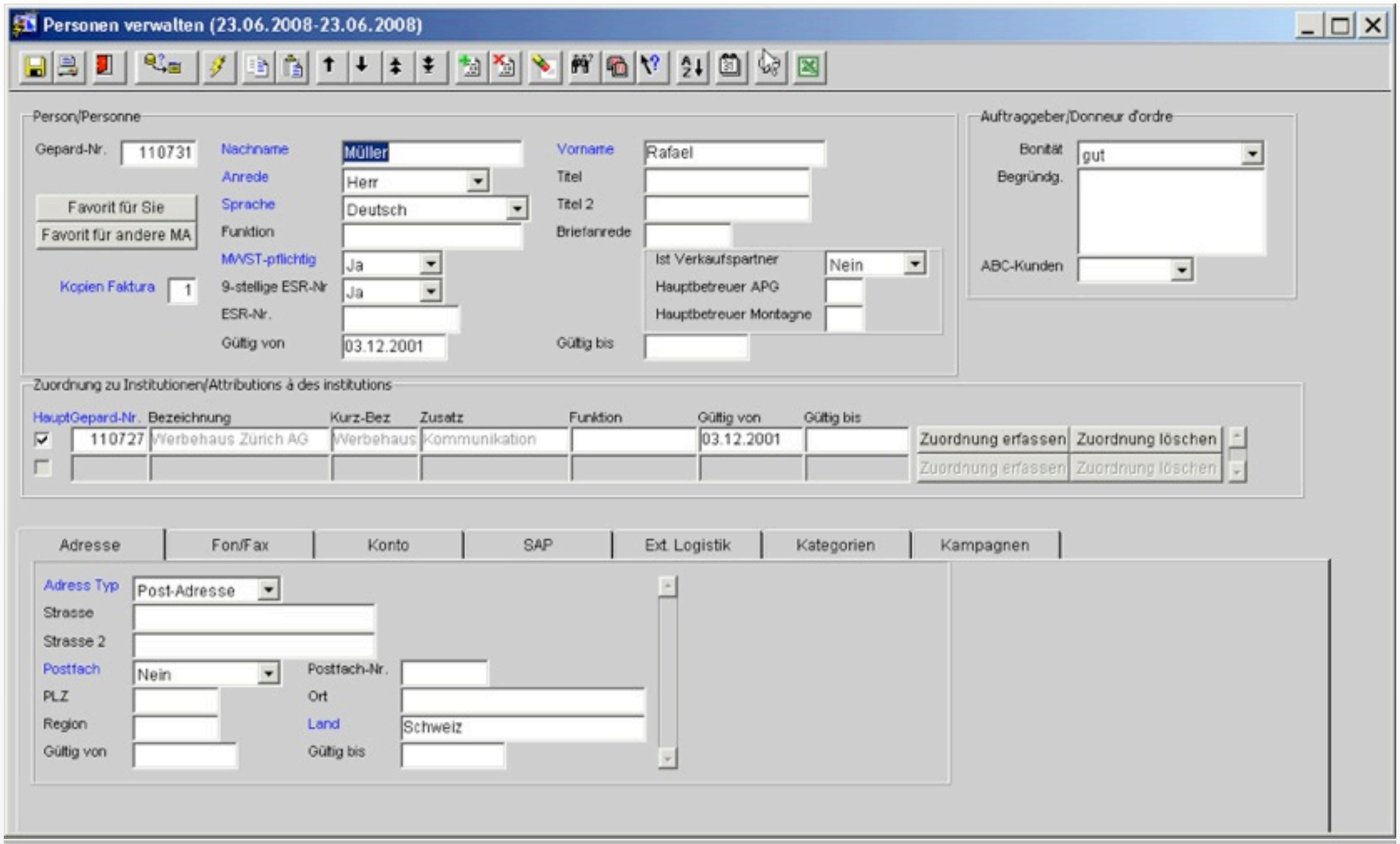
↳ **APG (by Dec 2007)**

Staff:	507
Billboards (rented 1-3 weeks):	75’700
Market share in Switzerland:	~ 70 %

↳ **APG IT Department (by Dec 2007)**

Development Staff:	20.6
Operations Staff:	7.9
Administration Staff:	2.2

From Beast



Personen verwalten (23.06.2008-23.06.2008)

Person/Personne

Gepard-Nr.	110731
Favorit für Sie	
Favorit für andere MA	
Kopien Faktura	1
Nachname	Müller
Anrede	Herr
Sprache	Deutsch
Funktion	
MWST-pflichtig	Ja
9-stellige ESR-Nr	Ja
ESR-Nr.	
Gültig von	03.12.2001
Vorname	Rafael
Titel	
Titel 2	
Briefanrede	
Ist Verkaufspartner	Nein
Hauptbetreuer APG	
Hauptbetreuer Montagne	
Gültig bis	

Auftraggeber/Donneur d'ordre

Bonität	gut
Begründg.	
ABC-Kunden	

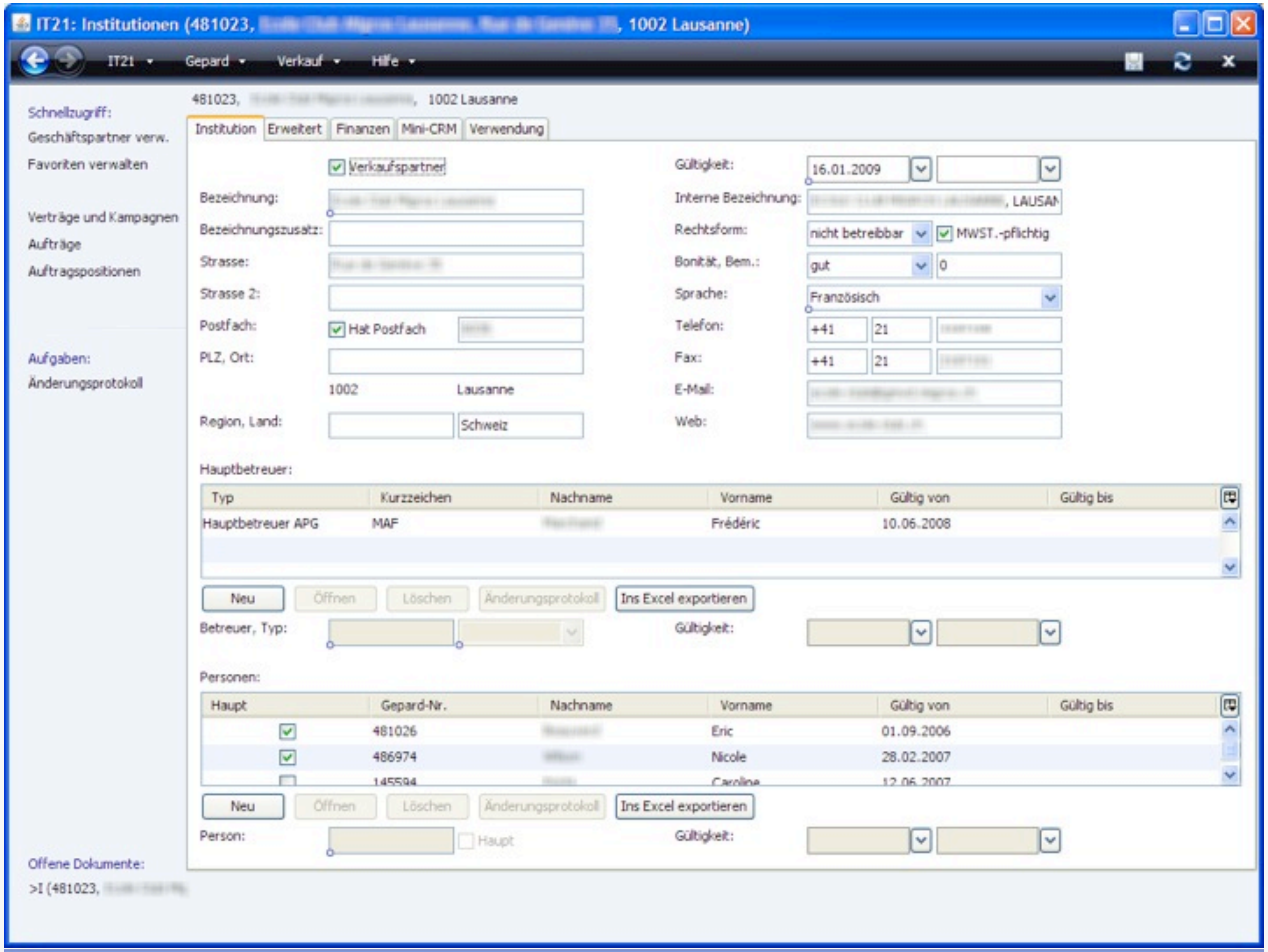
Zuordnung zu Institutionen/Attributions à des institutions

	HauptGepard-Nr.	Bezeichnung	Kurz-Bez	Zusatz	Funktion	Gültig von	Gültig bis		
☑	110727	Werbehaus Zürich AG	Werbehaus	Kommunikation		03.12.2001		Zuordnung erfassen	Zuordnung löschen
☐								Zuordnung erfassen	Zuordnung löschen

Adresse | Fon/Fax | Konto | SAP | Ext. Logistik | Kategorien | Kampagnen

Adress Typ	Post-Adresse
Strasse	
Strasse 2	
Postfach	Nein
PLZ	
Region	
Gültig von	
Postfach-Nr.	
Ort	
Land	Schweiz
Gültig bis	

To Beauty



IT21: Institutionen (481023, [redacted], 1002 Lausanne)

