Ercatons und Organisches Programmieren mit Java & XML: Planwirtschaft adé

Falk Langhammer (Intro)
Guido von Walter (Demo)
Oliver Imbusch (Q&A)

Living Pages Research GmbH (München)

w-jax® 2006

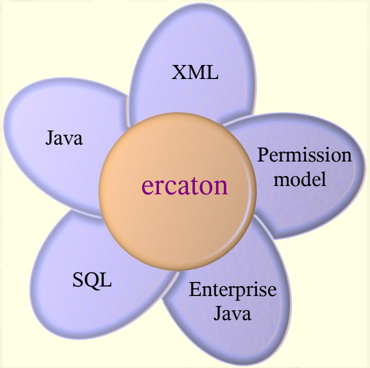
Enterprise ARCHITEKTUR KONFERENZ 2006

Table of Contents

I. **Things** & Organic Software Engineering

II. **Ercatons** (hands-on)

III. New horizon & going Open Source



(this is an ercaton ☺ *)*

Part I

Organic Programming

(*Things* & Organic Software Engineering)

"complexity emerges from simplicity"

The Manifesto of
Organic Programming*

*The **exception** is the rule.*

§1 *Our world is **rich** and complex*
rather than well-structured and simple.

§2 *Software must cover **irregular**, changing patterns*
rather than regular patterns.

§3 A *software system is an **organic** being*
rather than a set of mathematical algorithms.

§4 *Software components are an **integral part** of our rich world*
rather than entities at some meta level.

§5 *Complex software **emerges** when evolving from small to large*
rather than from concrete to abstract.

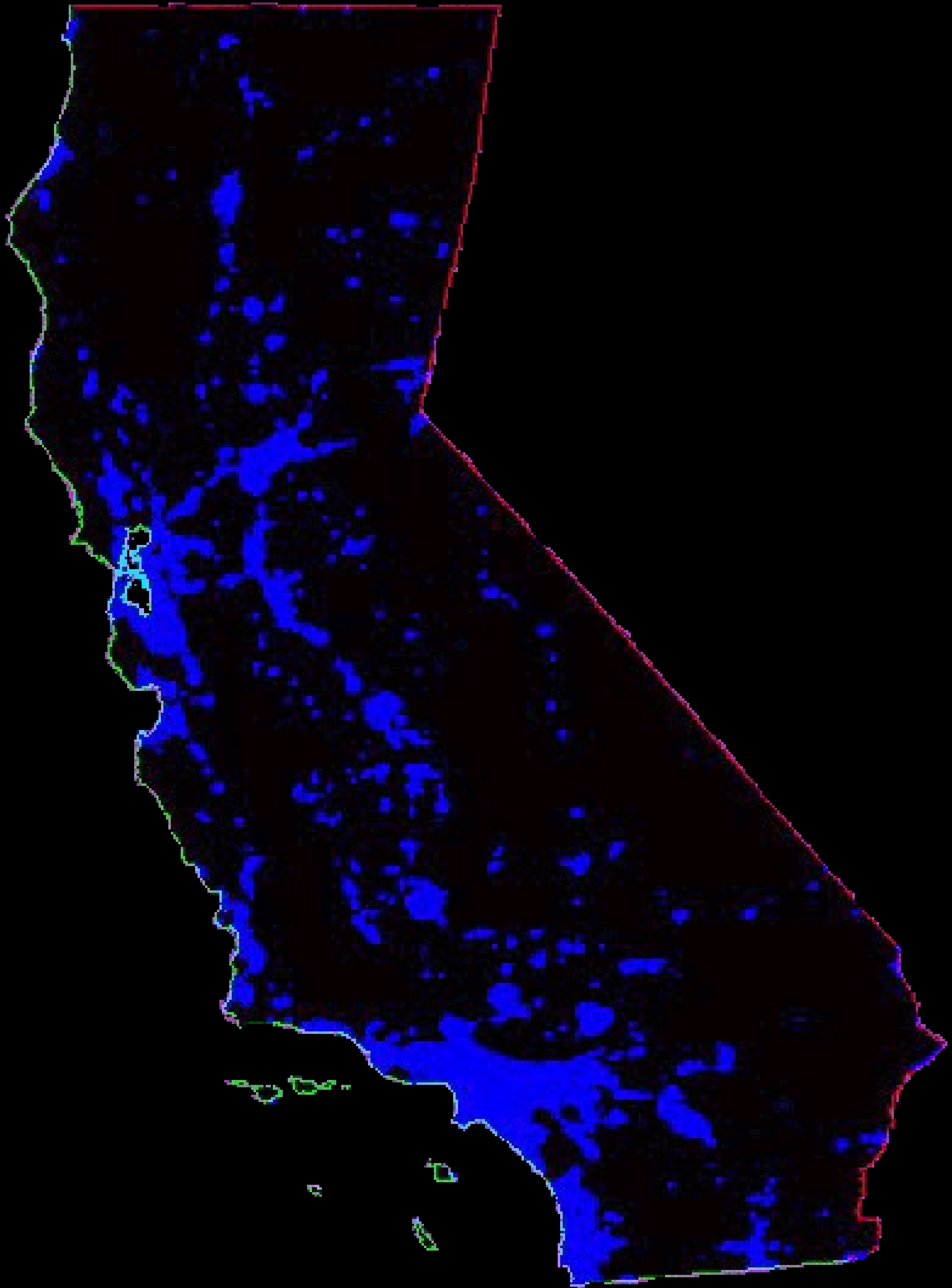
* Living Pages is part of the Organic Computing Initiative www.organic-computing.org

Software *is* **complex** ...

... isn't it ?
You are sure, aren't you?

Maybe, let's first have a look at ...

*...some **really** complex stuff*



by comparison actually, *software* is ***uncomplex***! (only causing terrible trouble to create)

- Today, (OO) software is regular and planned.
- Today, (OO) software cannot grow to the complexity of other systems built by humans.
- Today, (OO) software is harder than hardware...
- Today, we got a problem.

Point of failure of planful software engineering 1/5

For example: *Enterprise Java (Java EE) delivers cost-effective and adequate solutions to approach real-world complexity?*

