Process-based Application Development

A flexible and End-user centered Way of creating Software

Kaspar von Gunten
IvyTeam (Soreco Group)
740

Agenda

> **This talk proposes a new form of software development**

> **Software today**
  – Code based programming by software engineers
  – Partially service oriented (distributed applications)
  – Difficult to customize

> **Software tomorrow**
  – Process based SOA applications
  – Customization and configuration by end-user programmers
  – Intentional programming by domain experts

> **A glimpse at the future**
  – Software Demo (separate slot, not part of this talk)

> **Q&A**

Software Development today : Observations

> All software applications exist to execute processes

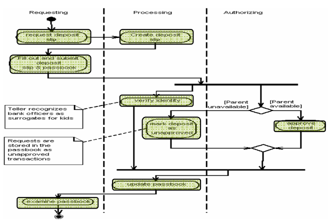
> Distributed applications are realized with SOA

> Standard Software is expensive to customize

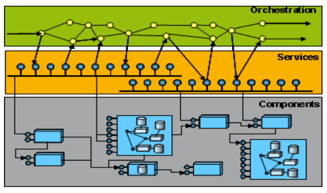
> Source code of software does not clearly show the original intention

Software executes Processes, SOA applications

> **Software is inherently process oriented**
  – All programs exist to perform at least one process.
  – Documented (UML) processes are manually transformed into code.
  – Implementation often diverts from the documentation when process changes.


Requesting
Processing
Authorizing
request deposit slip
Create deposit slip
Fill out and submit deposit slip & passbook
verify identity
[Parent unavailable]
[Parent available]
Teller recognizes bank officers as surrogates for kids
mark deposit as unapproved
approve deposit
Requests are stored in the passbook as unapproved transactions
update passbook
examine passbook

> **SOA applications**
  – A SOA consists of a service layer with orchestration layer on top. Services are rewired to form process or new service.
  – Most SOA use BPEL for orchestration.

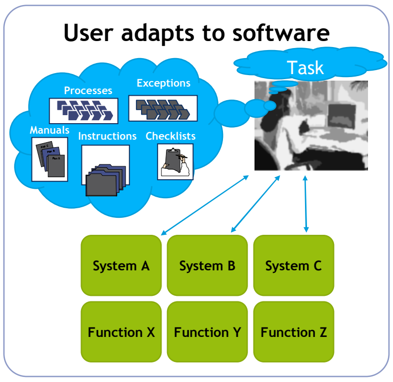


Standard Software is expensive to customize

> Customizing of standard software to the needs of an individual customer is tedious and expensive and therefore often not possible.

> Highly customizable software is often too complex, because it tries to be an “all-in-one device suitable for every purpose”.

> Software forces user to adapt to the application’s needs where the program should really serve the user’s demands.

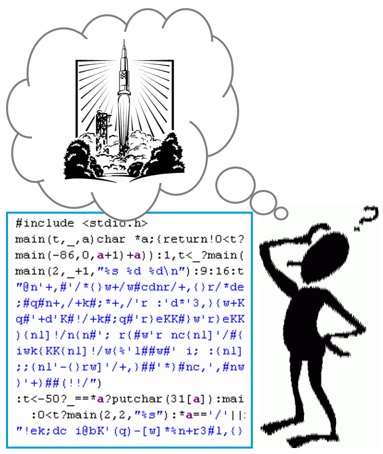


Software Implementation does not show Intention

> Software is developed by creating a (domain) concept model and then transforming the model into source code.

> The source code of an application lacks the clarity of the model and it is often hard to understand the original intention.

> Programmers must constantly “wrap” and “unwrap” domain concepts into code constructs and vice versa. This is an error-prone process.



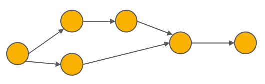
```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?
main(-86,0,a+1)+a)):1,t<_?main(
main(2,_+1,"%s %d %d\n"):9:16:t
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK
){nl]!/n{n#'; r{#w'r nc{nl]'/#{
iwk{KK{nl]!/w{%'l##w#' i; :{nl]
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):mai
  :0<t?main(2,2,"%s"):*a=='/'||
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}
```

Software Development tomorrow : How can we improve?

