

Universal Insurance Workplace: Eclipse-Technologie in Action

Ivo Eitner, Aspecta Lebensversicherung AG
Markus Herzog, Aspecta Lebensversicherung AG
Gernot Neppert, Aspecta Lebensversicherung AG
Martin Lippert, akquinet agile GmbH

Vortrag im Arbeitskreis Objekttechnologie
2. Oktober 2006



Überblick über den Vortrag

- Die Ausgangslage
- Das Architektur des GPO-Systems
- Plattform-basierte Entwicklung
 - flexible Erweiterbarkeit auf Basis von Eclipse
- Lessons Learned
 - Was würden wir wieder so machen
 - Was würden wir anders machen
- Fazit

Die Ausgangslage

Die wichtigsten Verwaltungsprozesse waren zu teuer und zu langsam.



Ursachen:

- Ineffektive Prozesse
- Schlechte IT-Unterstützung der Anwender durch
 - Gewachsene Systemlandschaft
 - Verschiedenste Technologien
 - Medienbrüche

Folge:

- Der Prozess orientiert sich an den Systemen statt umgekehrt.

Resultat:

- + Große Flexibilität
- Hohe Personalkosten
- Hohe Laufzeiten
- Geringe Transparenz



Geschäftsprozess-Optimierung

Projekt zur Geschäftsprozessoptimierung (GPO)

Qualität verbessern – Kundenzufriedenheit erhöhen – Kosten senken

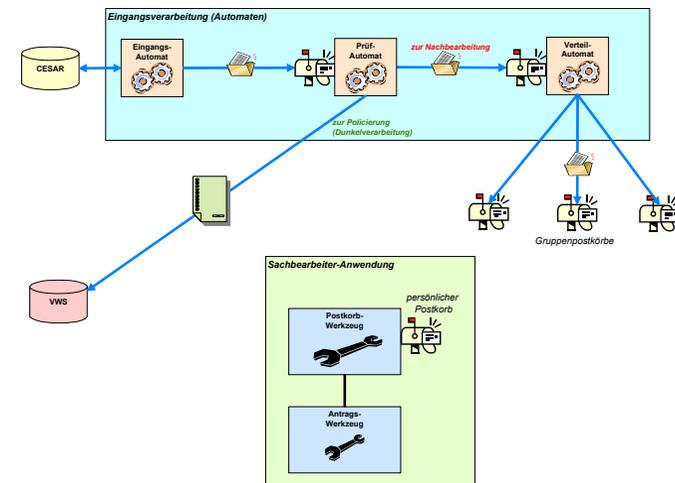
Analyse :

- Bestandsaufnahme
 - Prozesse, Stückzahlen, Durchlaufzeiten
 - Identifikation der Optimierungskandidaten
- Definition der fachlichen Sollprozesse



Umsetzung :

- Organisatorische Veränderungen
- Bereitstellung einer technischen Prozessunterstützung :
 - Der Prozess bestimmt die Technik
 - Vollständige Nutzung der vorhandenen Möglichkeiten (frühes Scannen)
 - Beseitigung von Medienbrüchen



Der Universal Insurance Workplace



Die Idee

Entwicklung einer integrierten Anwendung

- Verbergen der Systemvielfalt
- Verdichten der relevanten Informationen
- Automatisieren einfacher Bearbeitungsschritte

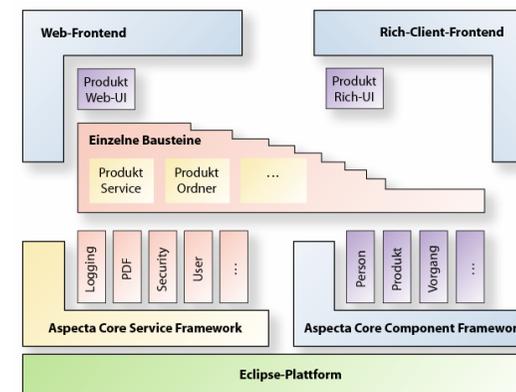
Nutzung vertrauter Metaphern:

- Vorgangsmappe
- Postkorb
- Notizblock
- Werkzeug
- ...



