LEONARDI

Model Driven Development
for Composing Business Applications

Jean-Loup Comeliau (W4)

jean-loup.comeliau@w4.eu

www.lyria.com www.w4global.com

13-14 May 2010, Rome : Workshop on Future Standards for MBUI

W4 : Presentation

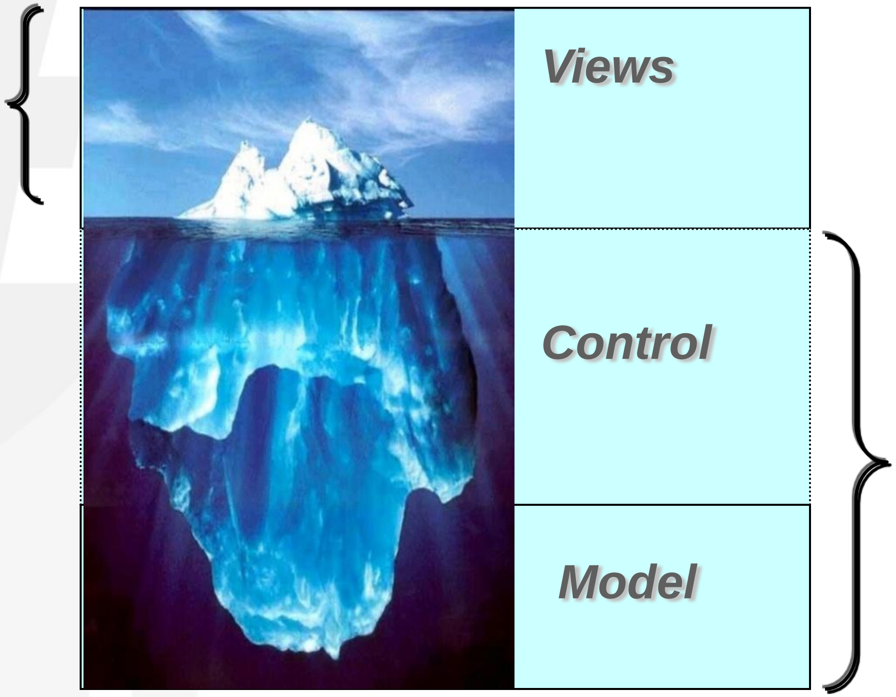
- **W4 : Software Editor created in 1996**
  - SME based in Paris, France
  - 60 persons

- **Specialized in:**
  - Workflow solutions
  - MDD, J2EE, Web technologies
  - GUIs : **LEONARDI (since 2001)**
  - Business Applications

- **Customers**
  - Fortune 100 Companies
  - Software Editors
  - Integrators

LEONARDI Scope

Application



Visible Part

Screens : RIAs, Fat clients, Eclipse…

- Login
- Tables
- Forms
- Dashboards
- Stats
- Maps
- Gantt
- …

Hidden Part

- Connection to the IS
- Navigation tree
- Screen feed
- User profiles
- Integration with underlying technologies
- Editing rules
- Consistency controls
- …

Issues to solve

➢**For end users, the GUI *is* the application!**

High Expectations: Look & Feel, robustness, user-friendliness, user experience…

➢**Implementing a GUI remains complex**

✓Technologies keep changing

✓Strong constraints : take into account legacy

✓Complex user driven combinations need to be anticipated

➢**Delivering a GUI is expensive**

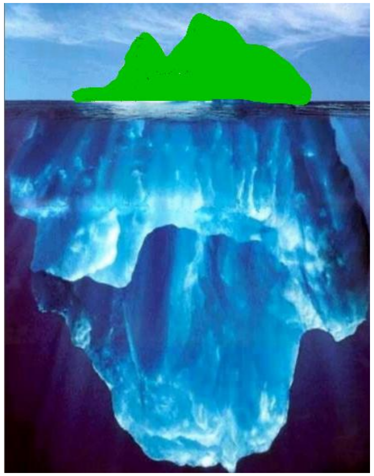
50% of code in development is related to the GUI (Source: IEEC)