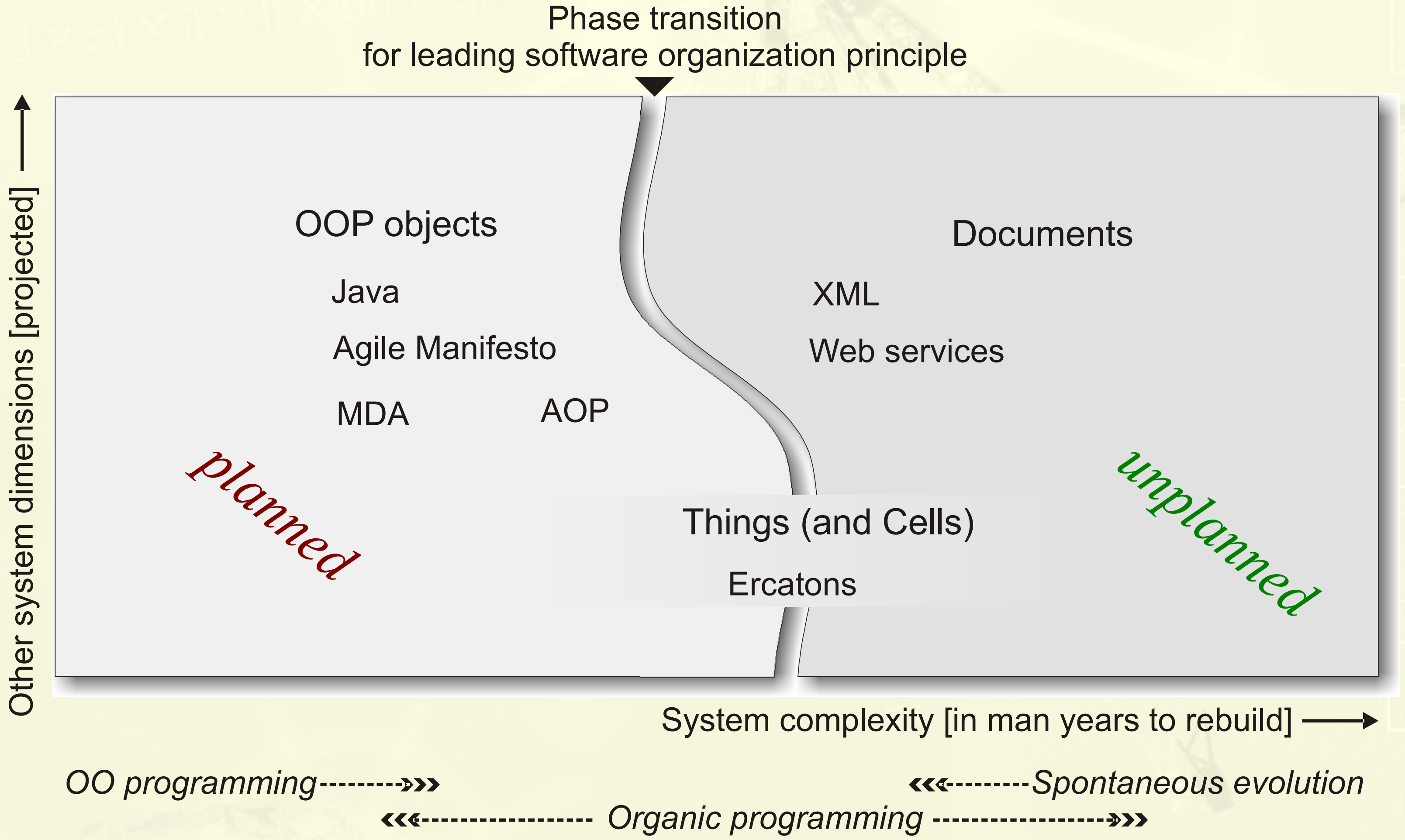
- Actually, it is a mess out there...
- We've seen
  1 billion $ projects
  for tax software ...
  ... fail!

Point of failure of planful software engineering 2/5

Ladies and Gentlemen,
and now
MDA – *Model Driven Architecture*
will do the trick?

(automated generation
of ***working*** programs
from their blueprints?)

Point of failure of planful software engineering 3/5


Phase transition
for leading software organization principle
Other system dimensions [projected]
OOP objects
Java
Agile Manifesto
MDA
AOP
planned
Documents
XML
Web services
unplanned
Things (and Cells)
Ercatons
System complexity [in man years to rebuild]
OO programming---------»»
«««---------Spontaneous evolution
«««------------------ Organic programming ------------------»»

Point of failure of planful software engineering 4/5

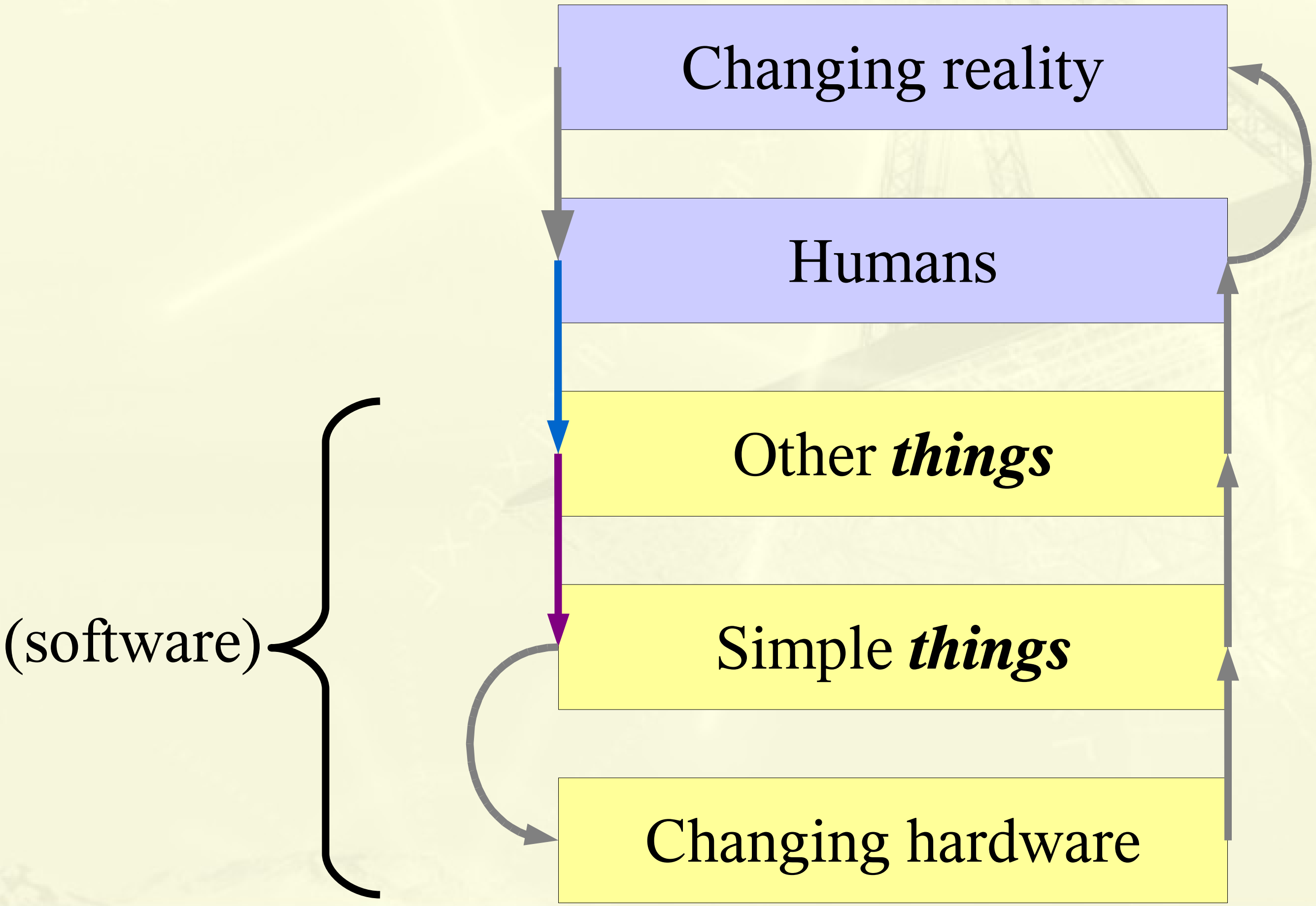
The failure really is:

True ***complexity***
emerges from ***simplicity***
and *cannot be planned*

Point of failure of planful software engineering 5/5

complexity emerges from simplicity
even in software engineering!

How simple (software) *things* can *grow complex*



"Real-world feed-back loop is unbroken."

