WHAT not HOW

Anwendungsentwicklung durch Geschätsregeln

(The Business Rules Approach)

WAS und nicht WIE !!

Ein andere Technologie um Anwendungen zu entwickeln, basierend
Auf dem Business-rules-approach

Rainer Frank,
System Engineer Versata Germany
rfrank@versata.de

Stephan Klanck,
System Engineer Versata Germany
sklanck@versata.de

Die Art und Weise wie wir in Zukunft Anwendungen entwickeln werden wird sich dank einer neuen Technologie dramatisch ändern. Diese Technologie ist bekannt unter dem Begriff: **Entwicklung durch Geschäftsregeln (Business Rules Approach)**
Deklarative, bis zu einem hohen Grad automatisierte Anwendungsentwicklung ist der herausstechende Vorteil dieser Technologie. („*no coding anymore*“)

Wir wollen an diesem Abend die Technologie unter den folgenden Aspekten beleuchten:
Was ist der Business Rules Approach ?
Kann der Ansatz alle Probleme lösen ?
Wie kann eine Umgebung aussehen, die diesen Ansatz unterstützt ?
Vorabinformationen unter:www.versata.com

- **Die Technologie**
- **Ein Framework !**
- **Life-Demo**
- **Q & A**

Als **Computer** so in den frühen 50ern herauskamen, waren sie nur sehr schwer zu bedienen. Es waren besondere Fähigkeiten notwendig:
-EDV-Techniker
-Hardware Spezialist

Mir der Zeit wurden die Systeme bedienerfreundlicher. Diese Systeme sind durchweg durch einen steigenden **Abstraktionslevel** entstanden.

Viele werden jetzt sagen:

-Was Neues ? ich bleibe bei meiner alten und bewährten Art der Software-Entwicklung!

Schauen wir doch einmal zurück, zu den Ursprüngen der EDV.

1GL -> 2GL -> 3GL -> 4GL -> ...

1GL: Maschinensprache
2GL: Assembler
3GL: „*high level*“-Sprachen (COBOL,Fortran)
4GL: Propietäre Sprachen (PowerScript, evtl. auch SQL)

Sequential files -> indexed (ISAM) files -> hierachische und Netzwerk DB -> Relationale DB

Über die Jahre hat das System mehr und mehr Details des Speicherns und Managen der Daten übernommen – kurz: der Prozeß wurde automatisiert.

Wenn es um die Anwendungsentwicklung geht, übernehmen heute Compiler und IDEs einen großen Teil der Arbeit.

Im Bereich der Persistenz von Daten macht sich heute keiner mehr Sorgen darüber wo die Daten physikalisch auf der Festplatte liegen. Auch für das Auffinden und sorgen Systeme.

Tabellenkalkulationen und Scriptsprachen abstrahieren weiter zu bedienerfreundlichen Systemen. (Niemand würde heute ein Fortranprogram schreiben um eine Aufgabe dieser Art zu lösen)

Historisch ist ein Trend von ***prozedural*** hin zu ***deklarativ*** zu erkennen (das bedeutet „WHAT not HOW“)

HOW: bedeutet wie die Arbeit zu tun ist, Schritt für Schritt
WHAT: bedeutet nur zu beschreiben was zu tun ist

Was ist gut daran ?
Deklarativ (WHAT) heißt das System macht die Arbeit, wo hingegen Procedural (HOW) bedeutet der Anwender macht die Arbeit

Deklarativ ist beser als prozedural!

Excel, VBA, Perl ,... Für jeden Zweck das geeignete Mittel.

Der Trend ist zu erkennen !! Prozedural->deklarativ

Eine Anwendung ist die Implementierung von Funktionalitäten zum Erledigen von Aufgaben, z.B.:

- Einer Bestellung ein Teil hinzufügen
- Eine Bestellung löschen
- Den aktuellen Warenbestand aktualisieren
- ...

Einteilung in 3 Aspekte einer Anwendung:

1. User-Interface Aspekte
2. Daten-Persistenz Aspekte
3. Aspekte die die Geschäftslogik betreffen

Aspekte 1. und 2. Sind schon seit Langem weitestgehend automatisiert

Es ist Zeit die Geschäftslogik zu automatisieren

Was ist eine Applikation ??