Schaffung einer zukunftsfähigen Basisarchitektur

- Erweiterbarkeit (weitere Geschäftsprozesse)
- Skalierbarkeit (weitere Anwender)



Hochintegrierte Sachbearbeitung

Das Ergebnis (am Beispiel des Antragssystems)

Vorgang: Neuantrag
Policennr. 350200253 Versicherungsnehmer Vorgangsnummer 16351

Hinweisblatt
Es liegen 6 schwerwiegende Probleme, 7 Probleme und 4 Hinweise vor.

Allgemeine Hinweise

- Die Antragsdaten wurden noch nicht in das Antragstracking übernommen.
- Die Antragsdaten wurden noch nicht in die Antragsstatistik überführt. Bitte tragen Sie die fehlenden Daten nach (Beitragssumme).
- Ort wurde automatisch von "Verden-Walle" auf "Verden (Aller)" korrigiert

Annahmerichtlinien verletzt

- falscher Dynamik-Beginn ok
- restliche Zahlungsdauer zu klein ok
- Fehler in Calciterative ok
- Zahlungsdauer der HV zu klein ok

Fehlende Angaben

- Es fehlt die Nationalität ok
- Der erlernte Beruf der 1. versicherten Person ist unbekannt.
- Einmalbeitrag: keine Dynamik möglich
- Die Legitimation ist unvollständig. Die fehlenden Angaben vom VN nachfordern. ok

Prüfungen

- Beitragszahlungsdauer zu klein
- Das Alter der VP am Ende der Aufbauphase muss mindestens 60 und höchstens 70 sein.
- Das Beginndatum liegt in der Vergangenheit. Rückwirkende Policierung ist möglich. ok
- Ungültige Zahlweise für diesen Tarif. Bitte prüfen.
- Der Fonds in Zeile 1 ist ungültig. Bitte einen aus der Liste auswählen.

Vorvertrag

- Es wurden folgende Vorverträge für V. gefunden

Mandant: **Aspecta**
Versicherungsscheinr. **Todesfallschutz**
350200230 unbekannt

Neuantrag - Allgemeines

Allgemeines

Tarif: 50 Tarifschlüssel: ADE10500000
Schichtkennzeichen: 1 Produktschlüssel: ADE10500000217B - PL
Aktionskennzeichen: [Dropdown]

Vermittler

Orga: 211138 VD: 044 Mandant: A
Agentur: tecis Finanzdienstleistungen AG Kundenservice
Straße/Postf.: Alter Teichweg 17
PLZ: 22081 Ort: Hamburg
Telefon: 040-69696969
Fax: 040-69696979
Email: Kundenservice@tecis.de
vermittlungsspezifisches Merkmal: [Empty field]

Zurück Weiter

Vertriebs-
Partner-
System

ASPECTA

Seite 6

02.10.2006

Hochintegrierte Sachbearbeitung

Das Ergebnis : Verbergen der Backendsysteme

The screenshot shows a software interface for processing insurance applications. The interface is divided into several sections:

- Header:** "Vorgang: Neuantrag" (Process: New Application), "Testversion!" (Test Version!).
- Navigation:** "Antrag ablegen" (Submit Application), "Antrag komplett prüfen" (Check Application Completely), "Antrag polizieren" (Process Application), "Antrag überführen" (Transfer Application), "Bestandssystem anzeigen" (Show Status System), "Neuen Brief anlegen" (Create New Letter), "Policierung im VWS" (Policy in VWS), "Manuellen Termin anlegen" (Create Manual Appointment).
- Left Panel (Inhalt):** A tree view showing the application structure: "Antrag" (Application) -> "Neuantrag - Wissmann" (New Application - Wissmann) -> "Allgemeines" (General) -> "Partner" (Partner), "Antrag (techn)" (Application (technical)), "Gesundheitserklärung" (Health Declaration), "Antrag (jur)" (Application (legal)), "Zahlweg, Legitimation" (Payment Method, Legitimation), "Votum" (Vote). Other options include "Scan", "Scan Neuantrag", "Notizblock (0)", "Hinweisblatt", and "Laufzettel".
- Main Content Area (Hinweisblatt):** A list of messages and warnings. The "Allgemeine Hinweise" (General Notes) section contains:
 - Die Antragsdaten wurden noch nicht in das Antragstracking übernommen.
 - Die Antragsdaten wurden noch nicht in die Antragsstatistik überführt. Bitte tragen Sie die fehlenden Daten nach (Beitragssumme).
 - Ort wurde automatisch von "Verden-Waale" auf "Verden (Aller)" korrigiert.The "Annahmerichtlinien verletzt" (Violated Acceptance Guidelines) section contains:
 - falscher Dynamik-Beginn (wrong dynamic start) [ok]
 - restliche Zahlungsdauer zu klein (remaining payment period too small) [ok]
 - Fehler in Calciterative (error in Calciterative) [ok]
 - Zahlungsdauer der HV zu klein (payment period of HV too small) [ok]The "Fehlende Angaben" (Missing Information) section contains:
 - Es fehlt die Nationalität (Nationality is missing) [ok]
 - Der erlernte Beruf der 1. versicherten Person ist unbekannt (The learned profession of the 1st insured person is unknown).
 - Einmalbeitrag: keine Dynamik möglich (One-time contribution: no dynamics possible).
 - Die Legitimation ist unvollständig. Die fehlenden Angaben vom VN nachfordern. (The legitimation is incomplete. Request the missing information from the VN.) [ok]The "Prüfungen" (Checks) section contains:
 - Beitragszahlungsdauer zu klein (Contribution payment period too small).
 - Das Alter der VP am Ende der Aufbauphase muss mindestens 60 und höchstens 70 sein. (The age of the VP at the end of the build-up phase must be at least 60 and at most 70).
 - Das Beginndatum liegt in der Vergangenheit. Rückwirkende Policierung ist möglich. (The start date is in the past. Retroactive policy is possible.) [ok]
 - Ungültige Zahlweise für diesen Tarif. Bitte prüfen. (Invalid payment method for this tariff. Please check.)
 - Der Fonds in Zeile 1 ist ungültig. Bitte einen aus der Liste auswählen. (The fund in row 1 is invalid. Please select one from the list.)The "Vorvertrag" (Pre-contract) section contains:
 - Es wurden folgende Vorverträge für V... gefunden (The following pre-contracts were found for V...)Below this, it shows:
 - Mandant: **Aspecta**
 - Versicherungsschein: **Todesfallschutz**
 - 350200230 unbekannt
- Right Panel (Neuantrag - Allgemeines):** A form for entering general application data:
 - Alter: 50
 - Schichtkennzeichen: 1
 - Aktionskennzeichen: [dropdown]
 - Produktsschlüssel: ADE10500000217B - PL
 - Ordnr.: 211138 VD: 044
 - Agentur: tecis Finanzdienstleistungen AG Kunden...
 - Straße/Postf.: Alter Teichweg 17
 - PLZ: 22061 Ort: Hamburg
 - Telefon: 040-6969999
 - Fax: 040-6969979
 - Email: kundenservice@tecis.de
 - vermittlerspezifisches Merkmal: [input field]

Arrows from yellow boxes on the right point to specific parts of the interface, indicating the integration of external systems:

- Adress-Pruefung** (Address Check) points to the "Ordnr." and "VD" fields.
- Partner-System** points to the "Agentur" field.
- Rechen-kern** (Calculation Core) points to the "Alter" field.
- Risiko-System** (Risk System) points to the "Produktsschlüssel" field.
- Annahme-Richtlinien** (Acceptance Guidelines) points to the "Hinweisblatt" section.
- Fonds-Liste** (Fund List) points to the "Prüfungen" section.
- Data-Warehouse** points to the "Vorvertrag" section.