How simple *things* can be defined (1/3)

- *things* are software entities which must behave like *real-world objects* as much as possible
- If a real-world counterpart is virtual, then *thing* and real-world object must be *indistinguishable* (bank account, tax declaration document, ...)
- *things* depend on a programming language <u>only</u> where algorithms get involved

How simple *things* can be defined (2/3)

Def.: A ***thing*** is a self-contained entity,
with **identity**, **behavior**, with inner **state** and **structure**,
with user† and model‡ **interfaces**, with **ownership** and
with self-determined **lifecycle** and **privacy**,
in both *software* and *reality*.

Def.: A ***cell*** is a thing which may act w/o external stimulus.

†: we can touch, see and manipulate
‡: we can abstract, in order to think about or to code algorithms

How simple *things* can be defined (3/3)

Why software can be harder than hardware:

- Assuming screws being things, (which may be shortened in the field)
- then Java classes aren't things! (cannot be modified in the field)

Is why OO isn't thing-oriented.

Is why OO breaks the real-world loop.

Such a simple thing?

- most simple thing example which may possibly work:

  Let's count votes
  e.g., George W. Bush vs. Al Gore ...

- Time for a quick demo :)

Such a simple thing!

and our 1st **ercaton** – both *document* & *object*

```
<?xml version="1.0"?>
<census>
   <election>US 2000</election>
   <district>
      Mission Bay, San Diego, California
   </district>
☞ <bush>1</bush>
☞ <gore>2</gore>
</census>
```

Such a simple thing – but no simpler

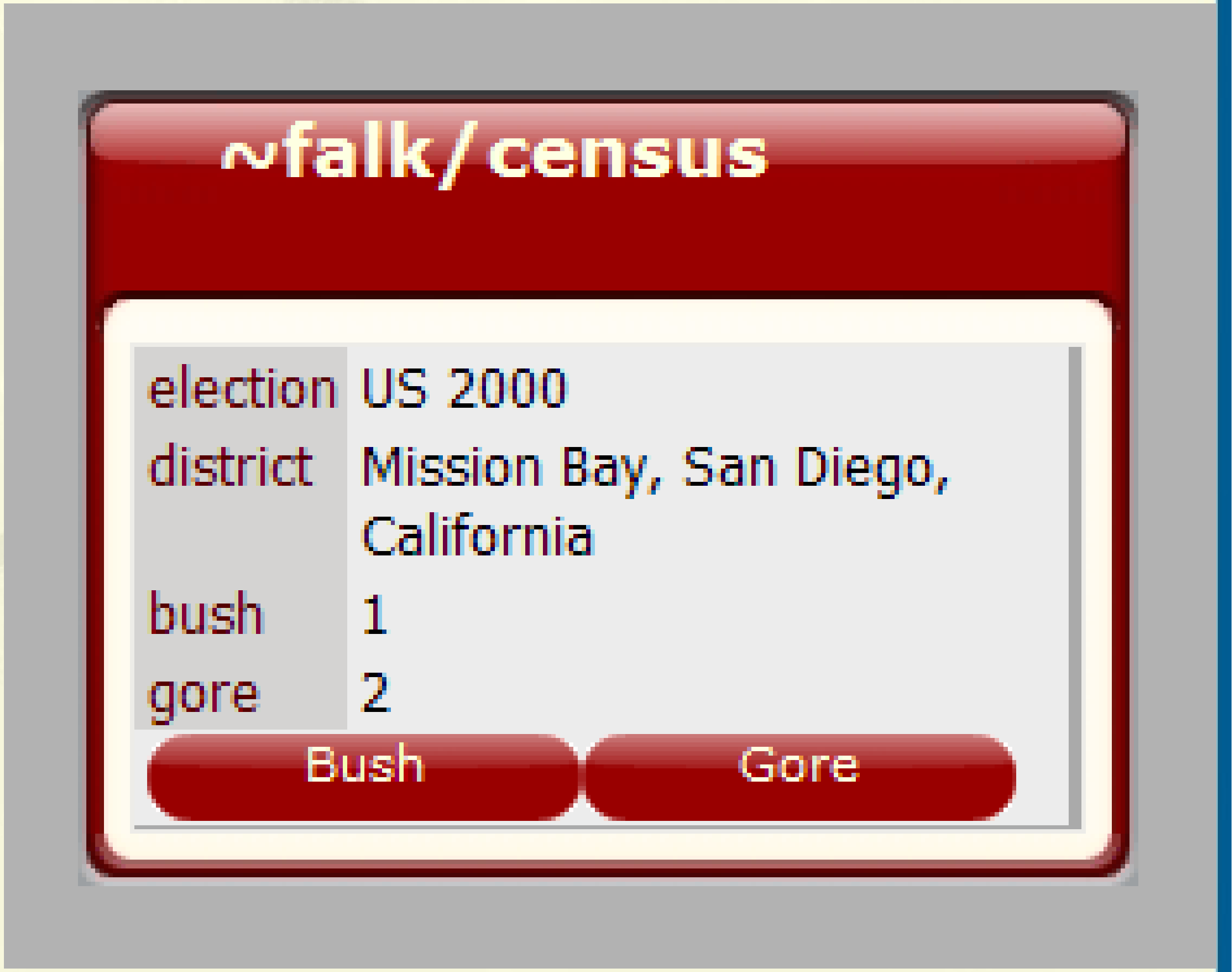
```
<?xml version="1.0" encoding="utf-8"?>
<census xmlns:erc="http://ercato.com/xmlns/ErcatoCore">
☞ <erc:id>~falk/census</erc:id>
   <election>US 2000</election>
   <district>
      Mission Bay, San Diego, California
   </district>
   <bush>1</bush>
   <gore>2</gore>
☞ <erc:action name="Gore">
      /bin/increment <erc:arg name="xpath"> //gore </erc:arg>
   </erc:action>
☞ <erc:action name="Bush">
      /bin/increment <erc:arg name="xpath"> //bush </erc:arg>
   </erc:action>
</census>
```

Such a simple thing – the GUI

- What we got...

... a transaction-safe system with persistence and a user interface to count the votes of Bush and Gore.

... which may be changed arbitrarily while in operation!



~falk/census

election	US 2000
district	Mission Bay, San Diego, California
bush	1
gore	2

Such a simple thing – achievements

- Provides some**thing** which is and feels as **simple** as a screw, in order to assemble arbitrarily complex systems from.

- The ***creation*** of software is like ***building***, not like *modelling*, *generating* or *programming*.

- ***Preserves*** all the inherent potential for ***organic growth*** which is present in our natural way of “building with things”.

- ***Confines programming*** to the purely *algorithmic* parts of a solution, there to be reused.