IT21 ▾ Gepard ▾ Verkauf ▾ Hilfe ▾

Schnellzugriff:
Geschäftspartner verw.
Favoriten verwalten

Verträge und Kampagnen
Aufträge
Auftragspositionen

Aufgaben:
Änderungsprotokoll

Offene Dokumente:
>I (481023, [redacted]

481023, [redacted], 1002 Lausanne

Institution | Erweitert | Finanzen | Mini-CRM | Verwendung

	✓ Verkaufspartner	Gültigkeit:	16.01.2009 ▾ / ▾
Bezeichnung:	[redacted]	Interne Bezeichnung:	[redacted], LAUSAN
Bezeichnungszusatz:		Rechtsform:	nicht betreibbar ▾ ✓ MWST.-pflichtig
Strasse:	[redacted]	Bonität, Bem.:	gut ▾ 0
Strasse 2:		Sprache:	Französisch ▾
Postfach:	✓ Hat Postfach [redacted]	Telefon:	+41 21 [redacted]
PLZ, Ort:	1002 Lausanne	Fax:	+41 21 [redacted]
		E-Mail:	[redacted]
Region, Land:	Schweiz	Web:	[redacted]

Hauptbetreuer:

Typ	Kurzzeichen	Nachname	Vorname	Gültig von	Gültig bis
Hauptbetreuer APG	MAF	[redacted]	Frédéric	10.06.2008	

Neu | Öffnen | Löschen | Änderungsprotokoll | Ins Excel exportieren

Betreuer, Typ: Gültigkeit:

Personen:

Haupt	Gepard-Nr.	Nachname	Vorname	Gültig von	Gültig bis
✓	481026	[redacted]	Eric	01.09.2006	
✓	486974	[redacted]	Nicole	28.02.2007	
☐	145594	[redacted]	Caroline	12.06.2007	

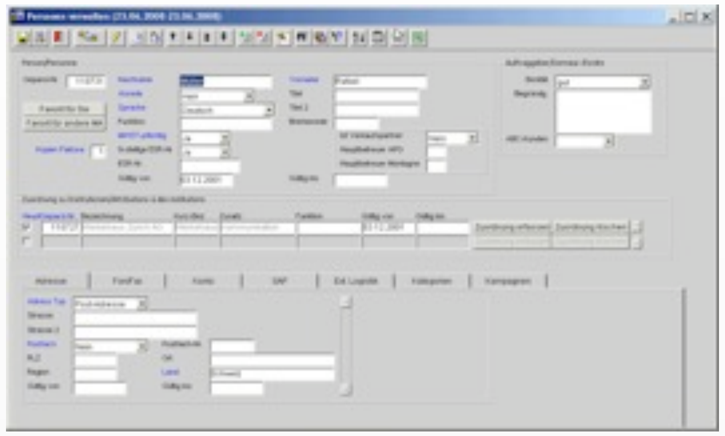
Neu | Öffnen | Löschen | Änderungsprotokoll | Ins Excel exportieren

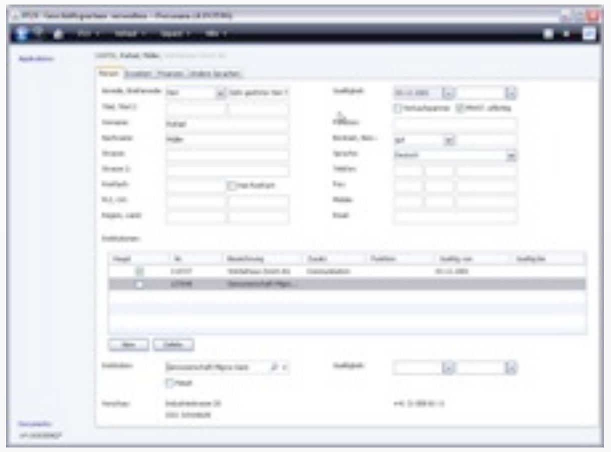
Person: ☐ Haupt Gültigkeit:

canoo

Business Case: new UI for Reservation System

- Billposting Business and IT
  - Processes are demanding and very specific
  - Data processing on the enterprise level offers operational benefits
  - Market offers no IT solutions for this business
  - ➜ Big players use proprietary solutions





- Reservation System IT21
  - Covers all relevant billposting processes: face acquisition and management, product management, sales, logistics, allowance payments
  - Built between 2001 and 2003 with an effort of some 43 PY in IT and project management
  - > 300 screens
  - Installed in Switzerland, Romania and Greece

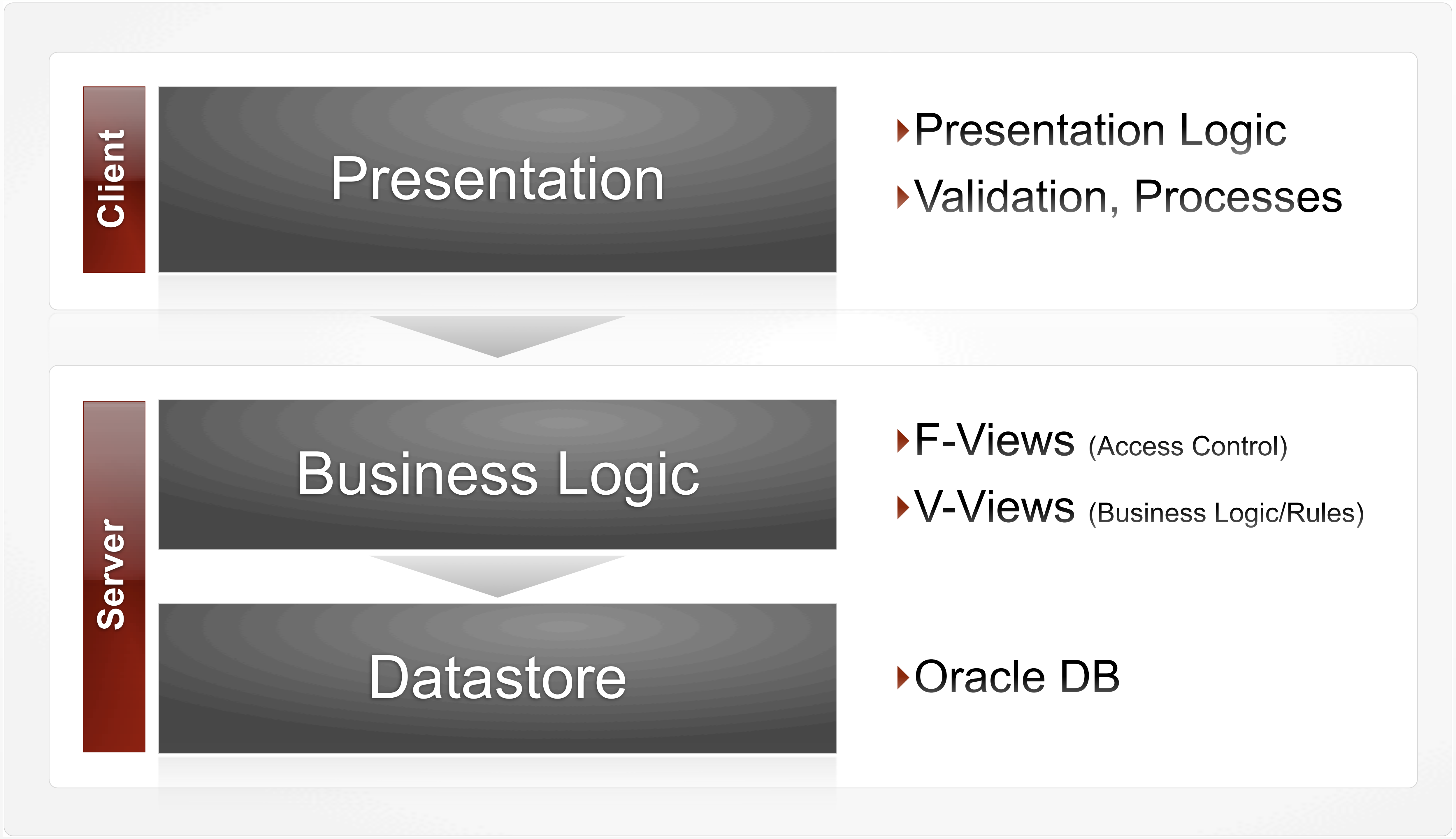
- **Problem:** Oracle Forms Client is discontinued

Requirements for Renovation

Development Efficiency	• Same speed for development and changes as with Oracle Designer/ Forms
User Interface Features	• At least equivalent to Oracle Designer/Forms • Option to extend Oracle Designer/Forms features with acceptable effort
Architecture	• Reuse of existing business logic (incl. authorization) with no change ➜ Two-Tier architecture stays in place • Support for reuse of larger UI building blocks • Browser support: not required (Citrix and Intranet only)
Technology	• Open, independent - no vendor hook in • Mature - and yet with viable future perspective • Widely used, community based - resources available
Development Environment	• Solid platform: allocation of responsibility, configuration management, testing etc.