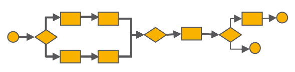
- Use SOA as a general architecture model
- Implement software with an explicit graphical process model
- Apply principles of intentional programming when creating software
- Execute an application’s process model directly
- Let end-users customize the process model of an application

SOA, Explicit Graphical Process Model

- **Use SOA as general architecture model**
  - There is no reason to limit the flexibility of SOA to distributed applications only.
  - Instead, SOA should become a general paradigm for software development.



- **Explicit graphical process model**
  - Since all programs are inherently process based, they should also be designed as a process model from the beginning.
  - Process model orchestrates the services of the underlying SOA.
  - A picture says more than a thousand words.



Excursus

Intentional Programming

> Collection of concepts which enable software source code to reflect the precise information, called intention, which programmers had in mind when conceiving their work.

> Originally introduced by Charles Simonyi at Microsoft Research in the late 1990-ies.

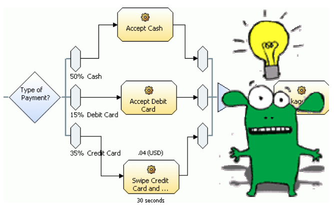
> Simonyi writes:

  "Most software is expressed in a general purpose programming language […] the programs record what is required for the computer […] rather than the problem details. This would not be an issue if only computers looked at programs, but that is evidently not the case."

> Domain Experts ➝ Domain Code ➝ Generated Source Code ➝ Run

Intentional Programming, Process Execution

> **Use Intentional Programming Principles**
- The implementation of an application should clearly show its intention.
- Domain experts should be able to participate actively in an applications development.



> **Process model execution**
- Manual transformation of graphical process model into code should not be necessary.
- Rather, the graphical process should be executed directly.
- If process model == execution model, then a implementation is self documenting.

End-User Programming

Excursus

> **End-user programmers (EUP)**
- Users that write programs for their own use, “almost anybody except a trained programmer”, e.g. teachers, engineers, accountants, etc.
- Often use tools like spread-sheets or databases to “get a job done”.
- Outnumber professional programmers by far, and numbers are steadily growing: Estimate for 2005 was 55 million vs. 3 million (in U.S.); recently revised to potential 90 million in 2012.

> **End-user programming**
- EUP neither have the time nor the interest to learn tools and principles of professional software engineering.
- Variety of techniques: sequence of GUI actions, spread-sheet formulae, application specific language, visual programming, etc.

End-User Customization

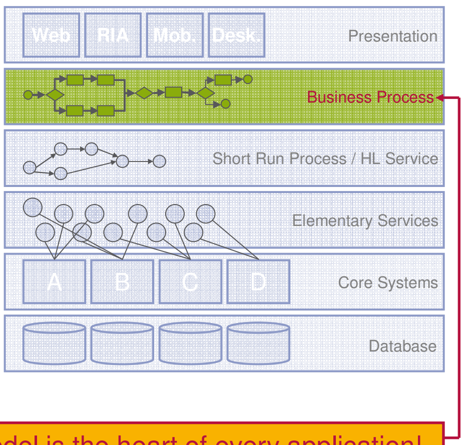
> **Let end-user customize software**
- Customization of applications should not have to be performed by the creator of the software.
- Rather, the end-user should be able to modify and extend software himself (at least to a certain degree).
- Acknowledge the millions of end-user programmers and domain experts that are using an application

> **Risks**
- Must control by whom and where manipulation of process model can take place

„The Dangers of End-User Programming“ by W. Henderson
IEEE Software Journal, Volume 21 , Issue 4, Pages 5-7
www.computer.org/portal/cms_docs_software/software/content/danger.pdf