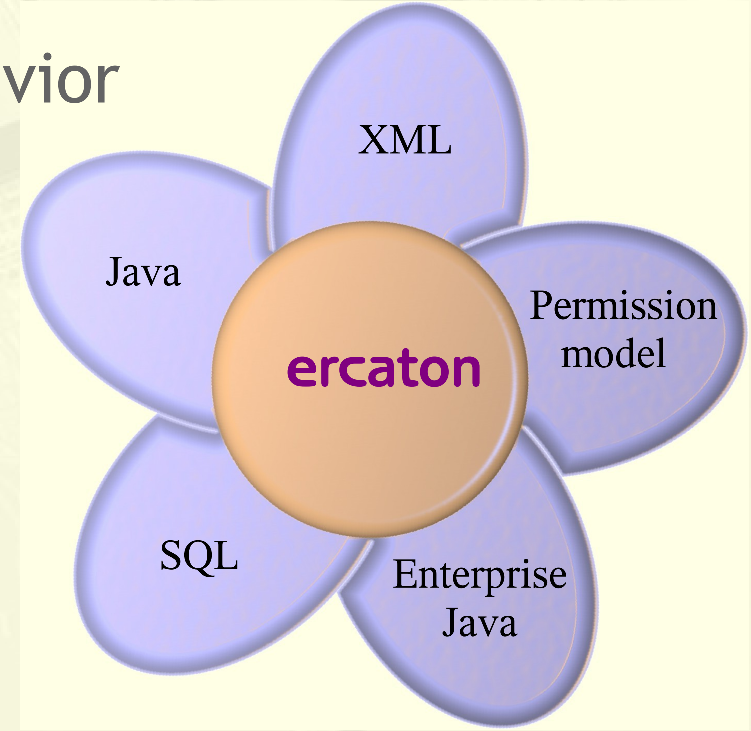
Part II

ercatons

*“everything is a thing,
by definition”*

ercatons "implement" ***things***

- **ercatons** are *things* based on XML and, e.g. Java
- **ercatons** encapsulate state *and* behavior
- **ercatons** are business **objects**
- **ercatons** are **documents**
- **ercatons** merge many older ideas into one unified concept



(ercato specification is a virtual machine contract for execution)

ercato specification summary

Def.: An ***ercaton*** is a **thing**, with at least,
a model interface to **XML**,
with **inheritance** and **polymorphism**,
with a mutable **web** user-interface,
with behaviour bound to **XSLT** and a **Java**-like language,
with **database** and **transaction** support and
with **autonomous life** (cell).

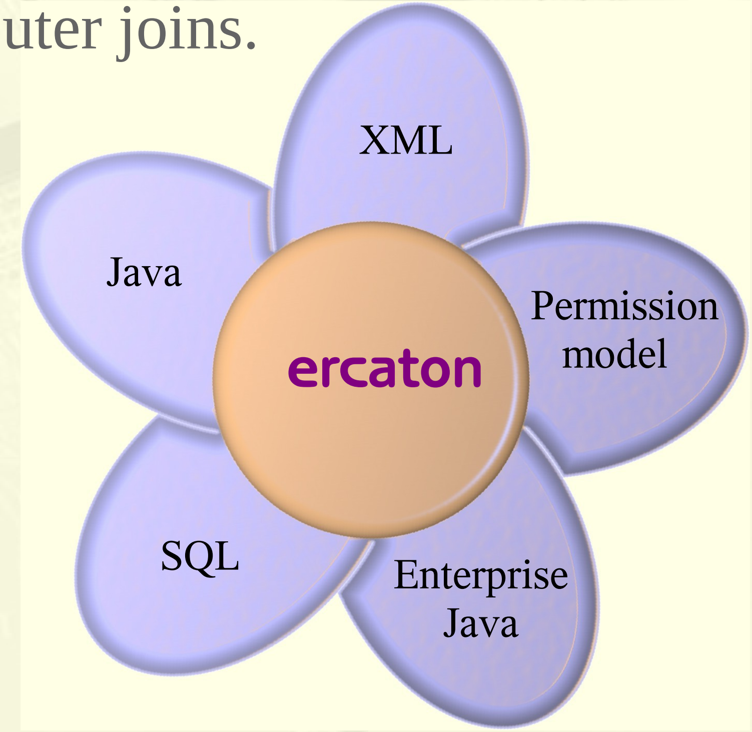
This means that an ercaton stands up for itself, e.g., it does not depend on a class, that it has a **unique name** and is persistent and protected, and that each ercaton is an individual entity where no two are equal

Named after mercato and elementary particle convention (electron)

ercatoJ virtual machine summary

ercatons ...

- ... are protected by **transactions** and **permissions** and ... are supported by **indexing** in a database.
- ... support **SQL**-like queries with inner and outer joins.
- ... are **persistent** and **versioned**.
- ... dynamically **inherit** allowing extraction of common parts of the business logic.
- ... have **user interfaces** by target pipes.
- ... may be binary “resource” ercatons; i.e., **code (Java) may change at runtime**.
- ... have **owners** and a capability chain to **protect** their state.

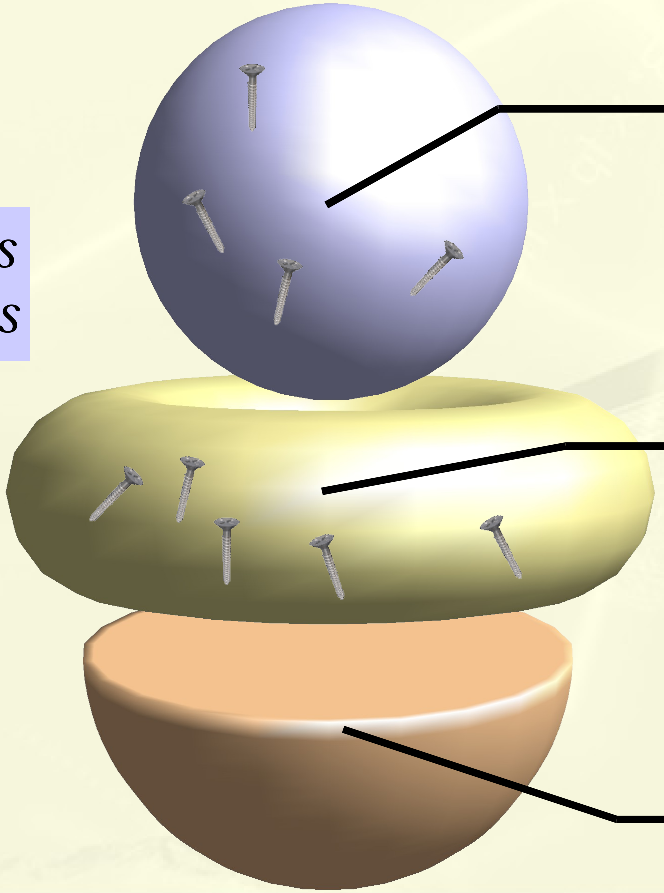


ercatoJ hierarchy

business ercatons

service ercatons

EJBs



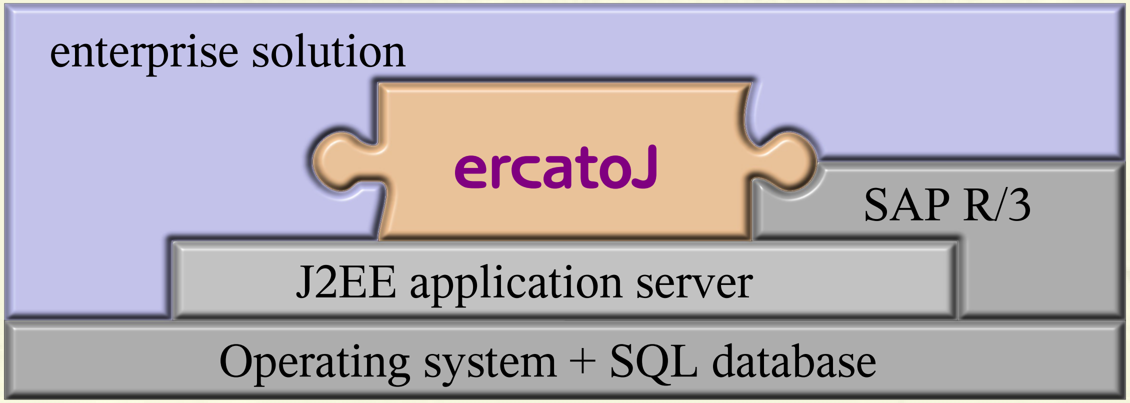
business processes:
customers
recipes, machines, goods, ...
status propagation, workflows
trigger external systems
check business validity
etc.

ercatoX standard extensions: (erx)
user interface management
standard services (cp, check, edit, etc.)
administration tools (backup etc.)
etc

ercatoJ machine: (erc)
ercatons
permissions, transactions
actions, clones, versions
xml interchange
webservices
http (browser) access
etc.

ercatoJ vm made by Living Pages:

(1) Ercatons are mapped onto **Enterprise JavaBeans** (EJBs).
(2) Powerful **algebra for XML** which maps OO principles to math ops.
(3) Behaviour of ercatons expressed in both Java and/or XSLT.
(4) WebServices and plain XML exchange available, e.g. for SAP/R3.
(5) Naked ercatons, user interface web-based, console via esh, or Swing-RC.
(6) Everything may be an ercaton, incl. images and binary code.
(7) Complex database schemes are generated and kept synchronized.