⇨ **Consequence :**

In traditional approaches, *80% of the time is dedicated to the infrastructure of the application and 20% only to its business dimension (Source: IDC)*

Alternatives

(1) Programming



- **Code : Views, Controls, Model**
    - ❑ A programming language
    - ❑ A component library
  - **Pros :** Custom-built GUI
  - **Cons :** Work load, maintenance, extensibility…

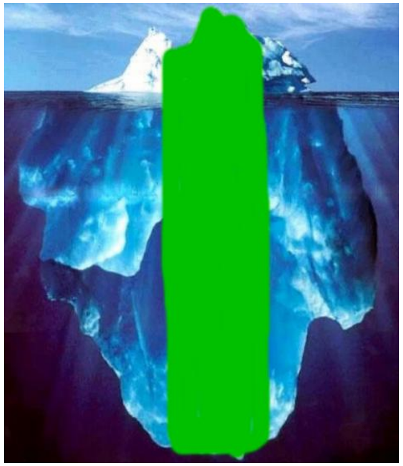
- **Graphical Builders : Drag & Drop Screen Design**
  - **Pros :** Quick programming of the presentation layer
  - **Cons :** Only addresses the presentation layer (views)

Alternatives

(2) 4GLs

MS Access, PowerBuilder, Clarion, 4Js…

Programming languages that elevate the abstraction level



- **Pros:**
  - Accelerates development
  - Requires less technical skills
  - Addresses the problem from end to end
- **Cons:**
  - Technical Limitations : OK for management software
  - Lacks openness : legacy technologies

Alternatives

(3) MDA Solutions

AndroMDA, MTF, Acceleo, MIA Generation, MDWorkbench…

- Model Transformations, PIMs, PSMs,
- Able to generate code, including for GUIs
- Use of IDEs

- **Pros :**
  - Decoupling business / technology
  - Automation of some tasks
  - Improvement of the SDLC
- **Cons :**
  - Only addresses the static part of GUIs
  - Some work at the IDE level
  - Generated code must evolve as the model evolves

W4’s Solution : LEONARDI

A Model-Driven framework for composing Business Applications

- The developer focuses on the data and the business rules
- LEONARDI is in charge of the technical infrastructure

Vision

Main characteristics of LEONARDI:

(1) Driven by the business model

- More abstraction
- Decoupling between technology / business
- less code = less bugs = less maintenance
- Agility

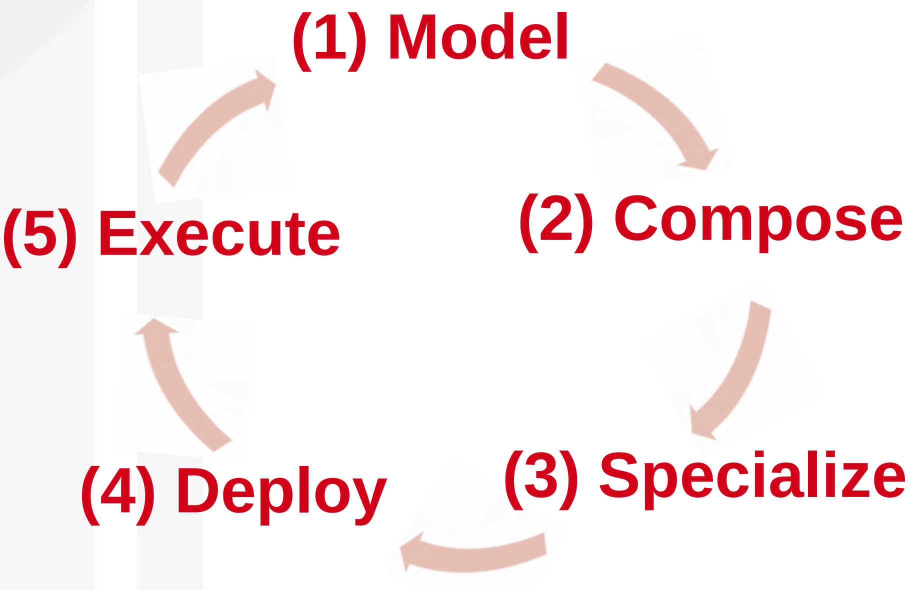
(2) Executed by an application engine

- On-the-fly execution of the model
- No code generation
- Multiple and heterogeneous environments:
  - Presentation layers: Swing, DHTML/Ajax, Eclipse plugins
  - Data sources: DBMS, LDAP, XML, Corba, ECM…

How does it work?