Eine App. läßt sich grob in 3 Teile aufteilen:

-User Interface

-Daten-Persitenz

-Geschäftslogik

Kein Entwickler zeichnet heute noch manuell Buttons und Menüs. Dieser Schritt ist durch Systeme automatisiert.

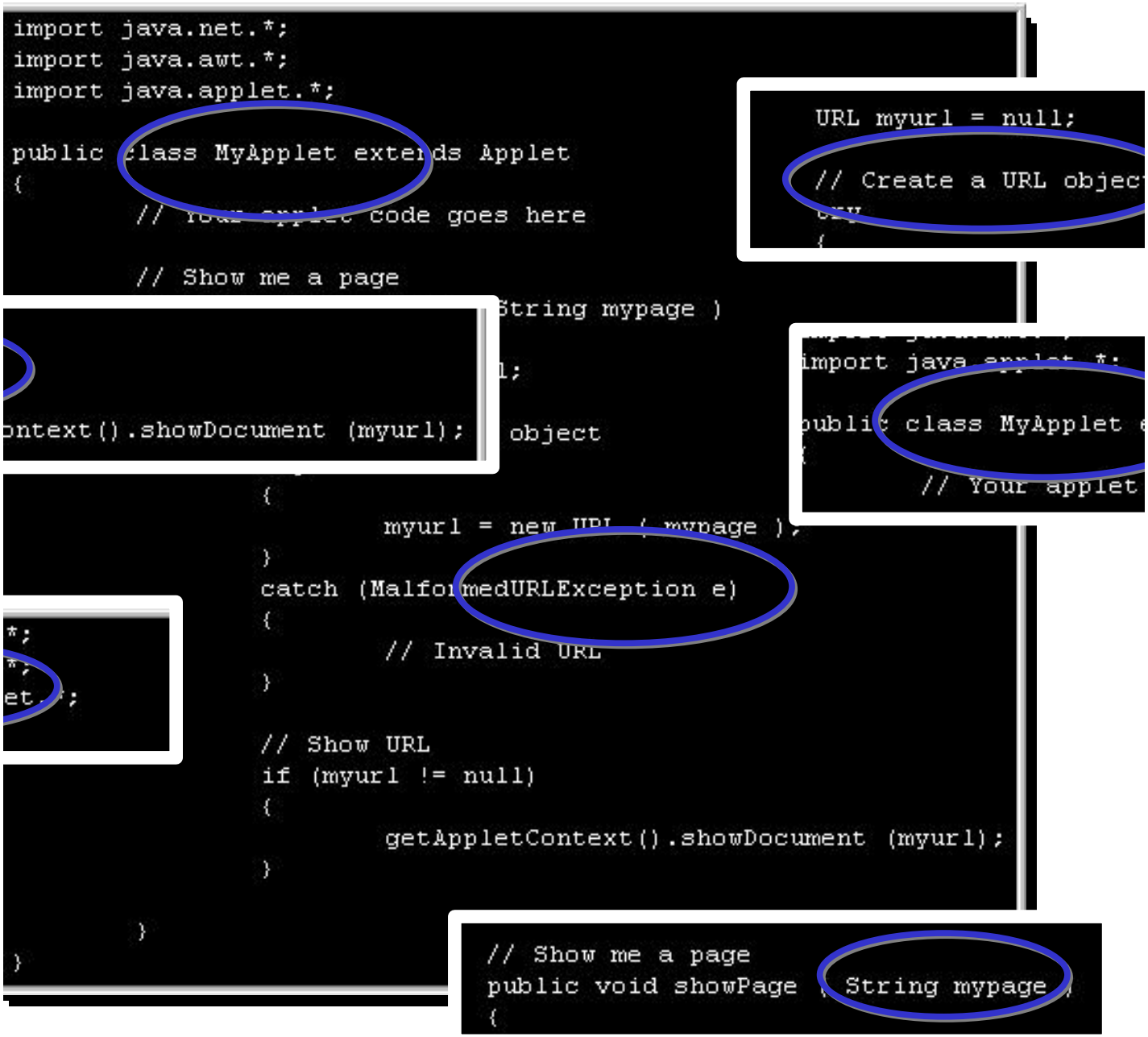
Bei der Datenpersistenz übernehmen DBMS einen großen Teil der Arbeit.

Wie sieht es bei der Geschäftslogik aus ??