Other simple demo examples

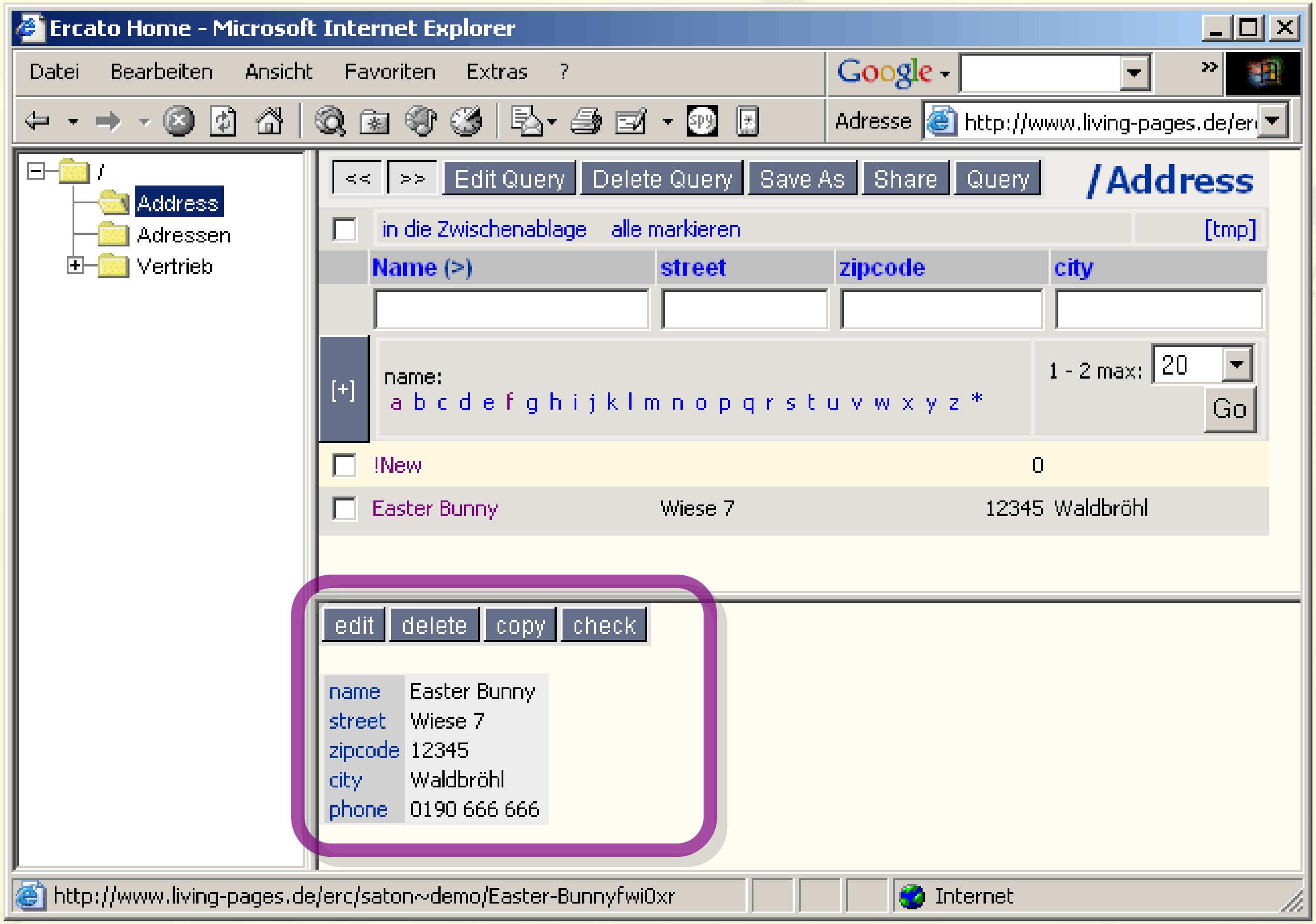
Task:	Design & implementation of an “**Address Manager**” application, then growed to an “**Invoice Manager**”.
Detail:	Enterprise-quality and extensible
Budget:	... 15 person minutes for part 1 ... ~1 day for part 2 (demonstrated *during* DMS 2006!)

here we go...

~falk/bunny is an **ercaton**

```
<?xml version="1.0" encoding="iso-8859-1"?>
<address
xmlns:erc="http://ercato.com/xmlns/ErcatoCore">
     <erc:id>~falk/bunny</erc:id>
     <erc:clone>~livis/adr/base</erc:clone>
     <name     >Easter Bunny</name>
     <street   >Wiese 7</street>
     <zipcode  >12345</zipcode>
   <phone    >0190 666 666</phone>
</address>
```