Rely on the logical (business) model to provide an ***operational*** and ***tunable*** end-to-end service.

Agile iteration cycles :



(1) Model

(2) Compose

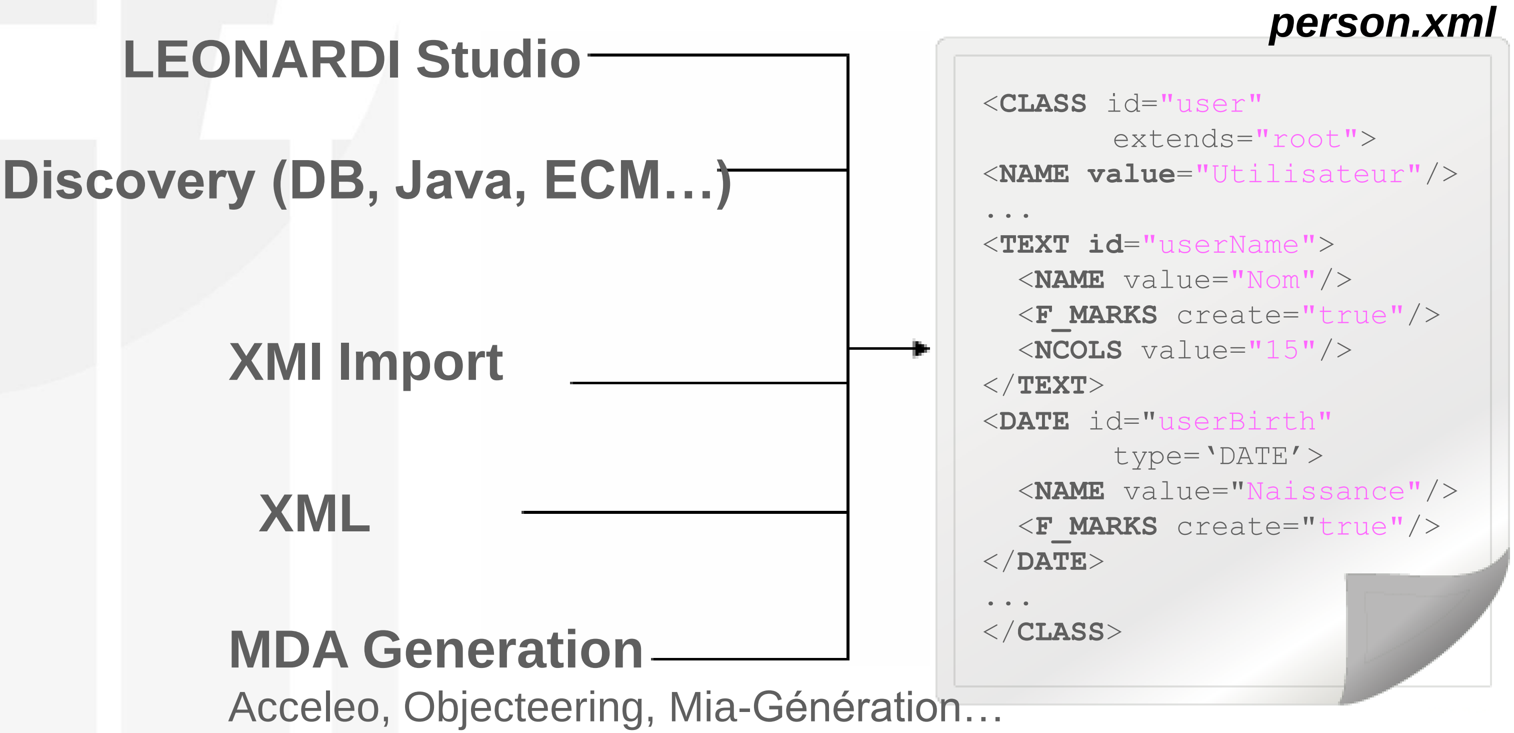
(3) Specialize

(4) Deploy

(5) Execute

How does it work?