Hochintegrierte Sachbearbeitung

Das Ergebnis : schnellen Überblick verschaffen, fallabschließende Bearbeitung

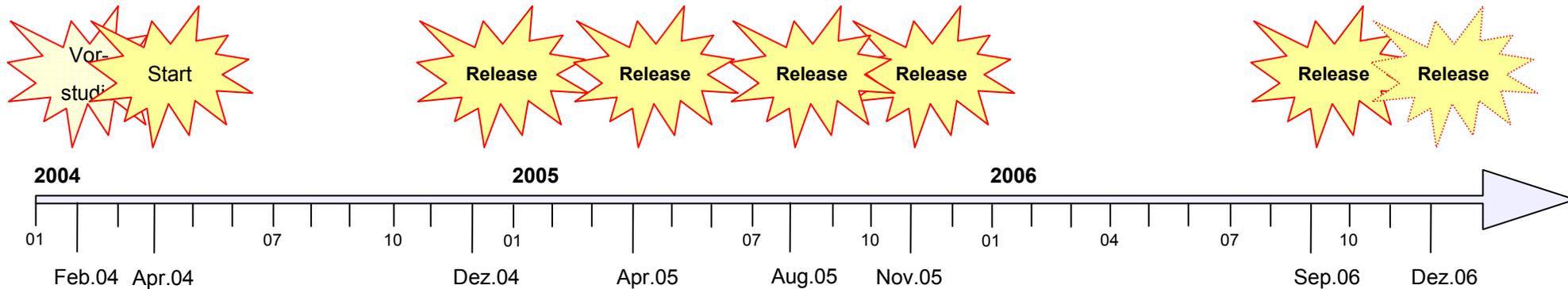
ASPECTA

Seite 8

02.10.2006

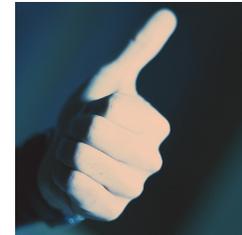
Das Projekt im Überblick

Ein kurzer Blick zurück



Ergebnis : *„Wir steuern in die richtige Richtung...“*

- Erhebliche Reduktion der Prozesskosten
- Verringerung der Durchlaufzeiten um > 50 %
- Deutliche Erhöhung der Prozessqualität
- Hohe Anwenderakzeptanz



ASPECTA

Überblick

- Fachliche Architektur
- Technische Architektur
- Modularisierung
- Plattform
- Erweiterbarkeit

Abbildung des Sachbearbeiter-Arbeitsplatzes auf die technische Welt

- Vorgangsmappen enthalten Dokumente
 - Alles, was zu einem Geschäftsvorfall gehört
 - Bildet eine Bearbeitungs-Einheit
- Dokumente können beispielsweise sein:
 - Scans, Anträge, Vertragsänderungen, Briefe, Termine, Notizen, etc.
- Vorgangsmappen können geöffnet, bearbeitet und gespeichert werden
 - Immer nur von einem Sachbearbeiter zur Zeit
- Postkörbe sind Aufbewahrungsorte für Vorgangsmappen
 - Vorgangsmappen können in Postkörbe weitergeleitet werden
 - Unterscheidung in persönliche, Gruppen- und technische Postkörbe

Konzepte

- Services
 - Stellen Funktionalität zur Bearbeitung der Business Objects zur Verfügung (z.B. Vorgang-Service, Bankprüf-Service)
 - Bestehen aus Interface, Factory und (ggf. unterschiedlichen) Service-Implementierung(en)
 - Werden über zentralen Service-Manager zur Verfügung gestellt
- Business Objects (BO)
- (Fach-)Werte