Idea: Our Oracle Designer/Forms platform is highly standardised and uses a very restricted number of patterns. If we narrow down our requirements to that little and find the right abstractions we ought to be able to develop in a Java environment just as efficiently

Architecture – Old



- Presentation Logic
- Validation, Processes
- F-Views (Access Control)
- V-Views (Business Logic/Rules)
- Oracle DB

Platform Options

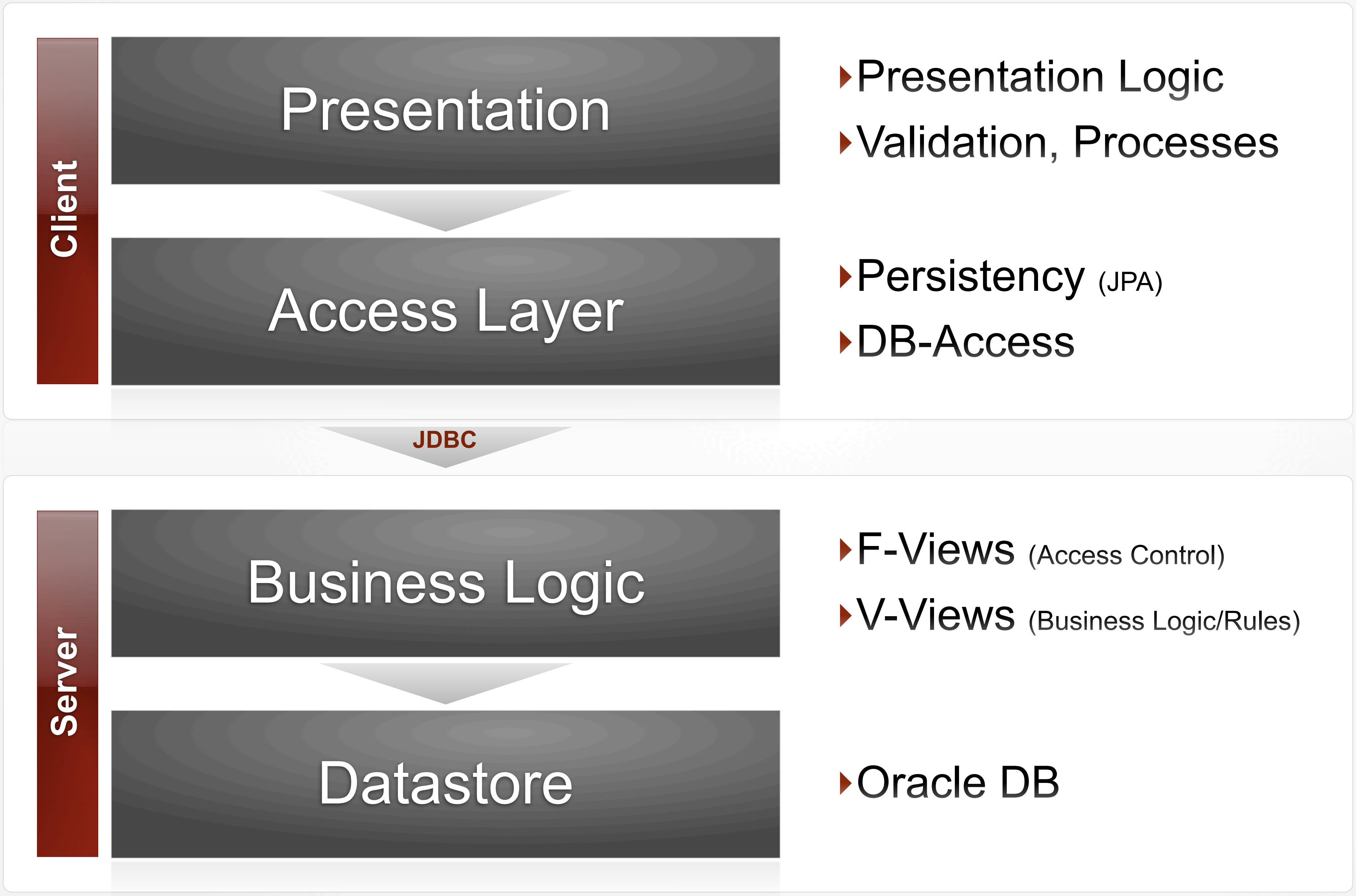
- Java (no need to establish a third technology besides Java and PL/SQL)
- Prototyping with Oracle ADF proved too complex
- AJAX was considered too complex

Mix and Match of Java Technologies

Principles, Components, and Players

- Principles
  - Make everyday things fast and easy
  - Make sophisticated and complex things still possible
  - Flat learning curve
  - Programmatic approach (complemented by generative/descriptive bits and pieces)
  - Do not touch server-side business logic and DB
- Components
  - Spring (core, JDBC, template, ORM[JPA/Hibernate], test)
  - Swing, Swing X, JGoodies Smart Client (forms, binding, validation, looks, ...), Jemmy
  - Log4J, EasyMock, c3pO, common bean utils
- Players
  - APG, Canoo, SpringSource (J. Höller), JGoodies (K. Lentzsch), openArchitectureWare (S. Efftinge)

Architecture - New



- Presentation Logic
- Validation, Processes
- Persistency (JPA)
- DB-Access
- F-Views (Access Control)
- V-Views (Business Logic/Rules)
- Oracle DB

Persistency I

- Persistence layer:
  - Data Access Objects (DAO)
  - Entirely generated
  - Can deal with type hierarchies
- Domain Specific Language (DSL)
  - Customized for architecture/application domain
  - Allows to extend the DB metainformation

```
entity BankenstaemmeF (id = (bstmId) sequenceName = BSTM_SEQ) {
   manyToOne LaenderBsF land (joinColumns = LAND_ID)
   notNull Number invalid (castTo = Boolean)
   notNull Number eingelesen (castTo = Boolean)
   Number postkontoBank (length = 11) // extended due to formatting with "-"
}
```

Persistency II

- Data Access Objects
  - No separate DTO (DTO is merged with DAO)
  - Developer only deals with properties
- openArchitectureWare:
  - Meta MDA technology for Eclipse: framework for developing MDA tools
  - Transforms model into DAO and descriptor
  - Generates JUnit stubs and skeletons for testing DAO

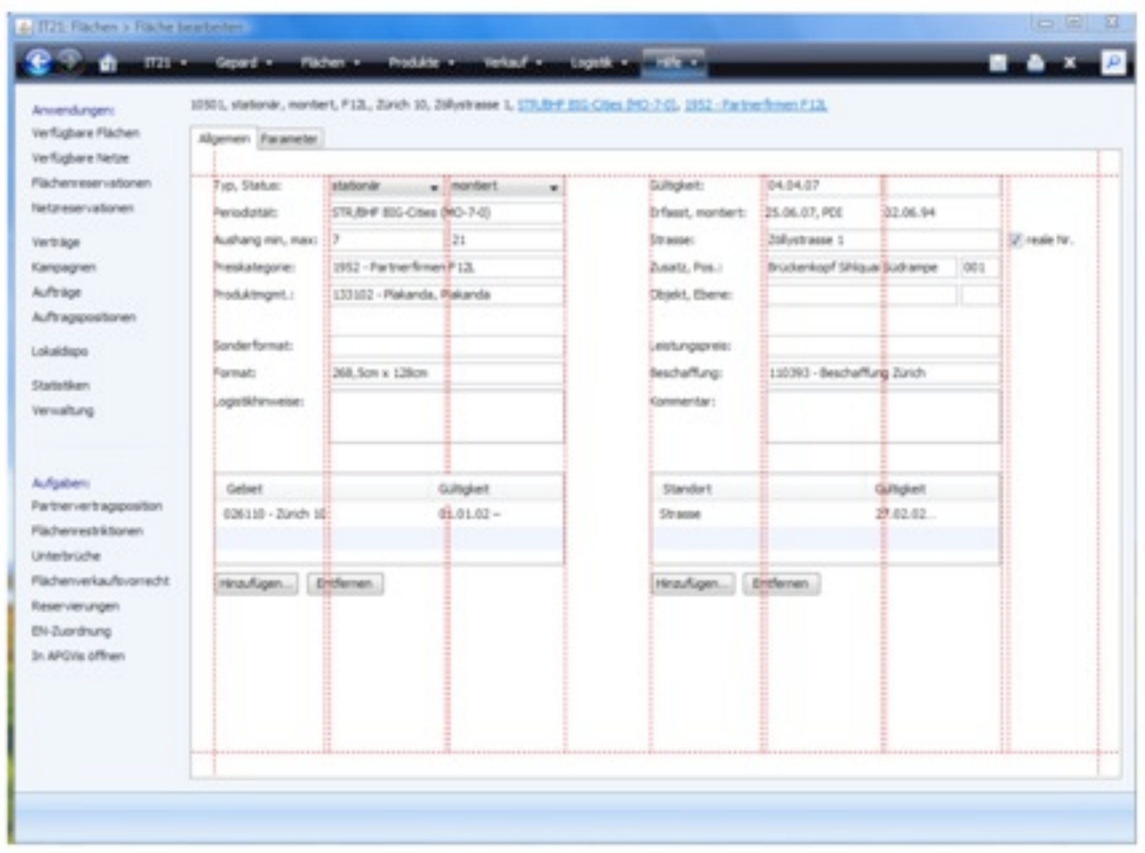
canoo

Persistency III

- Metainformation
  - Annotations in DAO
    - Used by JPA/Hibernate and the UI framework
  - Descriptors
    - Exposes DAO/DTO metainformation in a developer friendly way (i.e. as a Java class)
    - Framework makes use of this metainformation
    - Developers usually do not access metainformation
    - Type completion and compile time type checking
- Developers don’t have to care about transactions
  - Infrastructure handles transaction context
  - Transaction context is established per panel

Meta Design

- Developers are not designers!
  - Developers are not responsible for visual or interaction design
  - Developers should focus on content
- Plan the overall design upfront
- As rigid as possible, as flexible as needed
- Design is “hard-coded” in the framework!
- Elements of Meta Design
  - Typography, colors, sizes
  - Contrast, balance
  - Layout, order, gaps