Model: Describe the structure of the application data (business model)



person.xml

LEONARDI Studio

Discovery (DB, Java, ECM…)

XMI Import

XML

MDA Generation

Acceleo, Objecteering, Mia-Génération…

```
<CLASS id="user"
        extends="root">
<NAME value="Utilisateur"/>
...
<TEXT id="userName">
  <NAME value="Nom"/>
  <F_MARKS create="true"/>
  <NCOLS value="15"/>
</TEXT>
<DATE id="userBirth"
        type=‘DATE’>
  <NAME value="Naissance"/>
  <F_MARKS create="true"/>
</DATE>
...
</CLASS>
```

How does it work?

Compose: Build the navigation tree by configuring standard views with business model data

Action

- Login
- Tree
- Table
- Gantt
- Map
- Statistics
- Compounding
- Tabs
- Editing
- Prints
- Exports
- User Input …

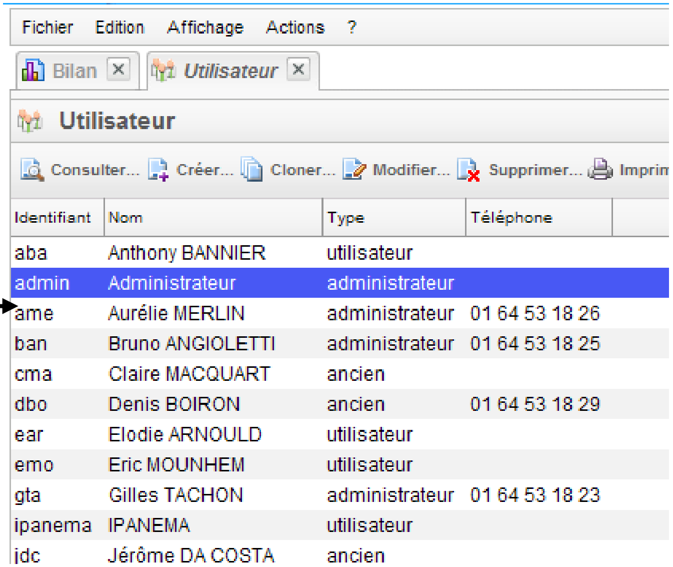
+ Parameters

- Class+
- Field*
- Filter* …

appli.xml

```
<ACTION_REF idref="_table"
class="user">
```

Execution :

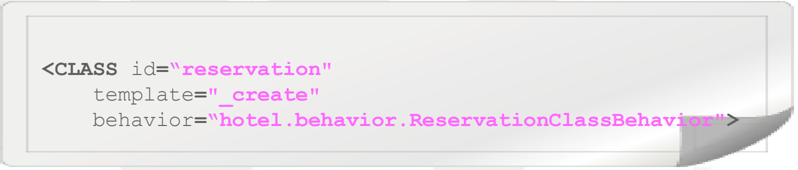


Identifiant	Nom	Type	Téléphone
aba	Anthony BANNIER	utilisateur	
admin	Administrateur	administrateur	
ame	Aurélie MERLIN	administrateur	01 64 53 18 26
ban	Bruno ANGIOLETTI	administrateur	01 64 53 18 25
cma	Claire MACQUART	ancien	
dbo	Denis BOIRON	ancien	01 64 53 18 29
ear	Elodie ARNOULD	utilisateur	
emo	Eric MOUNHEM	utilisateur	
gta	Gilles TACHON	administrateur	01 64 53 18 23
ipanema	IPANEMA	utilisateur	
jdc	Jérôme DA COSTA	ancien	