- *Die Summe aller nichtbezahlten Aufträge sind die Aussenstände*
- *Bestellungen mit einem Wert >100k€ erfordern eine Bestätigung*
- *Aussenstände sind in rot darzustellen*
- *Es gibt kein Gehalt >5000€*
- *Der Urlaub muß bis April des Folgejahres eingereicht werden*
- ...

Um diese Regeln umzusetzen ist jede Menge handgeschriebener Code erforderlich. Code der entwickelt, getestet und gewartet werden muß.
Der Code ist in den meisten Fällen das Problem bei der Anwendungsentwicklung und –pflege.
Die prinzipielle Lösung ist:

Weg mit dem Codieren!


import java.net.*;
import java.awt.*;
import java.applet.*;
public class MyApplet extends Applet
{
// Your applet code goes here
// Show me a page
String mypage)
;
ontext().showDocument (myurl);
object
{
myurl = new URL (mypage);
}
catch (MalformedURLException e)
{
// Invalid URL
}
// Show URL
if (myurl != null)
{
getAppletContext().showDocument (myurl);
}
}
}
URL myurl = null;
// Create a URL object
{
import java.applet.*;
public class MyApplet
{
// Your applet
*;
*;
et.*;
// Show me a page
public void showPage (String mypage)
{

Years of experience with information system development have taught us two important lessons:

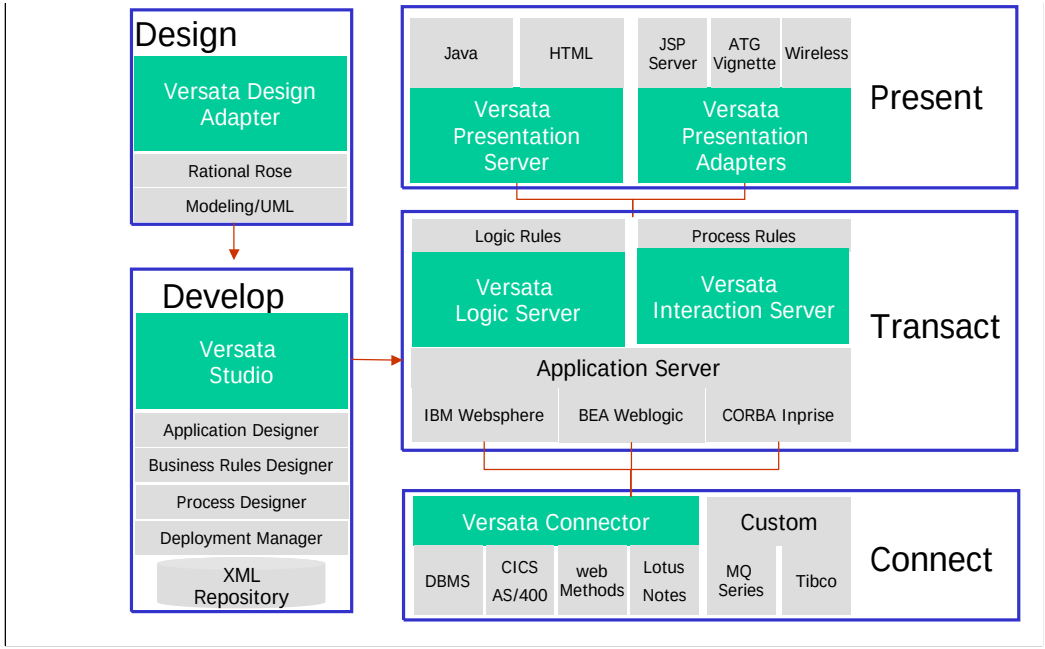
It takes far too long to turn a relatively simple set of requirements into a system that meets user needs, and