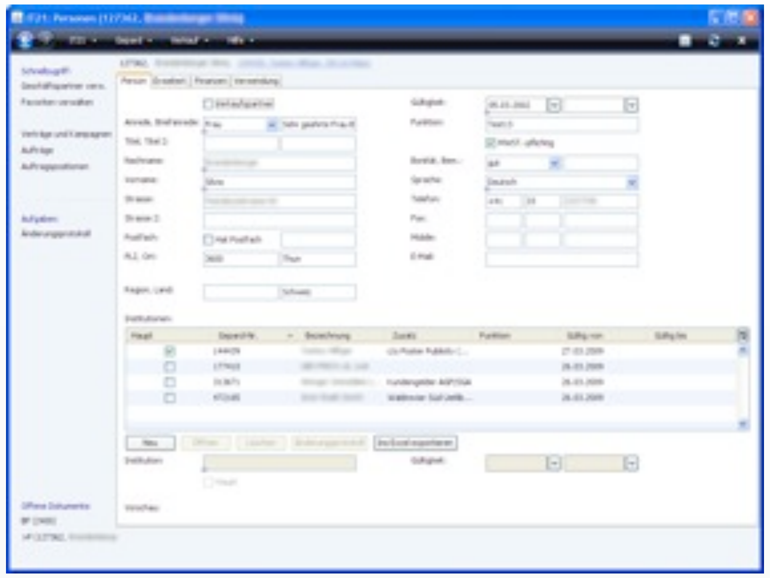
10501, stationär, montiert, F12L, Zürich 10, Zollystrasse 1, STR,BHF BIG-Cities (MO-7-0), 1952 - Partnerfirmen F12L

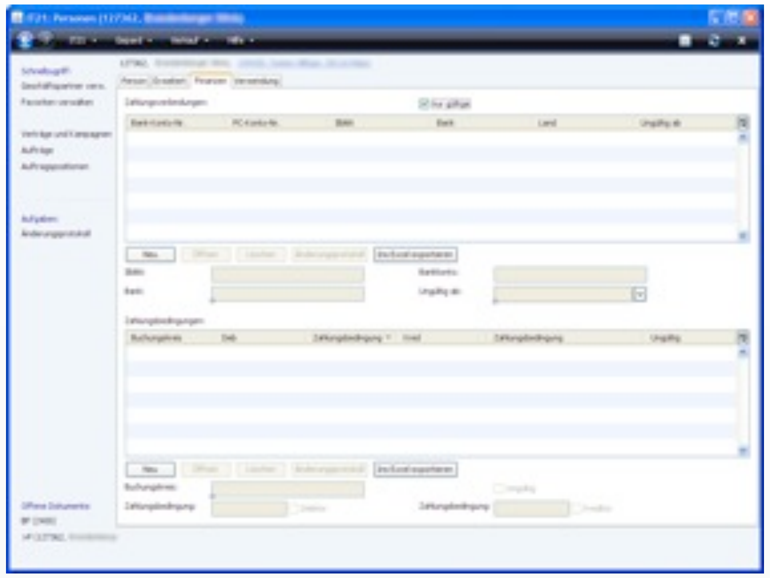
Typ, Status:	stationär	montiert	Gültigkeit:	04.04.07		
Periodizität:	STR,BHF BIG-Cities (MO-7-0)		Erfasst, montiert:	25.06.07, PDI	02.06.94	
Aushang min, max:	7	21	Strasse:	Zollystrasse 1		✓ reale Nr.
Preiskategorie:	1952 - Partnerfirmen F12L		Zusatz, Pos.:	Brückenkopf Sihlquai Südrampe	001	
Produktmgmt.:	133102 - Plakanda, Plakanda		Objekt, Ebene:			
Sonderformat:			Leistungspreis:			
Format:	268,5cm x 128cm		Beschaffung:	110393 - Beschaffung Zürich		
Logistikhinweise:			Kommentar:			

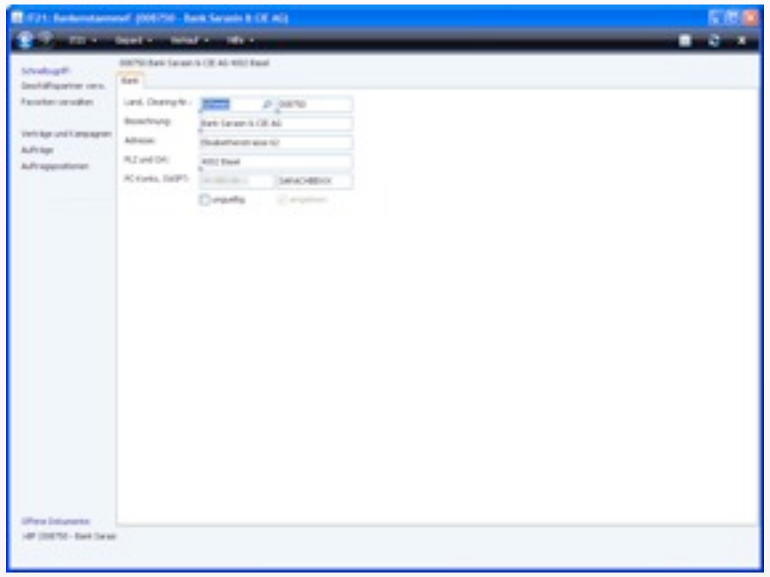
Gebiet	Gültigkeit
026110 - Zürich 10	01.01.02 –

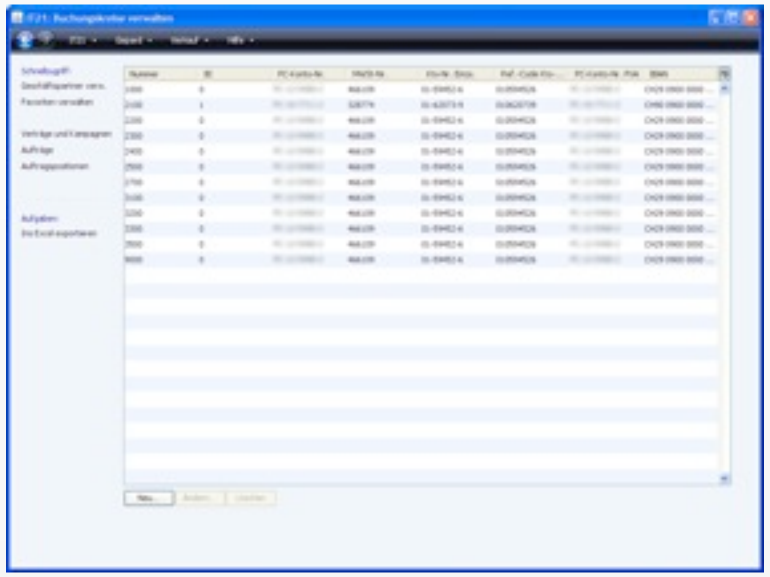
Standort	Gültigkeit
Strasse	27.02.02...