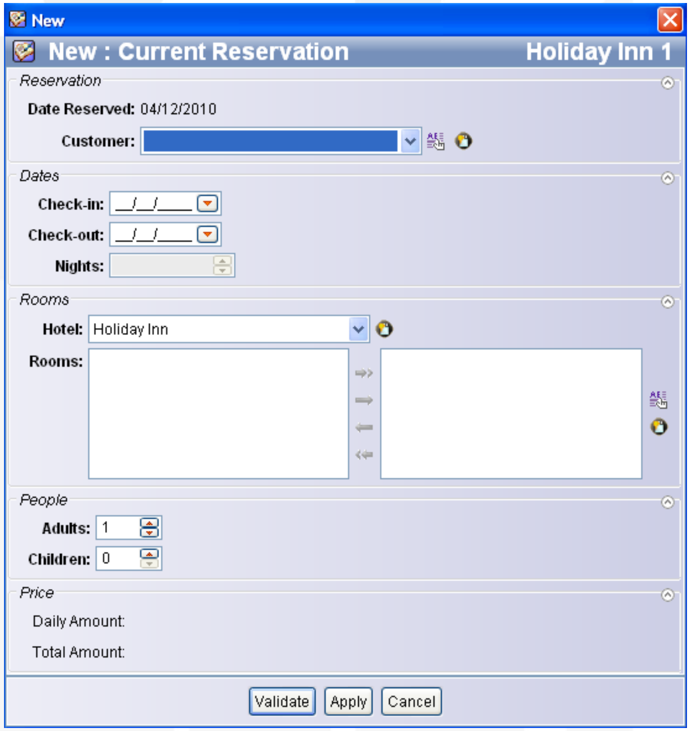
How does it work?

Specialize: Add dynamic behaviors in Java

Reservation.xml



```
<CLASS id="reservation"
    template="_create"
    behavior="hotel.behavior.ReservationClassBehavior">
```



Reservation

Date Reserved: 04/12/2010

Customer:

Dates

Check-in: __/__/____

Check-out: __/__/____

Nights:

Rooms

Hotel: Holiday Inn

Rooms:

People

Adults: 1

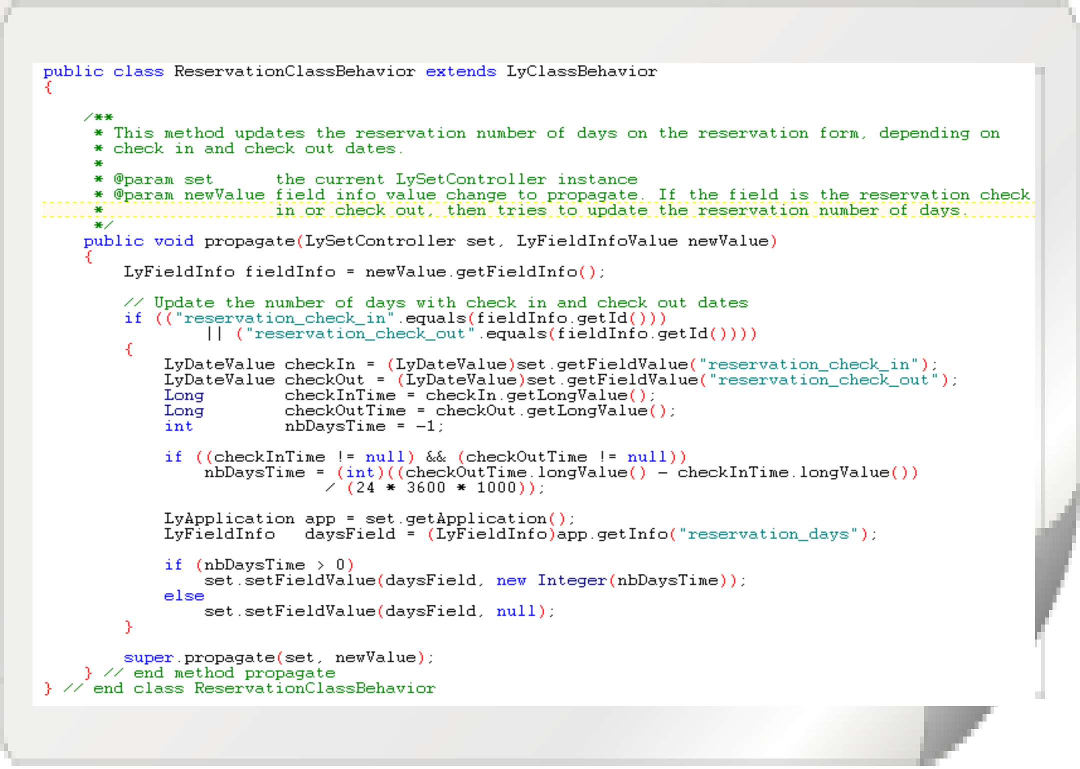
Children: 0

Price

Daily Amount:

Total Amount:

ReservationClassBehavior.xml



```
public class ReservationClassBehavior extends LyClassBehavior
{

    /**
     * This method updates the reservation number of days on the reservation form, depending on
     * check in and check out dates.
     *
     * @param set      the current LySetController instance
     * @param newValue field info value change to propagate. If the field is the reservation check
     *                 in or check out, then tries to update the reservation number of days.
     */
    public void propagate(LySetController set, LyFieldInfoValue newValue)
    {
        LyFieldInfo fieldInfo = newValue.getFieldInfo();

        // Update the number of days with check in and check out dates
        if (("reservation_check_in".equals(fieldInfo.getId()))
                || ("reservation_check_out".equals(fieldInfo.getId())))
        {
            LyDateValue checkIn = (LyDateValue)set.getFieldValue("reservation_check_in");
            LyDateValue checkOut = (LyDateValue)set.getFieldValue("reservation_check_out");
            Long        checkInTime = checkIn.getLongValue();
            Long        checkOutTime = checkOut.getLongValue();
            int         nbDaysTime = -1;

            if ((checkInTime != null) && (checkOutTime != null))
                nbDaysTime = (int)((checkOutTime.longValue() - checkInTime.longValue())
                           / (24 * 3600 * 1000));

            LyApplication app = set.getApplication();
            LyFieldInfo   daysField = (LyFieldInfo)app.getInfo("reservation_days");

            if (nbDaysTime > 0)
                set.setFieldValue(daysField, new Integer(nbDaysTime));
            else
                set.setFieldValue(daysField, null);
        }

        super.propagate(set, newValue);
    } // end method propagate
} // end class ReservationClassBehavior
```

How does it work?