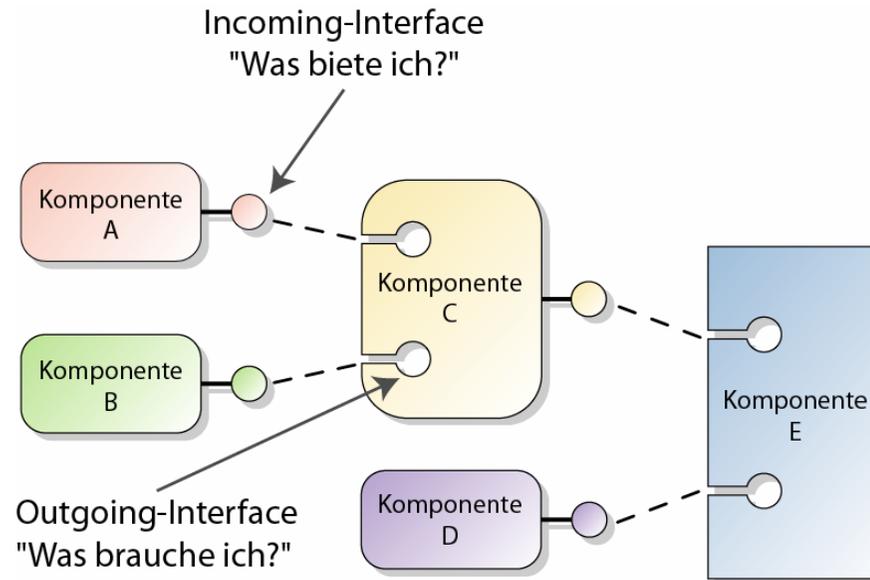
- Werkzeuge (UIs)
 - Rahmen ist Postkorbwerkzeug (Anzeige der Postkörbe mit den enthaltenen Vorgangsmappen)
 - Einzelne Vorgangsmappe wird in einem Vorgangsmappen-Werkzeug dargestellt
 - Unterschiedliche Dokumente werden mit unterschiedlichen Dokument-Bearbeitern dargestellt
- Automaten
 - Dienen zur Automatisierung von Abläufen (Folgepost zuordnen, Prüfen, etc.)
 - Dienen zur Entkopplung von Abläufen (Briefschreibung, Rückfrage zuordnen)
 - Anwendungen ohne UI, beruhend auf der selben technologischen Basis

Die grundlegende Architektur – technisch (2/2)

- Die Basis:
 - **Java 5**
 - **Equinox OSGi** Runtime (Eclipse-Plugin-Technologie)
 - **Extension-Registry** von Eclipse
- Weitere eingesetzte Technologien:
 - **Swing** (fürs UI)
 - **JDO** und **JDBC** (für die Persistenz)
 - **Spring** (fürs Remoting)
 - **JDBC**, **JNI**, **Axis**, etc. (für die Anbindung externer Systeme)
- Testing
 - Unit-Tests mit **JUnit**
 - Integrations-Tests mit **FIT** (noch am Anfang)
 - Hilfsmittel: **Mockrunner** (hauptsächlich JDBC-Mocks), **Easymock** (für alle anderen Mocks)

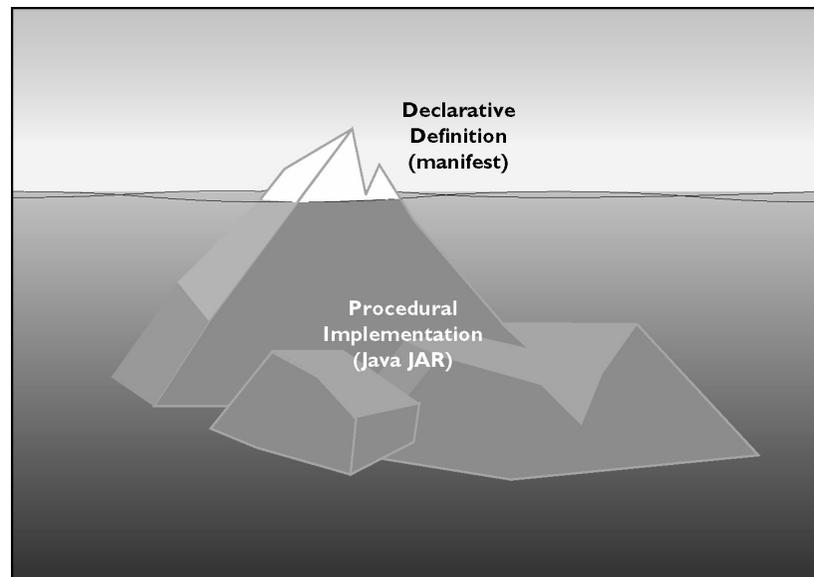
Was bedeutet Modularisierung mit Eclipse Plugins und OSGi?