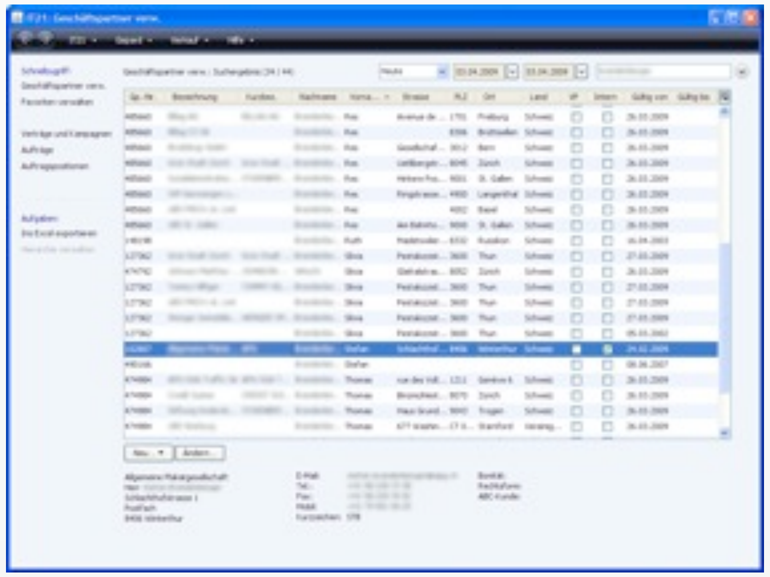
Templates





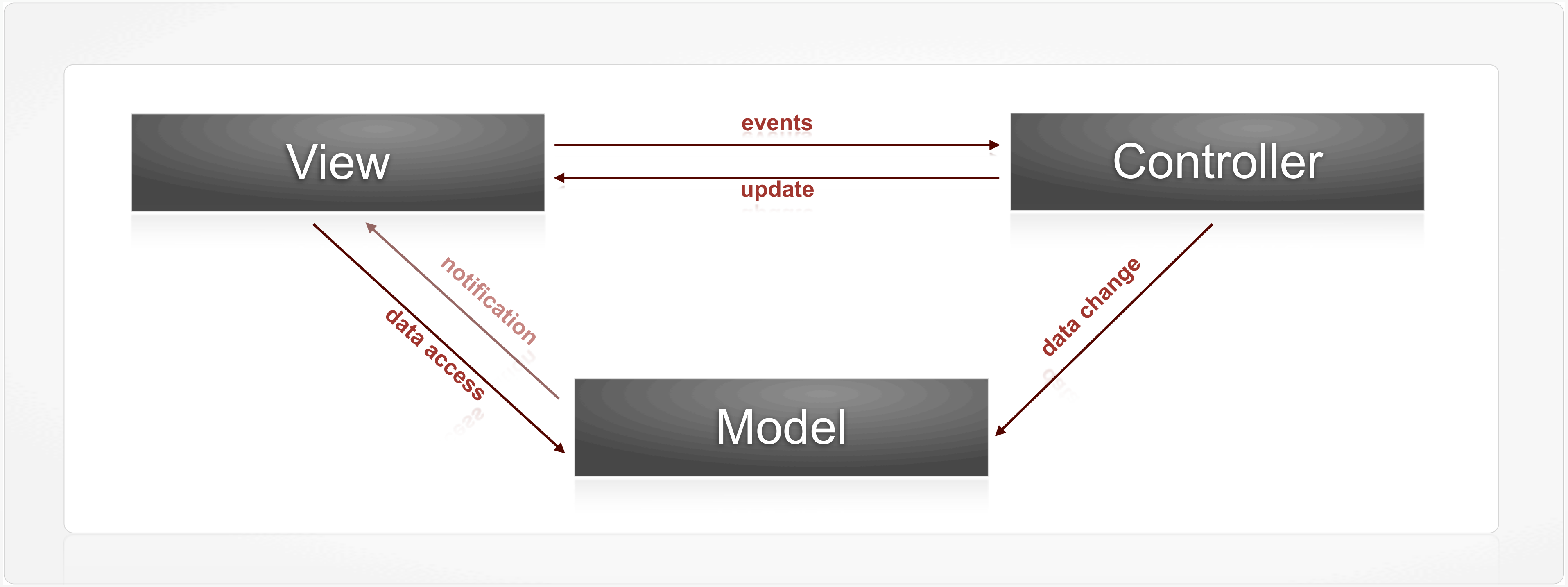




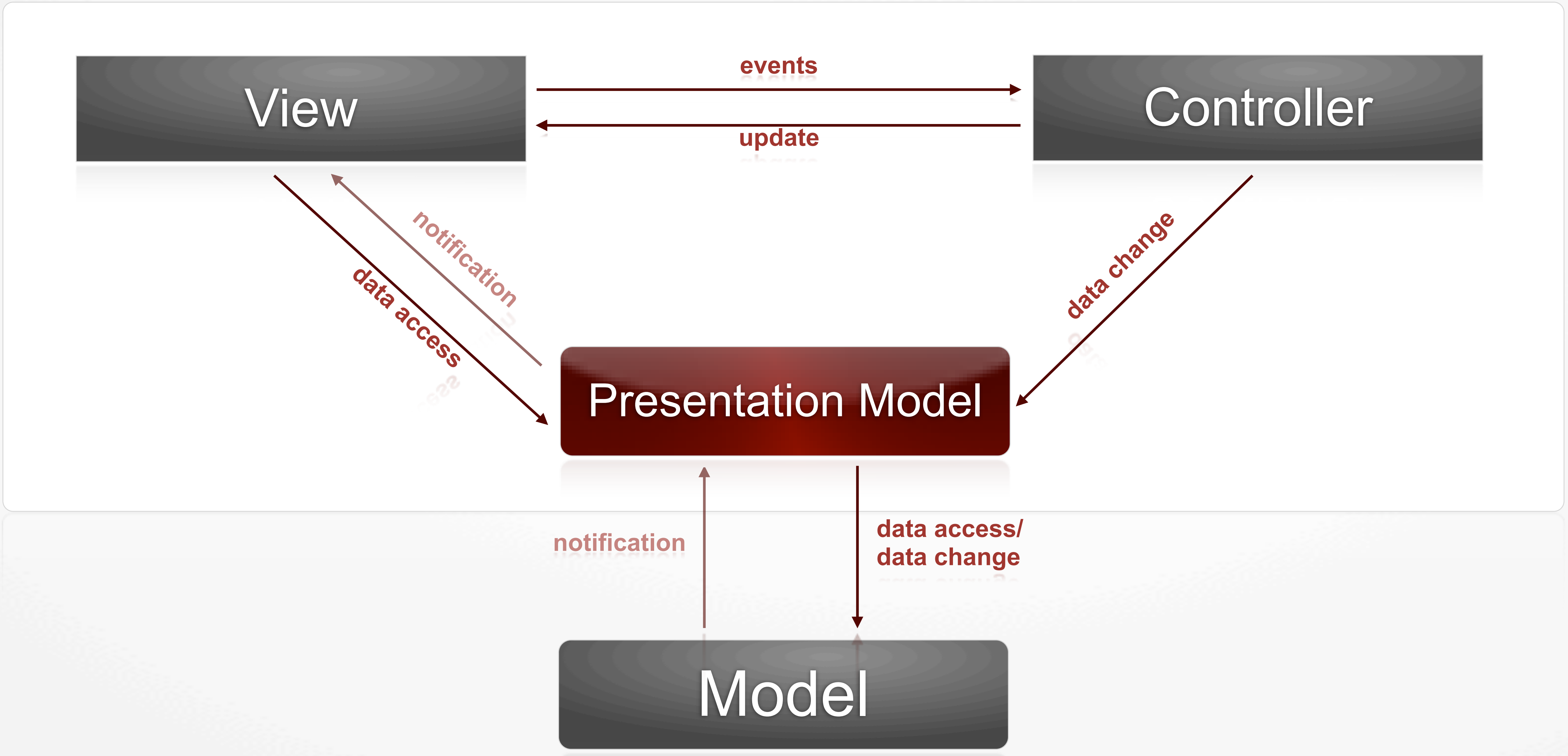


Presentation Model I

- MVC is a great design pattern for user interface components
- MVC is not so great as an architectural pattern for rich user interfaces
  - What is the model in a form-based application?
  - View code gets overly complex
  - Testing is challenging



Presentation Model II



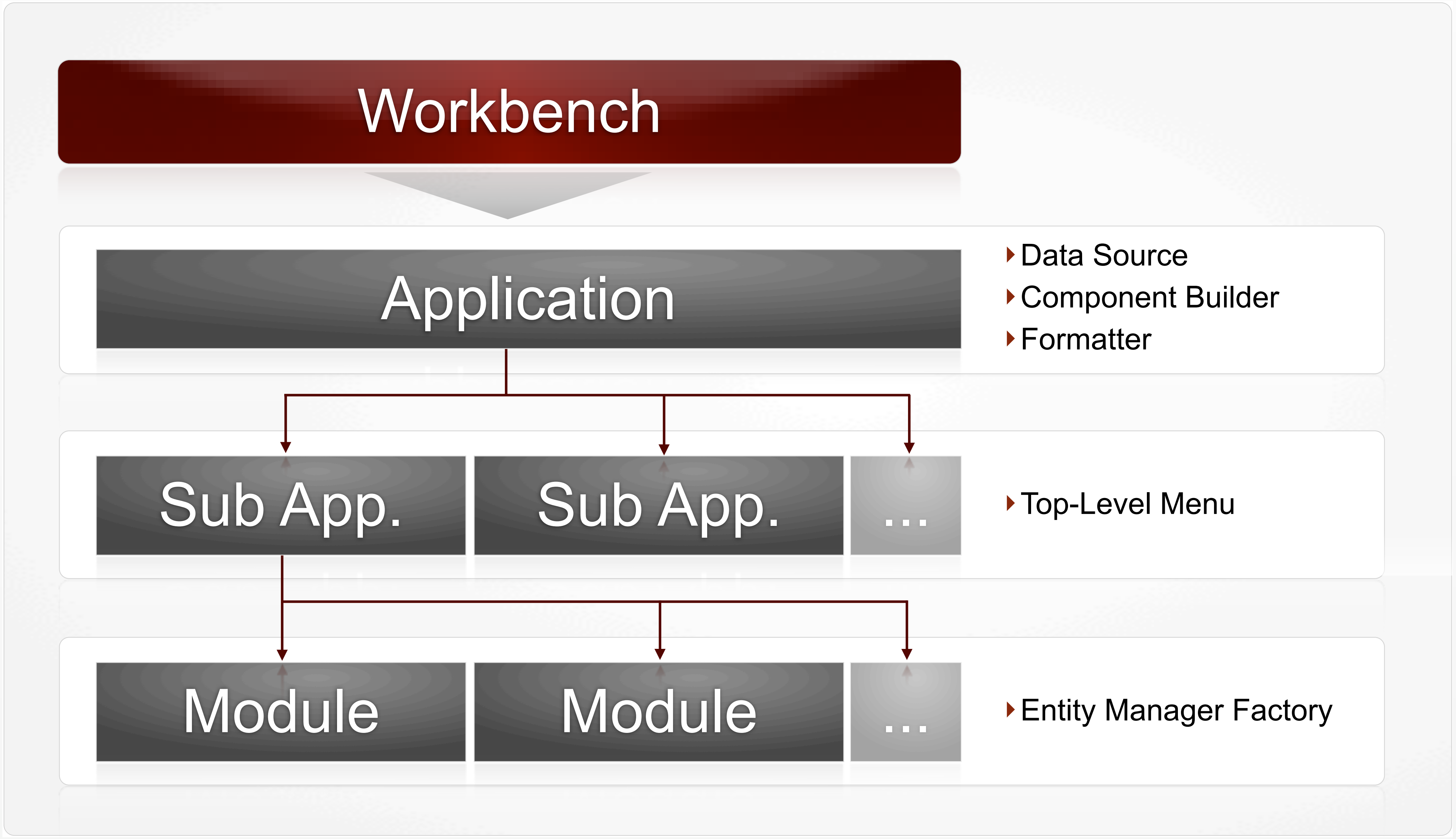
canoo

Presentation Model III

- Separate rendering code from presentation logic/state
- Presentation model
  - Takes care of entire presentation logic/state
  - Presentation state: all state that modifiable
  - Serves as an adapter to business objects
- View
  - Still keeps entire UI state
  - Component creation, layouting, event delegation
  - Observes presentation model
  - Simple code
- Testing
  - Unit testing for presentation model
  - View code testing rarely needed