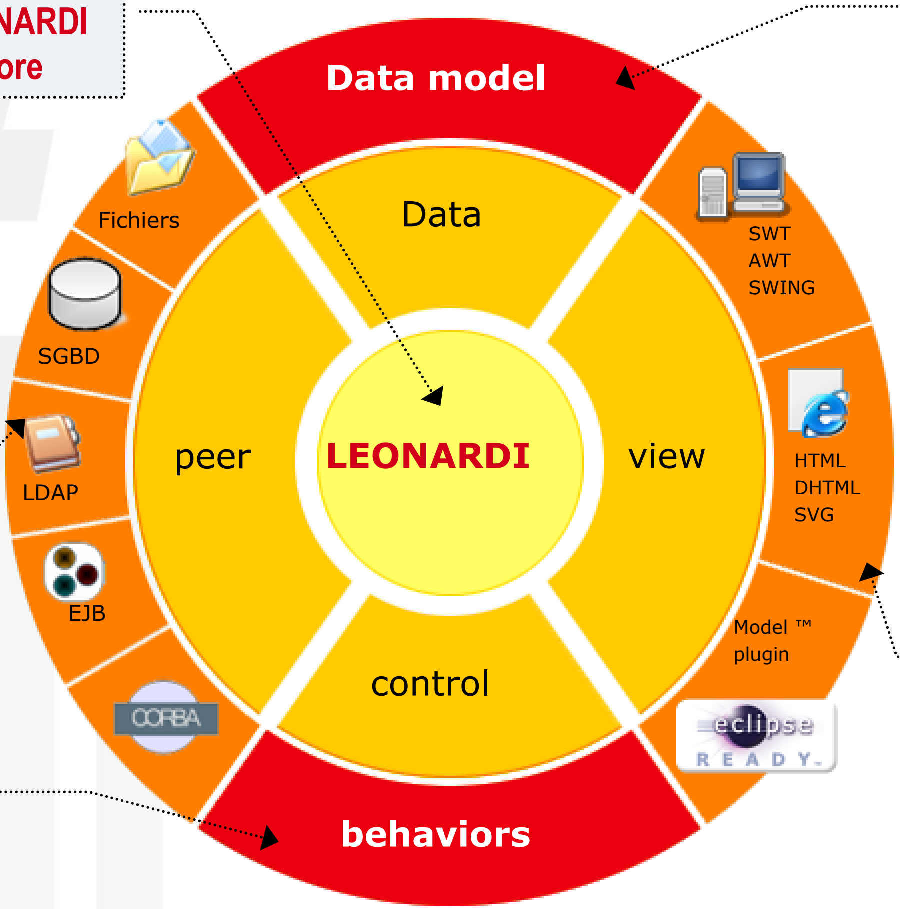
Deploy and Execute

- **Deploy**
  Multi-target presentation layers
  - Swing, SWT, Eclipse, DHTML/Ajax…
  - Specific skins

- **Execute**
  - Multi-data source : RDBMS, ECM, LDAP, Files (csv, xml, …),
  - On-the-fly building of screens
  - Editing controls
  - On-demand data loading
  - Automatic updates of views ...