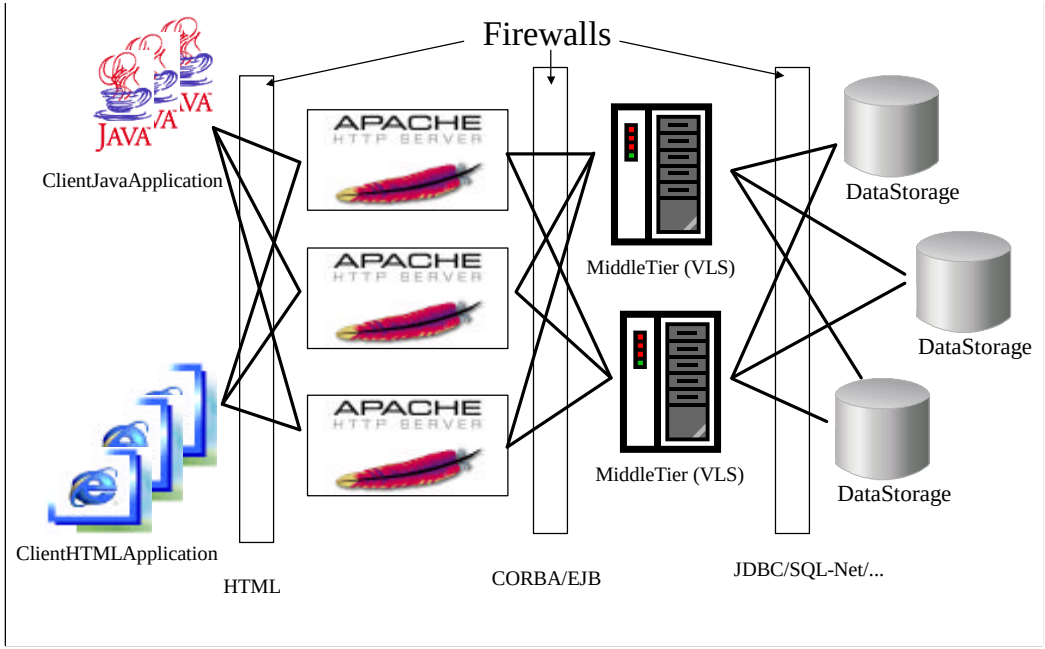
The cost of converting existing applications to new technologies is prohibitive ... The factor underlying both of these problems is the amount of code it takes to build a system ... If code is the problem, the only possible answer is to eliminate the coding by building systems directly from their specifications. That's what the rule-based approach does.

-Val Huber

Compilier- und ausführbare Spezifikationen!


versata™

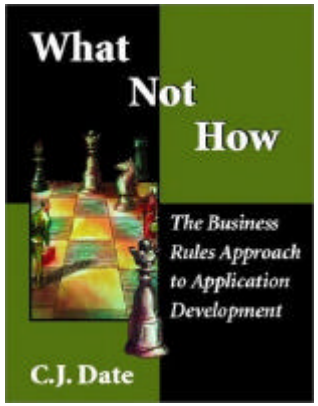

Design
Versata Design Adapter
Rational Rose
Modeling/UML
Develop
Versata Studio
Application Designer
Business Rules Designer
Process Designer
Deployment Manager
XML Repository
Java
HTML
JSP Server
ATG Vignette
Wireless
Versata Presentation Server
Versata Presentation Adapters
Present
Logic Rules
Process Rules
Versata Logic Server
Versata Interaction Server
Application Server
IBM Websphere
BEA Weblogic
CORBA Inprise
Transact
Versata Connector
Custom
DBMS
CICS AS/400
web Methods
Lotus Notes
MQ Series
Tibco
Connect


Firewalls
ClientJavaApplication
ClientHTMLApplication
APACHE
HTTP SERVER
APACHE
HTTP SERVER
APACHE
HTTP SERVER
MiddleTier (VLS)
MiddleTier (VLS)
DataStorage
DataStorage
DataStorage
HTML
CORBA/EJB
JDBC/SQL-Net/...

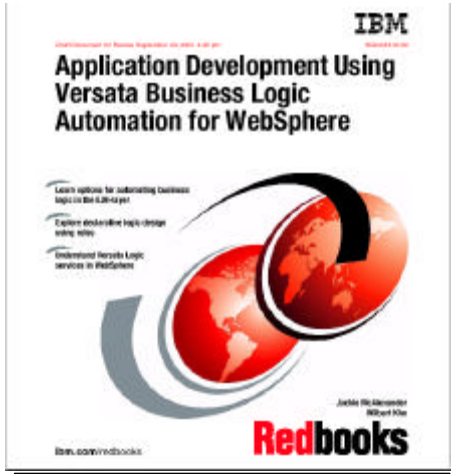
Der Ansatz ist immer dann erfolgreich, wenn die Anwendung:

- transaktionsorientiert ist
- der Prozeß in Geschäftsregeln formuliert werden kann

Loyale Kundenbasis



www.versata.com



IBM Redbook SG24-6510-00

- bestehendes Datenmodel
- Kunde – Auftrag
- Inter-/Intranetanwendung, HTML-basiert