~falk/bunny and (4) friends



Adresse: http://www.living-pages.de/er

	Name (>)	street	zipcode	city
□	!New		0	
□	Easter Bunny	Wiese 7	12345	Waldbröhl

name	Easter Bunny
street	Wiese 7
zipcode	12345
city	Waldbröhl
phone	0190 666 666

http://www.living-pages.de/erc/saton~demo/Easter-Bunnyfwi0xr

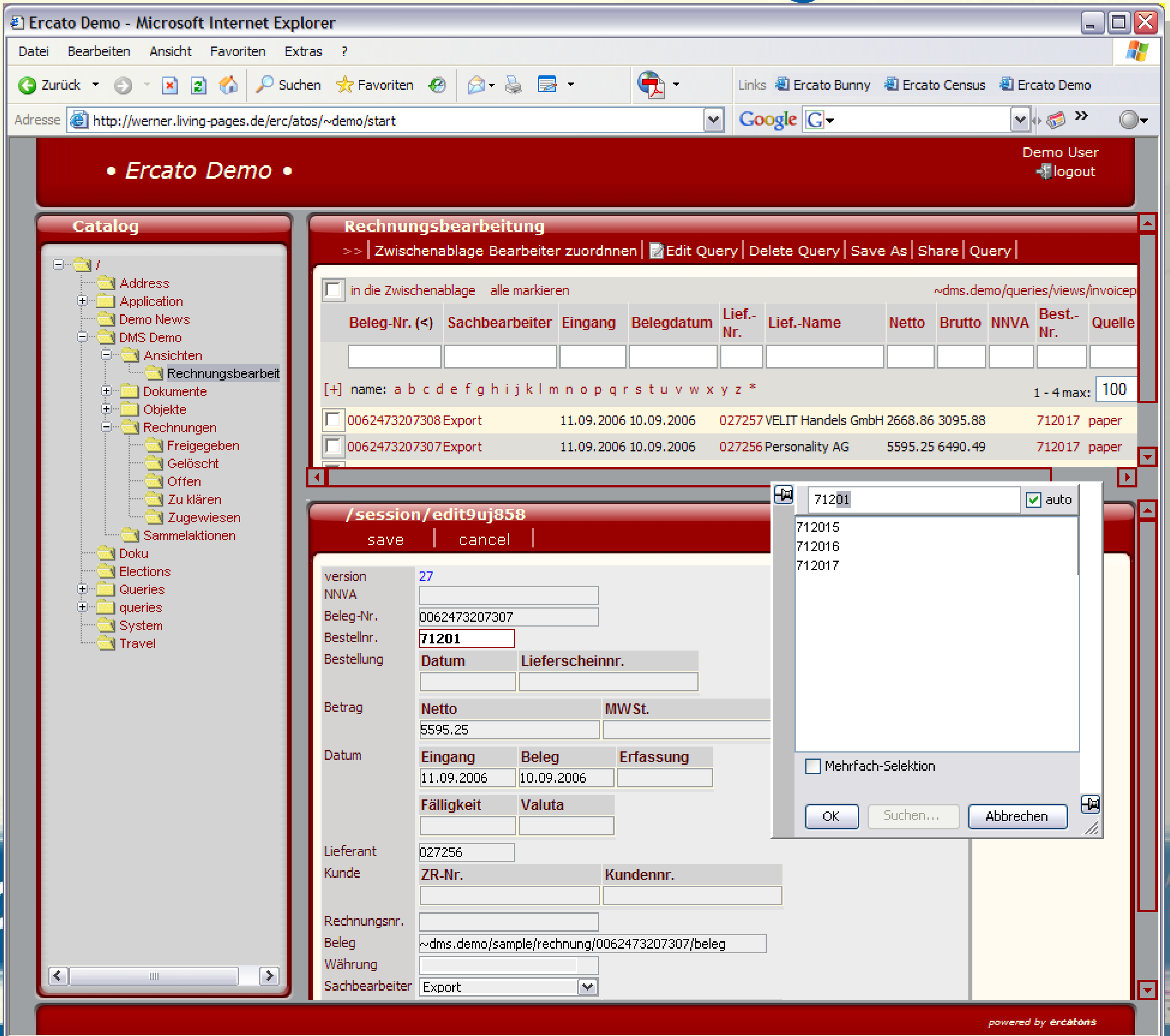
~livis/adr/base : class or template?

```
<?xml version="1.0" encoding="utf-8"?>
<address xmlns:erc="http://ercato.com/xmlns/ErcatoCore"
         xmlns:erx="http://ercato.com/xmlns/ErcatoExtensions">
        <erc:id>~livis/adr/base</erc:id>
        <erc:type>prototype</erc:type>
        <erc:catalog category="/Address" id-ref="~livis/catalog"/>

        <name    erx:field-ref="string" erc:index="~livis/catalog"/>
        <street  erx:field-ref="string" erc:index="~livis/catalog"/>
        <zipcode erx:field-ref="int"    erc:index="~livis/catalog"/>
        <city    erx:field-ref="string" erc:index="~livis/catalog"/>
        <phone   erx:field-ref="string"/>

        <erc:action name="edit">   /bin/edit       </erc:action>
        <erc:action name="delete"> /bin/rm!wizard  </erc:action>
        <erc:action name="copy">   /bin/cp!forEdit </erc:action>
        <erc:action name="check">  ~livis/check.xsl
                <erc:arg name="default">San Diego</erc:arg>
        </erc:action>
        <erc:trigger name="on-change">!check</erc:trigger>
</address>
```

Invoice manager



http://werner.living-pages.de/erc/atos/~demo/start

Catalog

- /
  - Address
  - Application
  - Demo News
  - DMS Demo
    - Ansichten
      - Rechnungsbearbeit
    - Dokumente
    - Objekte
    - Rechnungen
      - Freigegeben
      - Gelöscht
      - Offen
      - Zu klären
      - Zugewiesen
    - Sammelaktionen
  - Doku
  - Elections
  - Queries
  - queries
  - System
  - Travel

Rechnungsbearbeitung

>> | Zwischenablage Bearbeiter zuordnnen | Edit Query | Delete Query | Save As | Share | Query |

□ in die Zwischenablage alle markieren ~dms.demo/queries/views/invoicep

[+] name: a b c d e f g h i j k l m n o p q r s t u v w x y z * 1 - 4 max: 100

	Beleg-Nr. (<)	Sachbearbeiter	Eingang	Belegdatum	Lief.-Nr.	Lief.-Name	Netto	Brutto	NNVA	Best.-Nr.	Quelle
□	0062473207308	Export	11.09.2006	10.09.2006	027257	VELIT Handels GmbH	2668.86	3095.88		712017	paper
□	0062473207307	Export	11.09.2006	10.09.2006	027256	Personality AG	5595.25	6490.49		712017	paper

/session/edit9uj858

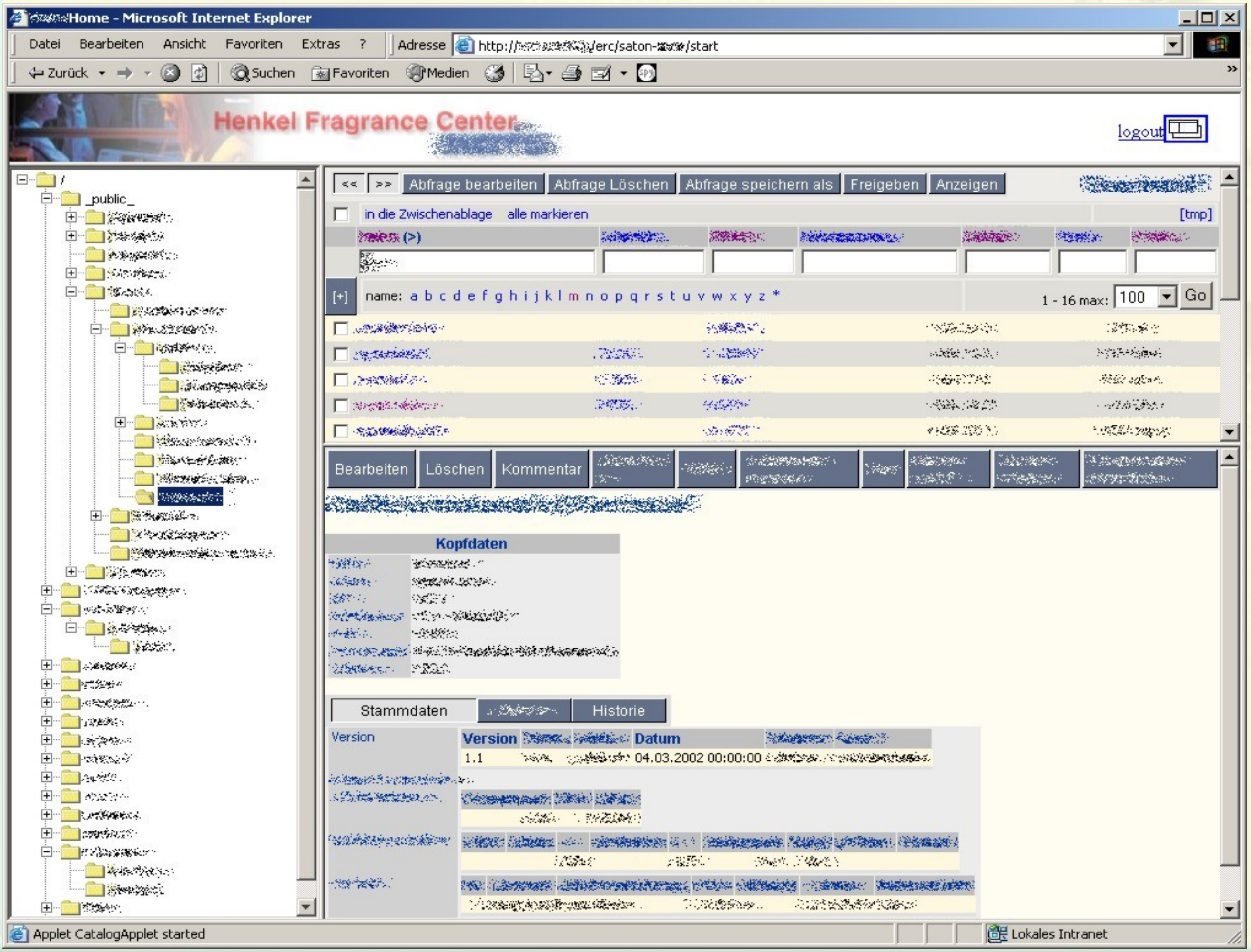
version	27		
NNVA			
Beleg-Nr.	0062473207307		
Bestellnr.	71201		
Bestellung	Datum	Lieferscheinnr.	
Betrag	Netto	MWSt.	
	5595.25		
Datum	Eingang	Beleg	Erfassung
	11.09.2006	10.09.2006	
	Fälligkeit	Valuta	
Lieferant	027256		
Kunde	ZR-Nr.	Kundennr.	
Rechnungsnr.			
Beleg	~dms.demo/sample/rechnung/0062473207307/beleg		
Währung			
Sachbearbeiter	Export		