UI Framework / Infrastructure I

- Application configuration
  - Spring and Java-based configuration



UI Framework / Infrastructure II

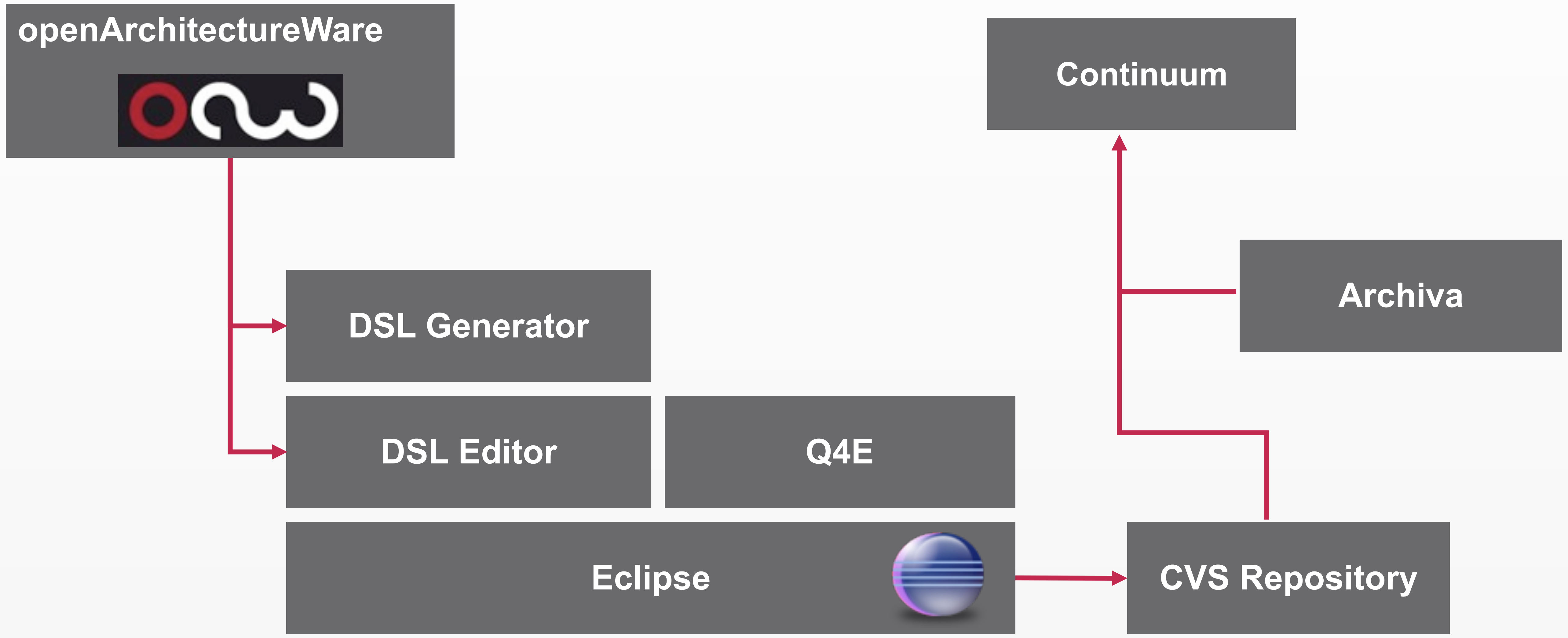
- Codifies meta design
  - Overall UI structure (➜ application configuration)
  - Layout class abstracts from layout details
  - Layout can be adapted (if required)
  - No visual builder!

- Component factory
  - UI components are customized to domain types
  - Binding is established
  - DB meta information is added

UI Framework / Infrastructure III

- Presentation Model
  - Generic presentation model for simple forms
  - Developer can provide a presentation model of her own
- Varia
  - Window, document, and menu management
  - Error handling, ...
- Developer does not need to know DAO details
  - Only DAO class and descriptors are required by a developer