Putting it together: A Vision of future Software

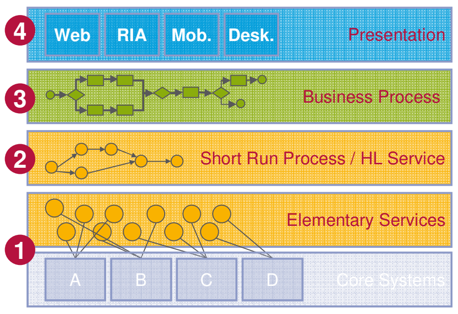
- Software has a process based SOA with a suitable front-end technology.
- The delivered software implements standard functionality.
- The process model is interpreted and modifiable. Can be edited at various levels of abstraction, allowing for end-user customization.
- Model is self-documenting and shows intention of the program.



The process model is the heart of every application!

How do we get there?

- New or legacy software is organized into elementary services
- Elementary services are wired together to form high-level services
- Business Process calls services, performs data transitions on the fly, implements workflow, interacts with the user, integrates systems.
- GUI actions drive (business) processes



What are the Technologies we should use?

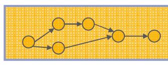
> Use BPEL for *simple* service orchestration

> Use a BPMS for process modeling and -management

> Use a RIA front-end for user interaction with the system

BPEL: Composing Services, Short Run Process

> BPEL (Business Process Execution Language) was created to "compose services into business processes" (and exposing the latter as services)



Short Run Process / HL Service

> The result of a BPEL composition is not actually a true business process, due to the lack of the language to support

– Workflow (Users, Roles, Tasks, Queues)
– Adapters and Transformations
– User Interaction
– Duration

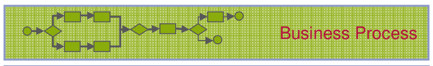
Support must be added outside the BPEL scope by using services

> However, BPEL is perfect for creating relatively short-run, head-less processes and for composing high level services from elementary services

BPM: Business Process Management

> BPM System (BPMS): A software managing business processes and the associated interactions between persons and systems.
- Workflow (modeling users, roles, tasks)
- User and system interaction
- Data transformation

> BPM includes
- Process *Analysis*
- Process *Design*
- Process *Execution*
- Process *Monitoring*



XPDL (XML Process Definition Language)

> Although many vendors advertise BPM capabilities, there are only a few completely integrated BPMS on the market capable of doing all of the above.

BPM: Sales Manager Talk (Buzzword Collection)

> The goals of BPM are
- Individualization
  Extensibility
- Flexibility
  Customization
- Simplicity
  Intentional Software

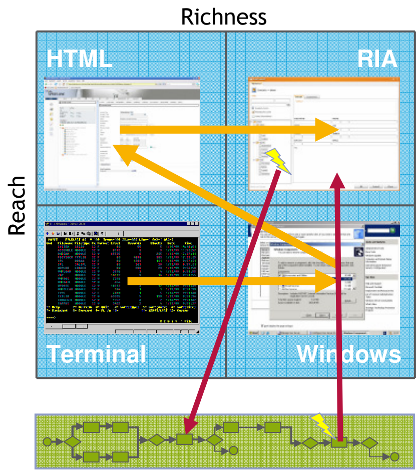
> Generally the promises of SOA and BPM are
- Business *Agility*
  Adaptation
- Business *Excellence*
- Business *Compliance*
  Self-documentation
- Business *Visibility*
  Intentional Software

… which results in Real Time Enterprise Management.