71201 ☑ auto

712015
712016
712017

□ Mehrfach-Selektion

Henkel A deployed corporate solution...


Home - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://.../erc/saton-.../start
Zurück Suchen Favoriten Medien
Henkel Fragrance Center
logout
/
public
<< >> Abfrage bearbeiten Abfrage Löschen Abfrage speichern als Freigeben Anzeigen
in die Zwischenablage alle markieren [tmp]
[+] name: a b c d e f g h i j k l m n o p q r s t u v w x y z *
1 - 16 max: 100 Go
Bearbeiten Löschen Kommentar
Kopfdaten
Stammdaten Historie
Version
Applet CatalogApplet started
Lokales Intranet

Version	Datum
1.1	04.03.2002 00:00:00

(screenshot: courtesy of Henkel Fragrance Center GmbH)

- **Browser based intranet solution**
- **J2EE server application**

Features:

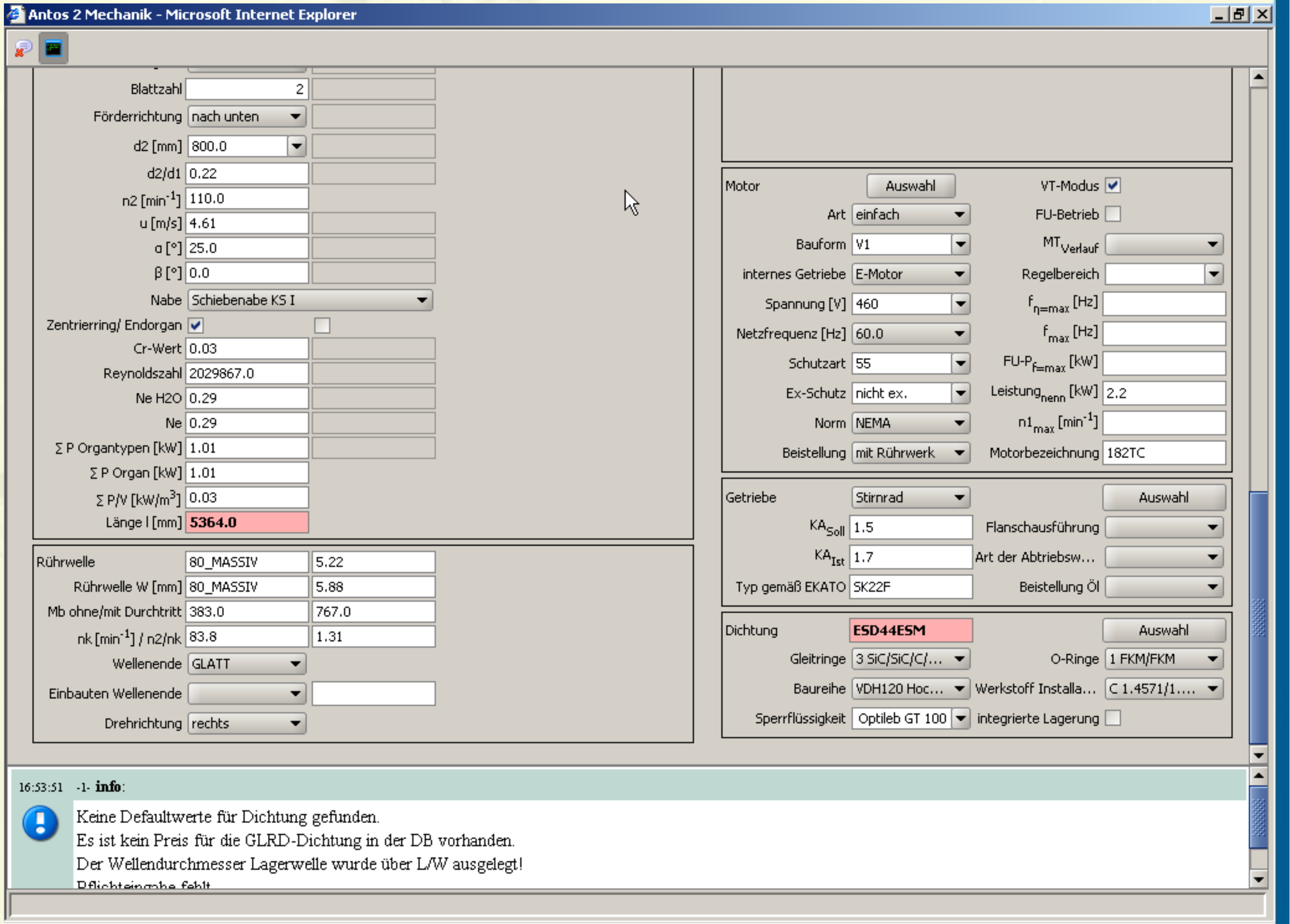
- Document-centric, versioned
- Efficient configurable data mining
- Generated user interface
- “Better than html” controls
- Comprehensible implementation of workflow

EKATO Another deployed solution…

- **Rich intranet client**
- **J2EE server application**

Features:

- Solution archive
- Complex business rule set
- Busines rule factory in Java
- Generated “Rich Client”
- Powerful output mngmnt.
multi-language/
multi-format (PDF, CAD,...)
- Developed by customer



Antos 2 Mechanik - Microsoft Internet Explorer

Blattzahl	2	
Förderrichtung	nach unten	
d2 [mm]	800.0	
d2/d1	0.22	
n2 [min^{-1}]	110.0	
u [m/s]	4.61	
α [°]	25.0	
β [°]	0.0	
Nabe	Schiebenabe KS I	
Zentrierring/ Endorgan	☑	□
Cr-Wert	0.03	
Reynoldszahl	2029867.0	
Ne H2O	0.29	
Ne	0.29	
Σ P Organtypen [kW]	1.01	
Σ P Organ [kW]	1.01	
Σ P/V [kW/m^3]	0.03	
Länge l [mm]	5364.0	