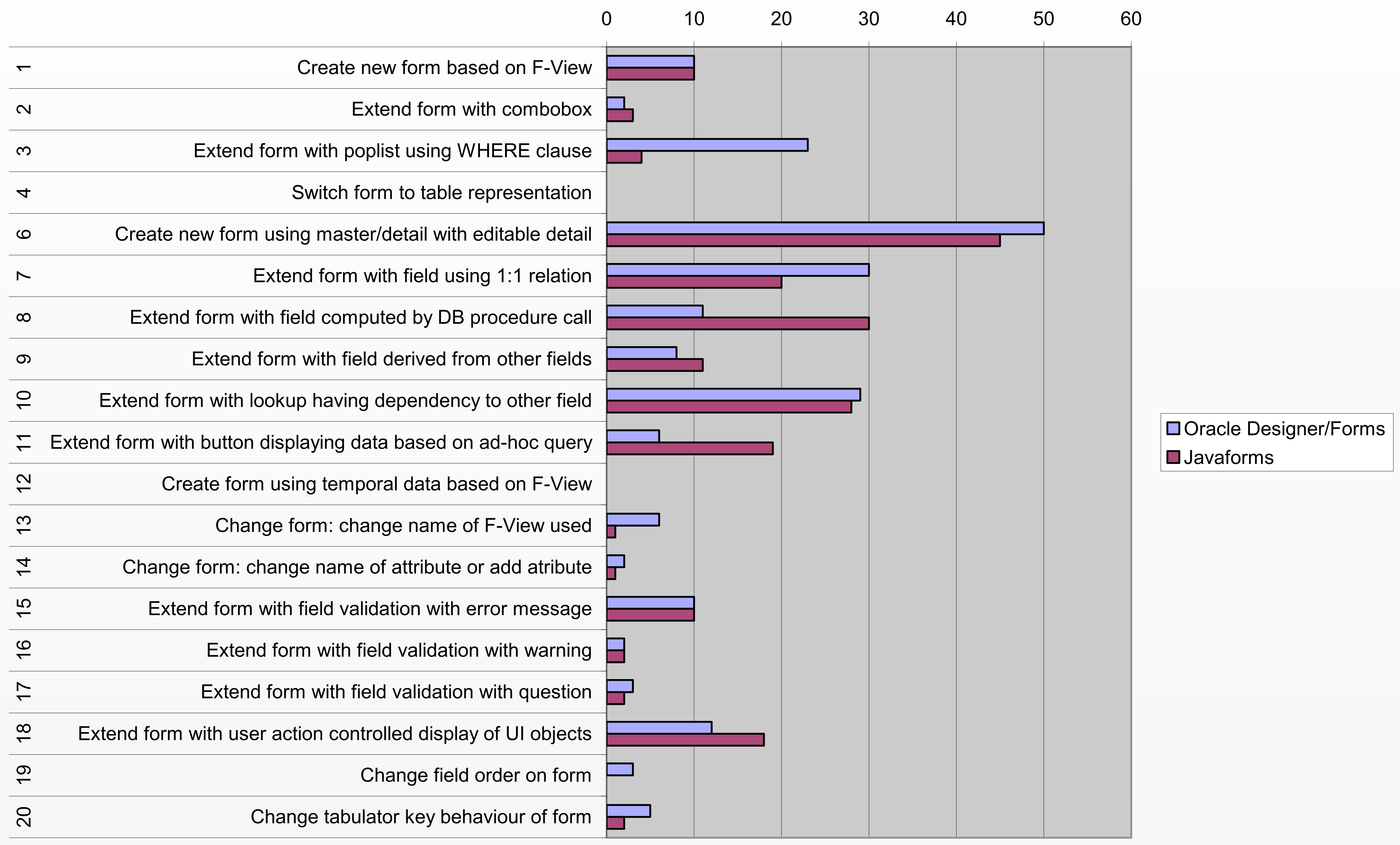
Development Environment



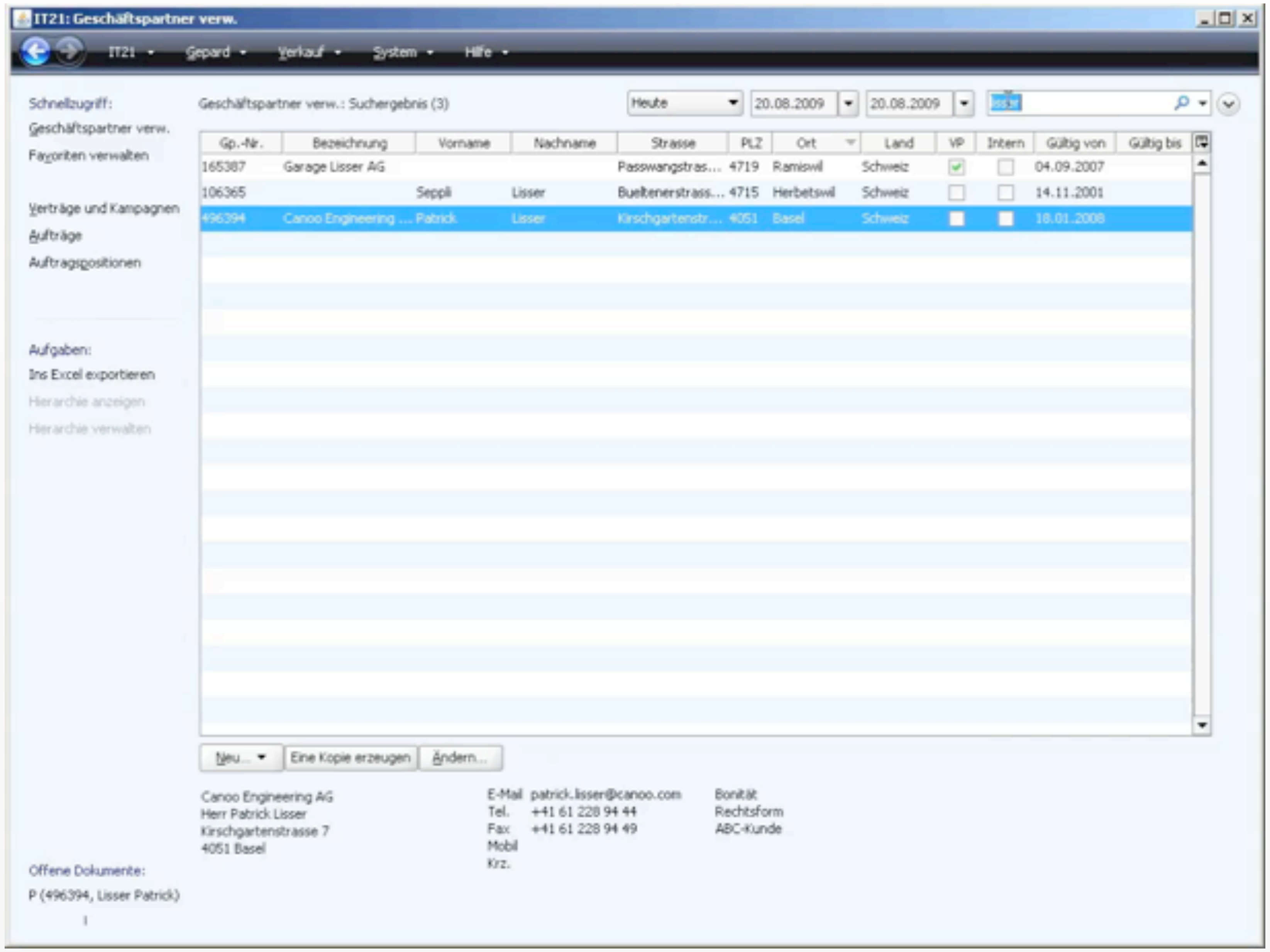
Development Model

- Modules
  - Can be assigned independently to developers
  - Configuration assembles a sub-application from modules
  - Maven is crucial for managing module (version) dependencies
- Testing
  - JUnit tests for DAO + DB functionality (based on Spring test)
  - JUnit and Jemmy for UI
    - JUnit for testing complex presentation models
    - JUnit & Jemmy for functional testing
  - EasyMock for stubbing

Development Efficiency Benchmark



Demo



Schnellzugriff:
Geschäftspartner verw.
Favoriten verwalten

Verträge und Kampagnen
Aufträge
Auftragspositionen

Aufgaben:
Ins Excel exportieren
Hierarchie anzeigen
Hierarchie verwalten

Offene Dokumente:
P (496394, Lisser Patrick)

Geschäftspartner verw.: Suchergebnis (3)

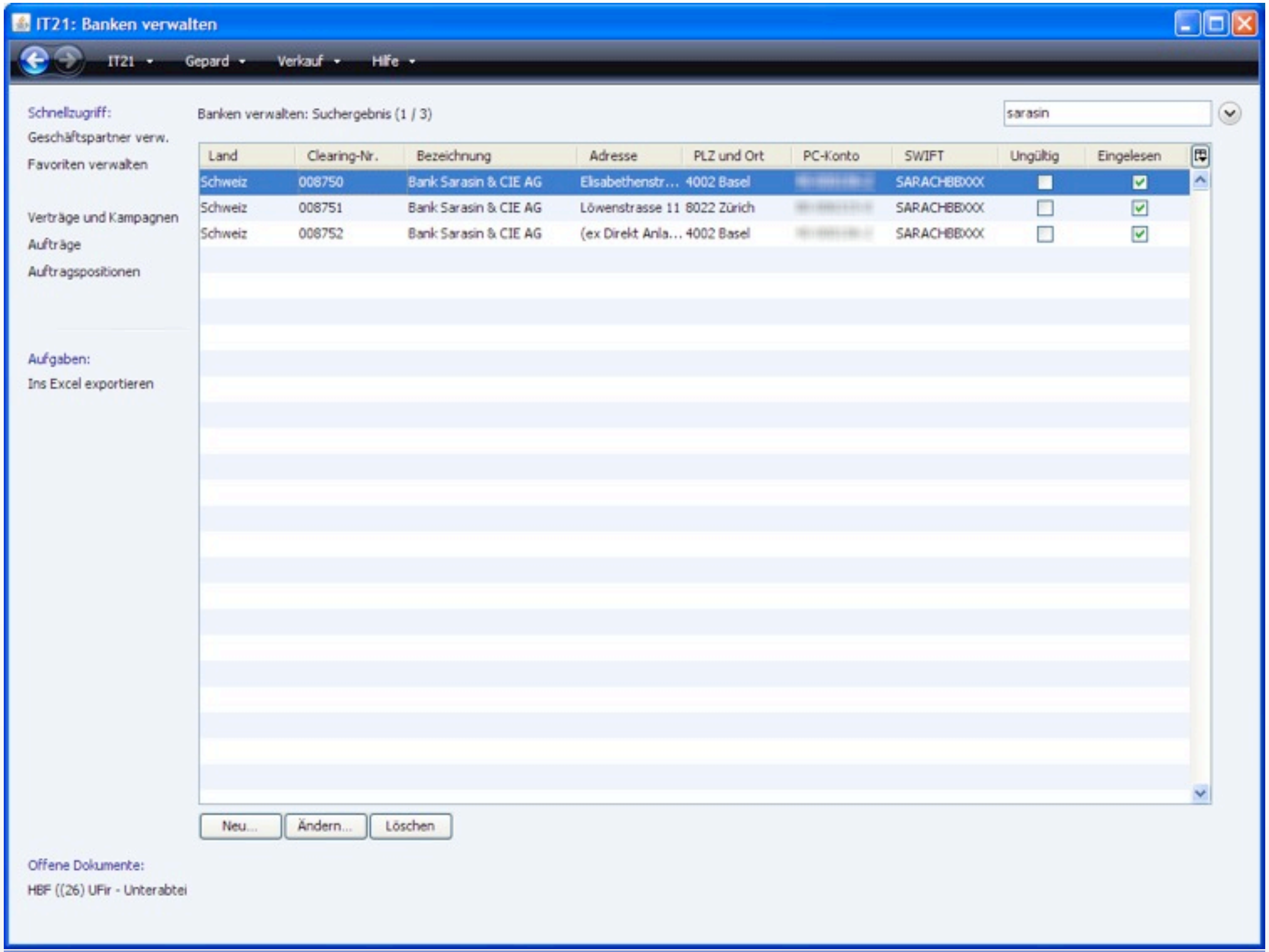
Gp.-Nr.	Bezeichnung	Vorname	Nachname	Strasse	PLZ	Ort	Land	VP	Intern	Gültig von	Gültig bis
165387	Garage Lisser AG			Passwangstras...	4719	Ramiswil	Schweiz	✓	☐	04.09.2007	
106365		Seppli	Lisser	Bueltenerstrass...	4715	Herbetswil	Schweiz	☐	☐	14.11.2001	
496394	Canoo Engineering ...	Patrick	Lisser	Kirschgartenstr...	4051	Basel	Schweiz	☐	☐	18.01.2008	

Canoo Engineering AG
Herr Patrick Lisser
Kirschgartenstrasse 7
4051 Basel

E-Mail	patrick.lisser@canoo.com
Tel.	+41 61 228 94 44
Fax	+41 61 228 94 49
Mobil	
Krz.	