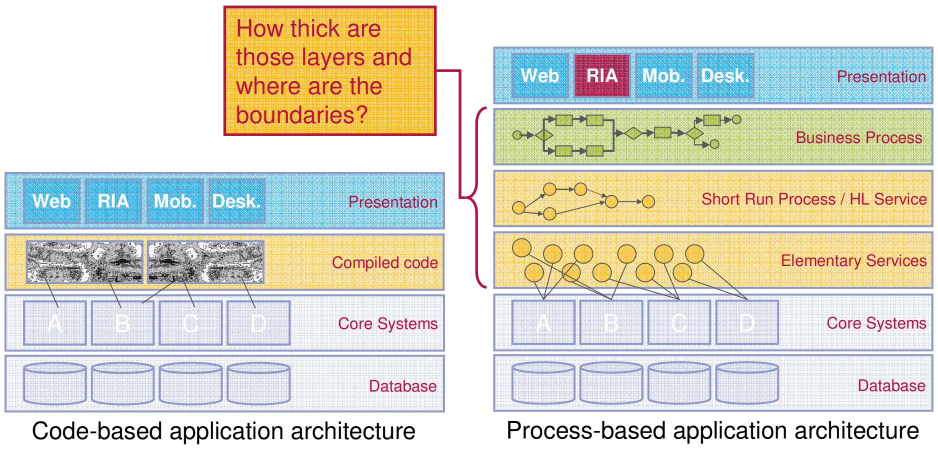
> All supported by process-based development!

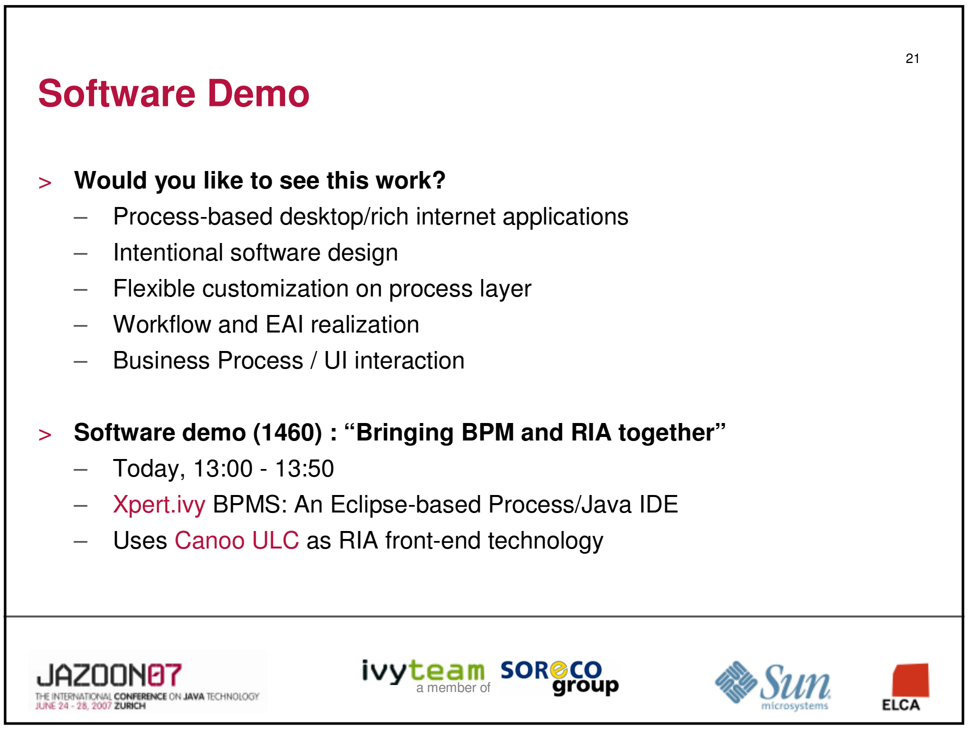
RIA: Implementing the Presentation Layer

- Richness vs. Reach
- RIA: Rich Internet Applications
  - Various technologies (e.g. ULC)
  - Distributed
  - Highly responsive
  - ? Zero Installation
  - ? Server-side execution
- User Interaction
  - Event from GUI triggers process
  - Process updates/alters GUI
- Data binding