Rührwelle	80_MASSIV	5.22
Rührwelle W [mm]	80_MASSIV	5.88
Mb ohne/mit Durchtritt	383.0	767.0
nk [min^{-1}] / n2/nk	83.8	1.31
Wellenende	GLATT	
Einbauten Wellenende		
Drehrichtung	rechts	

Motor	Auswahl	VT-Modus	☑
Art	einfach	FU-Betrieb	□
Bauform	V1	$MT_{Verlauf}$	
internes Getriebe	E-Motor	Regelbereich	
Spannung [V]	460	$f_{\eta=max}$ [Hz]	
Netzfrequenz [Hz]	60.0	f_{max} [Hz]	
Schutzart	55	FU-$P_{f=max}$ [kW]	
Ex-Schutz	nicht ex.	$Leistung_{nenn}$ [kW]	2.2
Norm	NEMA	$n1_{max}$ [min^{-1}]	
Beistellung	mit Rührwerk	Motorbezeichnung	182TC

Getriebe	Stirnrad		Auswahl
KA_{Soll}	1.5	Flanschausführung	
KA_{Ist}	1.7	Art der Abtriebsw...	
Typ gemäß EKATO	SK22F	Beistellung Öl	

Dichtung	ESD44ESM		Auswahl
Gleitringe	3 SiC/SiC/C/...	O-Ringe	1 FKM/FKM
Baureihe	VDH120 Hoc...	Werkstoff Installa...	C 1.4571/1....
Sperrflüssigkeit	Optileb GT 100	integrierte Lagerung	□

16:53:51 -1- **info**:

Keine Defaultwerte für Dichtung gefunden.
Es ist kein Preis für die GLRD-Dichtung in der DB vorhanden.
Der Wellendurchmesser Lagerwelle wurde über L/W ausgelegt!
Pflichteingabe fehlt

Part III

New Horizon & going Open Source

“ideas are unanticipated”

Patterns we did not expect

- Builder ercatons.
- Aspect weavers.
- Self protection and healing.
- Repackaging pattern.
- Separation of “text” and “data” re-emerges as a pattern:
  - The “firewall” pattern
- Agent ercatons (after adding a “goal”).
- Autonomous evolution:
  - Cross combination is just another XML operator.
  - Mutations don't hurt algorithms.
  - Selection by observer (“Other” ercatons or user).
- ...

going Open Source

Open source project **ercato.org** considered

- License model:
  - GNU source code model
  - No constraint on consultancy business models
  - Constraint on competing commercial re-implementations
- Pre condition:
  - 95% feature list completion
  - 80% documentation completion
  - Evidence of external interest

Problem: **ercatons** are massive innovation where open source projects typically copy existing stuff.

Conclusion

Turns software engineers into creators rather than programmers.

Large, complex systems are feasible and stay to grow organically.

It is saving lot of time and money.

Organic software engineering exists and works!

“*Maybe someday all large-scale objects will be Ercatonical.*”

David Ungar
Principal Investigator
Sun Labs, Mountain View

P.S. *The specification is open,*
the ercatoJ implementation is free
for research partners,
an open-source project is considered.

“*Ercatons were easy to use and breathtakingly efficient. Once you get the idea you wonder how you ever worked without it.*”

Dr. Ralf Marsula
Senior Consultant
Clavis berater sozietät GmbH, Bremen

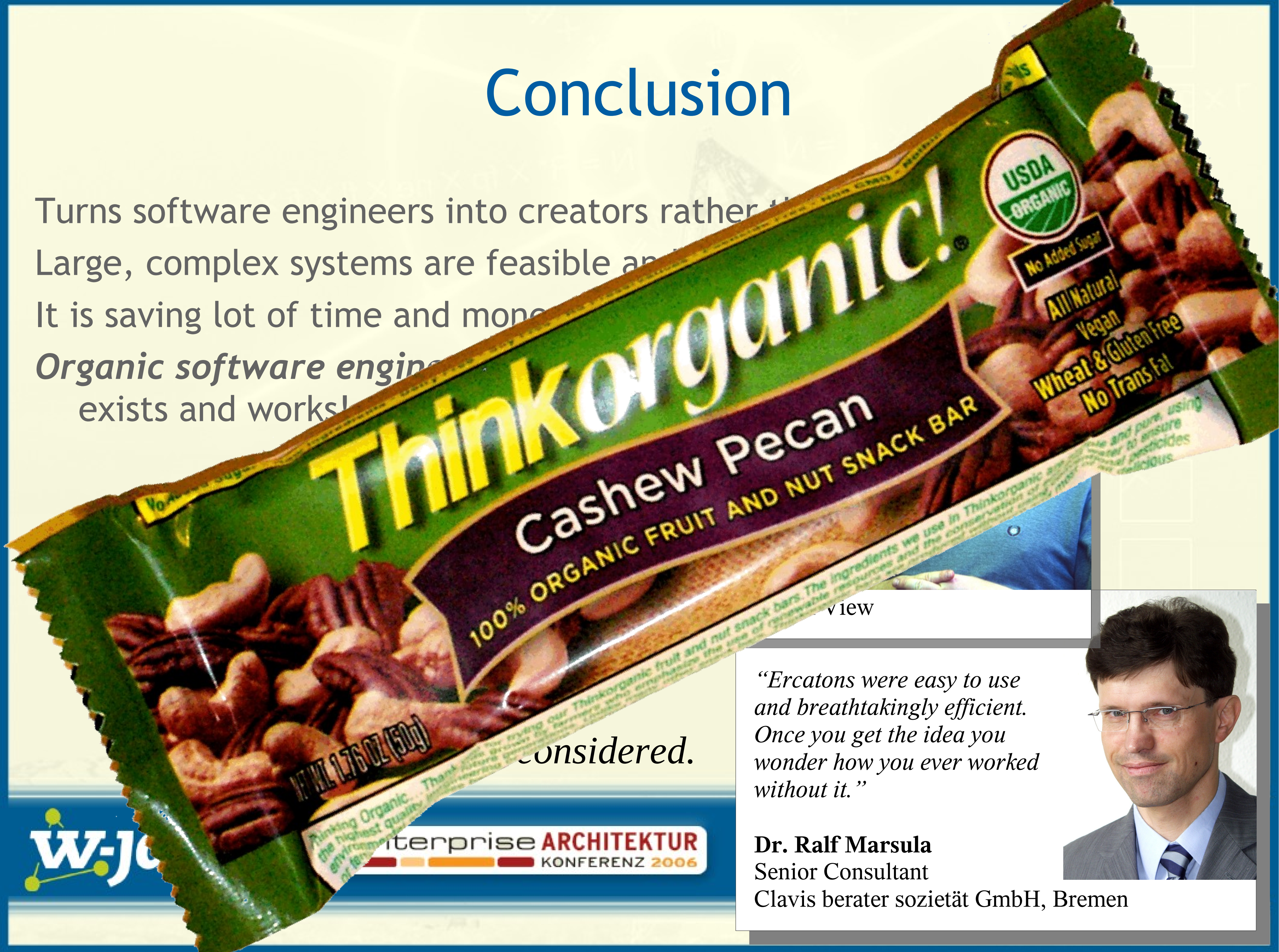
Conclusion

Turns software engineers into creators rather t

Large, complex systems are feasible an

It is saving lot of time and mone

Organic software engin exists and works!



onsidered.

view

"Ercatons were easy to use and breathtakingly efficient. Once you get the idea you wonder how you ever worked without it."

Dr. Ralf Marsula
Senior Consultant
Clavis berater sozietät GmbH, Bremen