Bonität
Rechtsform
ABC-Kunde

Example – Home Panel



Schnellzugriff:
Geschäftspartner verw.
Favoriten verwalten

Verträge und Kampagnen
Aufträge
Auftragspositionen

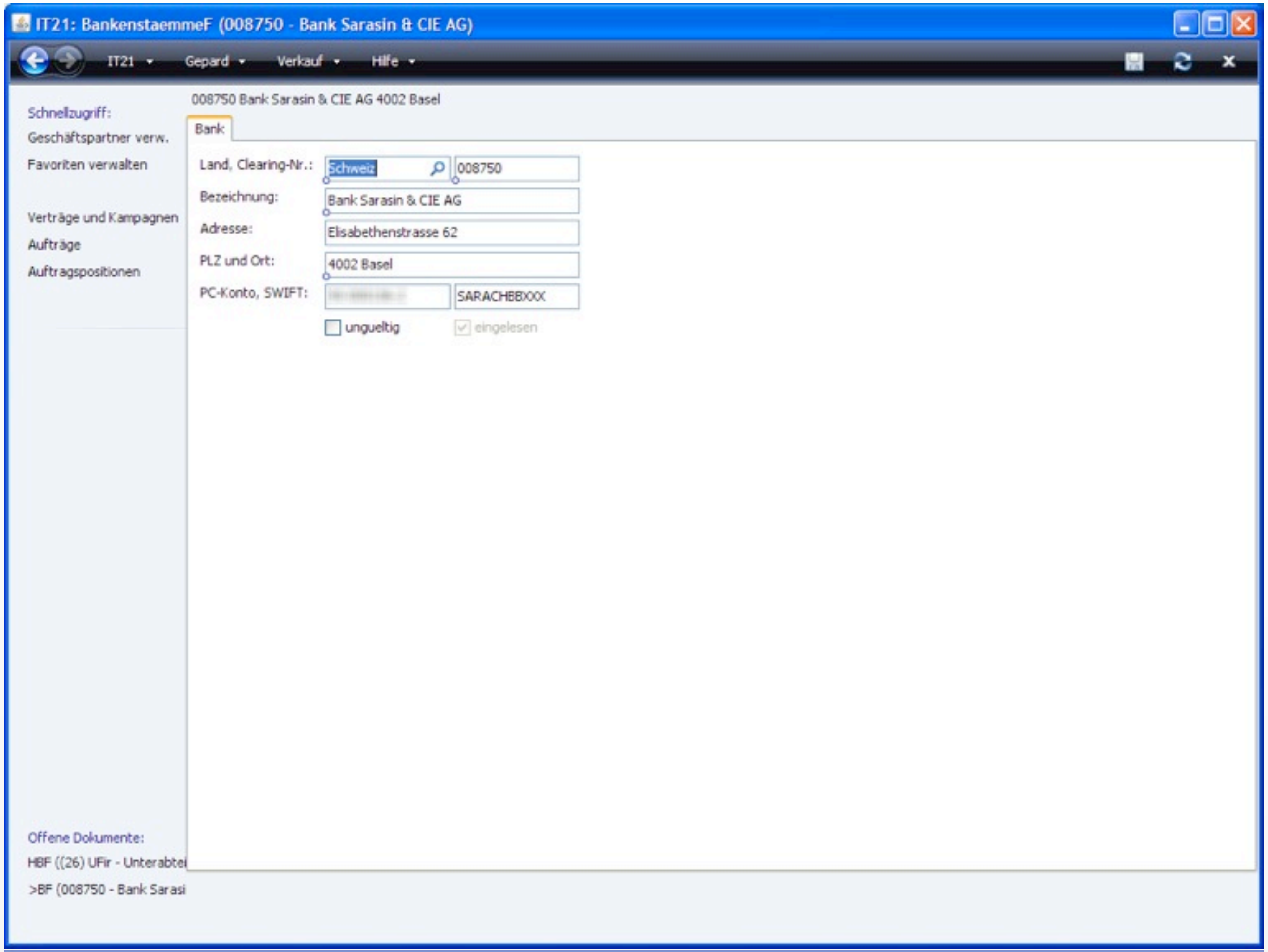
Aufgaben:
Ins Excel exportieren

Offene Dokumente:
HBF ((26) UFir - Unterabtei

Banken verwalten: Suchergebnis (1 / 3)

Land	Clearing-Nr.	Bezeichnung	Adresse	PLZ und Ort	PC-Konto	SWIFT	Ungültig	Eingelesen
Schweiz	008750	Bank Sarasin & CIE AG	Elisabethenstr…	4002 Basel	[redacted]	SARACHBBXXX	☐	✓
Schweiz	008751	Bank Sarasin & CIE AG	Löwenstrasse 11	8022 Zürich	[redacted]	SARACHBBXXX	☐	✓
Schweiz	008752	Bank Sarasin & CIE AG	(ex Direkt Anla…	4002 Basel	[redacted]	SARACHBBXXX	☐	✓

Example – Detail Form



008750 Bank Sarasin & CIE AG 4002 Basel

Land, Clearing-Nr.:	Schweiz	008750
Bezeichnung:	Bank Sarasin & CIE AG	
Adresse:	Elisabethenstrasse 62	
PLZ und Ort:	4002 Basel	
PC-Konto, SWIFT:	[redacted]	SARACHBBXXX
	☐ ungueltig	✓ eingelesen

Example – Home Panel

```
public class BankenHomePanel extends AbstractSearchHomePanel<BankenHomePanelModel> {
 public BankenHomePanel() {
  super(BankenPreview.class);
 }

 @Override
 public JTable buildResultsTable(final TableBuilder builder) {
  builder.addColumn(BankenstaemmeF.DESC.land().landbezeichnung()).setPrototypeValue("xxxxxxxxxx");
  builder.addColumn(BankenstaemmeF.DESC.bankIdentifikator());
  //dito for additional columns
  return builder.getTable();
 }

 @Override
 protected BankenHomePanelModel createHomePanelPresentationModel(ModuleConfiguration
moduleConfiguration, Conversation conversation) {
  return new BankenHomePanelModel(moduleConfiguration, conversation);
 }
}
```

Example - Home Panel Model

```
public class BankenHomePanelModel extends AbstractSearchHomePanelPresentationModel<BankenstaemmeF>
{

 public BankenHomePanelModel(final ModuleConfiguration moduleConfiguration, final Conversation
conversation) {
  super(moduleConfiguration, conversation, BankenstaemmeF.class, false);

  getSearchableProperties().remove(BankenstaemmeF.DESC.bankSchluessel());
  getSearchableProperties().remove(BankenstaemmeF.DESC.bstmId());
  getSearchableProperties().remove(BankenstaemmeF.DESC.jVersion());
  addEntityFactory(new DefaultEntityFactory("Bankenstamm", BankenstaemmeF.class));
 }

 @Override
 public String getQuickSearchWhereClause(final String quickSearchString) {
  //Assemble where clause for quick search
  }
 }

 @Override
 public String getOrderByClause() {
  return "bankIdentifikator";
 }

 @Override
 public int getNumberOfHitsLimit() {
  return 9999;
 }
}
```

Example - Detail Form

```
public class BankenForm extends AbstractForm<BankenstaemmeF> {

 @Override
 protected Component createHeader(final HeaderBuilder headerBuilder) {
  final BankenstaemmeFDesc desc = BankenstaemmeF.DESC;
  headerBuilder.addText(new MessageFormat("{0} {1} {2}"), desc.bankIdentifikator(),
desc.geldinstName(), desc.ort());
  return headerBuilder.getHeader();
 }

 @Override
 protected void initSubForms() {
  addSubForm(BankenSubForm.class);
 }

 @Override
 protected FormPresentationModel<BankenstaemmeF> createFormPresentationModel(
   final ModuleConfiguration moduleConfiguration, final Conversation conversation,
   final TemporalContext temporalContext, final AbstractDescriptor<BankenstaemmeF> descriptor) {
  return new BankenModel(moduleConfiguration, conversation, temporalContext, descriptor);
 }
}
```

Example - Detail Form Model

```
public class BankenModel extends FormPresentationModel<BankenstaemmeF> {

 public BankenModel(final ModuleConfiguration moduleConfiguration, final Conversation conversation,
   final TemporalContext temporalContext, final AbstractDescriptor<BankenstaemmeF> descriptor) {
  super(moduleConfiguration, conversation, temporalContext, descriptor);
 }

 @Validate
 public void checkPostkontoBank() {
  if (getBean().getPostkontoBank() != null
    && (getBean().getPostkontoBank().toString().length() < 4 || !(CheckUtils
          .checkPcKontoNrPruefziffer(getBean().getPostkontoBank())))) {
   message("validation.pckonto.msg", Severity.ERROR, BankenstaemmeF.DESC.postkontoBank());
  }
 }
}
```

Example - Subform

```
public class BankenSubForm extends AbstractSubForm<BankenstaemmeF> {
 private JComponent landSuchFeld;
 //dito für die weiteren Komponenten

 public BankenSubForm() { super("BankenSubForm");}

 @Override
 protected void initComponents() {
  final ComponentBuilder builder = ComponentBuilder.instance(getBuilderFactory(),
    getModel(), getFormats(), getResourceMap());
  final GepardBuilder gepardBuilder = new GepardBuilder(getBuilderFactory(), getFormats(),
    getResourceMap());
  final BankenstaemmeFDesc desc = BankenstaemmeF.DESC;

  landSuchFeld = gepardBuilder.createLandSearchField(getModel(), desc.land());
  clearingNrTextFeld = builder.createTextField(desc.bankIdentifikator());
  //dito for additional components
 }

 @Override
 protected JComponent buildPanel() {
  final TwoColumnsPanelBuilder builder = TwoColumnsPanelBuilder.instance(getBuilderFactory(),
getResourceMap());
  builder.add("land", landSuchFeld, "clearing", clearingNrTextFeld);
  //dito für die weiteren Komponenten
  return builder.getPanel();
 }}
```

Experience I

- Development
  - Developers with average Swing/Java know-how can be as efficient as 4GL developers
  - Better results (both user interface and code)
  - Development is a lot more fun
  - Some business logic was cleaned up against original intentions
- Run-Time performance
  - DB and business logic layer performance unchanged
  - GUI performance even better
  - Increased end-user performance due to much improved UI

Experience II

- Loading the persistence layer can be expensive
  - Modules are loaded lazily
  - 3-tier architecture might be helpful
- Effort for framework/infrastructure: ca. 4 PY

Future Directions

- More Domain Specific Languages
  - Plain vanilla forms can easily be described by DSL
  - Issues: linking to events/presentation logic, refactoring, ...
- Better support for Java-based business logic
  - Much easier to develop complex business logic in Java than in PL/SQL
  - Encapsulate the business logic in a service of its own
- Scripting
  - Migrate application configuration from Java to Groovy
  - Glue code

Summary

- Meta design is crucial for development productivity and UI quality
- Mix and match rather than reinventing the wheel
- Application/domain specific framework/infrastructure vastly increases productivity
- Infrastructure pays off even for medium sized-projects

Vielen Dank!

Bruno Schäffer
Canoo Engineering AG