Summary

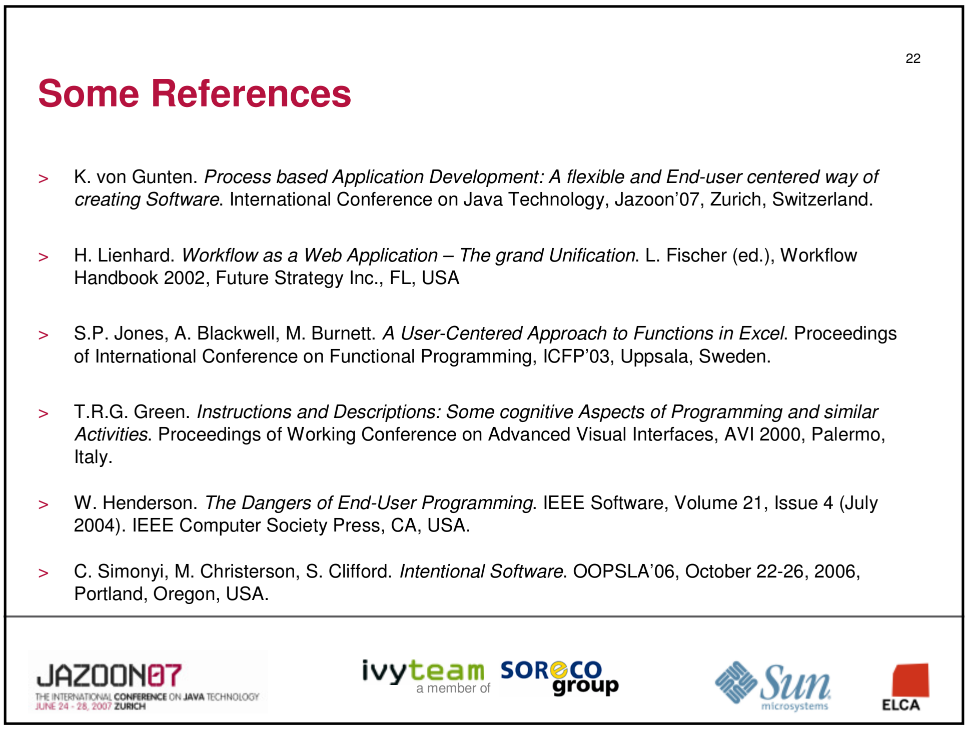

How thick are those layers and where are the boundaries?
Web
RIA
Mob.
Desk.
Presentation
Compiled code
A
B
C
D
Core Systems
Database
Code-based application architecture
Web
RIA
Mob.
Desk.
Presentation
Business Process
Short Run Process / HL Service
Elementary Services
A
B
C
D
Core Systems
Database
Process-based application architecture



Software Demo

> **Would you like to see this work?**
  – Process-based desktop/rich internet applications
  – Intentional software design
  – Flexible customization on process layer
  – Workflow and EAI realization
  – Business Process / UI interaction

> **Software demo (1460) : "Bringing BPM and RIA together"**
  – Today, 13:00 - 13:50
  – Xpert.ivy BPMS: An Eclipse-based Process/Java IDE
  – Uses Canoo ULC as RIA front-end technology



Some References

> K. von Gunten. *Process based Application Development: A flexible and End-user centered way of creating Software*. International Conference on Java Technology, Jazoon'07, Zurich, Switzerland.

> H. Lienhard. *Workflow as a Web Application – The grand Unification.* L. Fischer (ed.), Workflow Handbook 2002, Future Strategy Inc., FL, USA

> S.P. Jones, A. Blackwell, M. Burnett. *A User-Centered Approach to Functions in Excel.* Proceedings of International Conference on Functional Programming, ICFP'03, Uppsala, Sweden.

> T.R.G. Green. *Instructions and Descriptions: Some cognitive Aspects of Programming and similar Activities*. Proceedings of Working Conference on Advanced Visual Interfaces, AVI 2000, Palermo, Italy.

> W. Henderson. *The Dangers of End-User Programming.* IEEE Software, Volume 21, Issue 4 (July 2004). IEEE Computer Society Press, CA, USA.

> C. Simonyi, M. Christerson, S. Clifford. *Intentional Software.* OOPSLA'06, October 22-26, 2006, Portland, Oregon, USA.

Kaspar von Gunten
IvyTeam (Soreco Group)

http://www.ivyteam.ch
kaspar.vongunten@ivyteam.ch