Multi-OS: 100% Java Windows, UNIX, Linux, Mac…

Architecture



LEONARDI Core

XML : Actions + Data Description

Data Source Connectors

Viewers: Web/Ajax, Eclipse, Swing …

Java : Business Logic, GUI

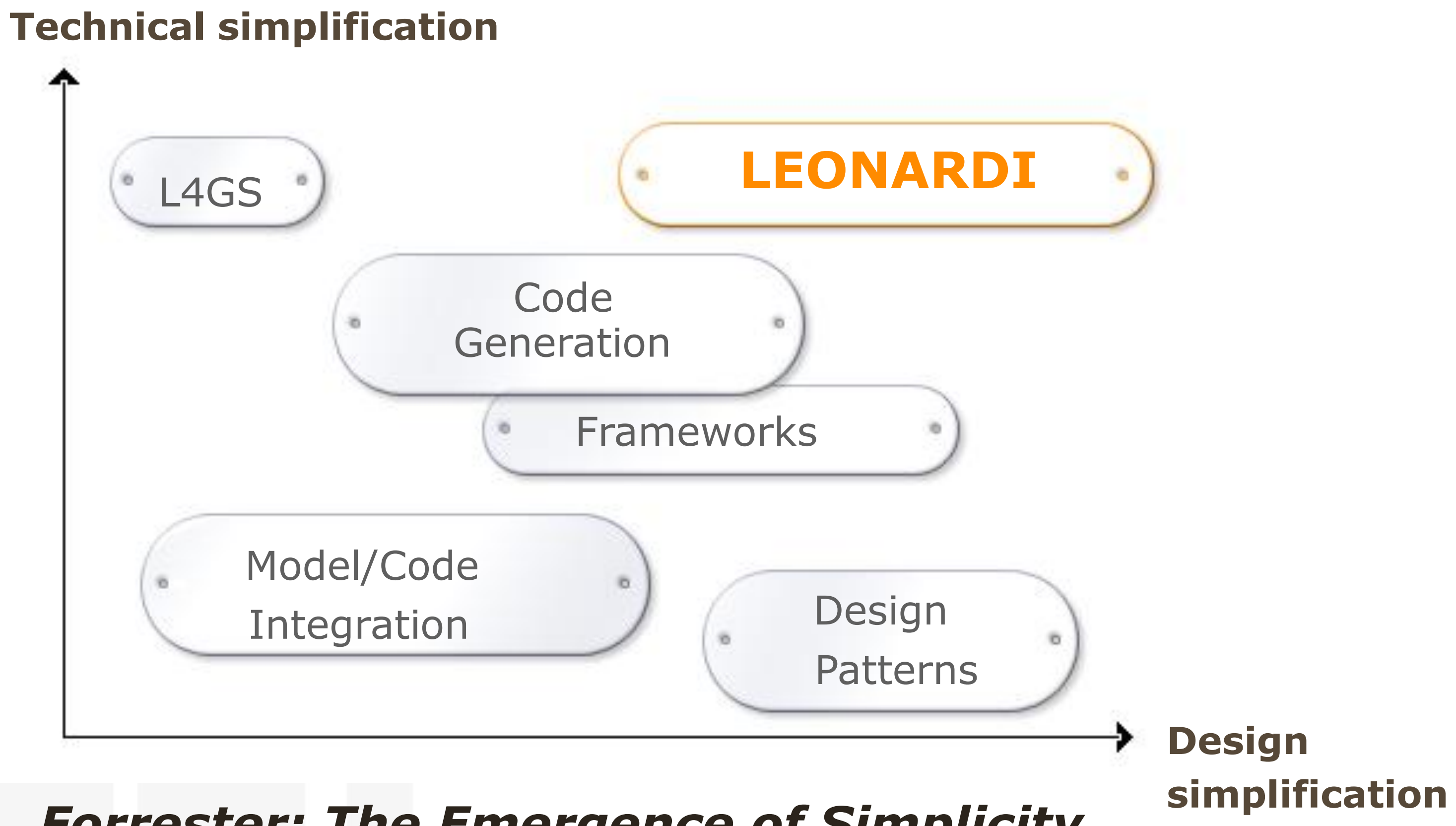
Benefits

Optimized ROI: Lower costs and time to market

- ✓ Simplification: no need to be a technical expert
- ✓ Portability, openness
- ✓ Robustness, quality
- ✓ Reusability
- ✓ Agility
- ✓ Durability: Technically and functionally
  - Business and technology loosely coupled
  - When the model changes, the application changes!
- ✓ Features
  - Database discovery, XMI Import
  - Multi-language
  - Customization
  - …

Benefits

LEONARDI : *GUIs Made Simple*



Forrester: The Emergence of Simplicity

Application types

Technical applications with:

✓Complex information systems: lots of data to address, lots of screens to display

✓Low time to market

- ERPs
- SCM
- CRM
- Supervision
- Configuration
- Commanding systems
- Stock Management

…

Customers and typical needs

- **Integrators:** Project leaders, CTO, project managers
  - Off-the-shelf solutions, in-house development
  - Prototypes
  - Internal Tools
- **Software editors:** CTO, project managers

  Migration, modernization of existing software packages
- **Final users:**
  - RAD projects with integration of legacy products such as SAP, Siebel…
  - Modernization of client-server applications written with L4Gs : Forms, PowerBuilder, Windev, Uniface…
  - Migration of legacy, complex systems
  - Specific turn-key solutions

Use Cases

Typical LEONARDI implementations:

➢Software Factories for Fortune 100 companies, in the IT department, to fulfill internal needs such as stock management, ERP, CRM, HR…

➢VAR Solutions for specialized editors who can then focus on their added value in their own business: Network supervision, ECM, SCM…

➢Project mode for editors and integrators, to accelerate the implementation of strategic projects: monitoring, reporting…

Customers

LEONARDI: Pragmatic MBUI with concrete market results

atpi ware
GROUPE BEAUMANOIR
cache cache
AIR FRANCE
ELIOPS
msa
santé
famille
retraite
services
Getima
EADS
ALSTOM
ALBERT DELOIN SA
CONSEIL EN ORGANISATION
CREATEUR ET EDITEUR D'ADA
AD
AIRBUS
CS
Atos Origin
LASCO
GROUPE JJL
FTx
mixcom
NIJI
SITA
STRUCTIS
TECH'advantage
Groupe IFP
equant
ipanema
Technologies
Gluon
NETWORKS
tdf
orange
LDCOM
NETWORKS
AON
NETWORKS
l'Assurance Maladie
Caisse Nationale
DUST
NETWORKS
Kabira
sab