- Separate Einheiten
 - Ein eigener Classpath pro Modul
- Definierte Abhängigkeiten
 - Keine Zyklen
- Expliziter Im- und Export von Packages und Klassen



Mehr als Abhängigkeits-Management...

- Trennung von Deklaration und Code mittels Manifest-Dateien
 - Dadurch gute Skalierbarkeit auch für große Systeme (mehrere tausend Plugins)
 - Schnelles Classloading



Aber: Nur Module/Plugins selbst reichen nicht

Was passiert, wenn mein System aus mehreren hundert oder tausend Plugins besteht?

- Wir brauchen ein weiteres Strukturierungsmittel:
 - Erster Ansatz: Features in Eclipse
 - Fassen Plugins zu Features zusammen, als deploybare Einheit
 - Abhängigkeiten zwischen einzelnen Plugins trotzdem möglich
 - Zweiter Ansatz: Eine Plattform
 - Gemeinsamkeiten zwischen verschiedenen abgebildeten Geschäftsprozessen werden in einer allgemeinen Plattform für Lebensversicherungs-Anwendungen implementiert
 - Spezifische Geschäftsprozesse werden unabhängig voneinander auf Basis der Plattform implementiert

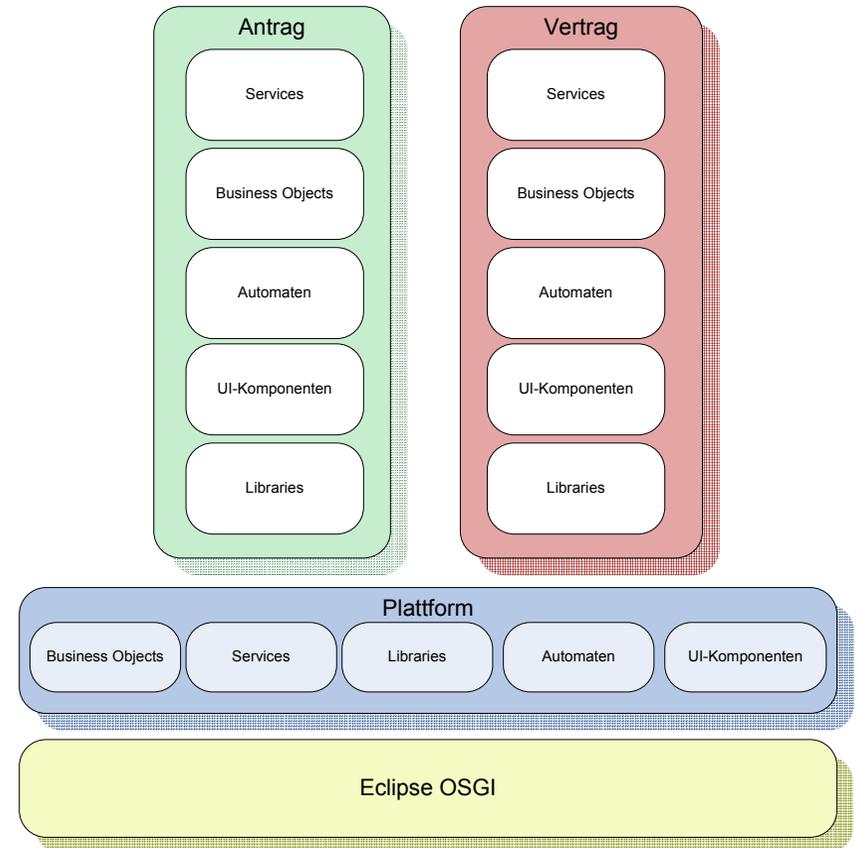
Plattform-basierte Entwicklung

Eine gemeinsame Plattform Geschäftsprozesse als Säulen

- + Integrierte Benutzungsoberfläche mit homogenem Verhalten
- + Leichte Wiederverwendung gemeinsamer Konzepte und Komponenten
- + Entwicklung lässt sich gut skalieren, indem Teams parallel an unterschiedlichen Säulen implementieren können

Nicht neu erfunden, sondern abgeschaut bei Eclipse:

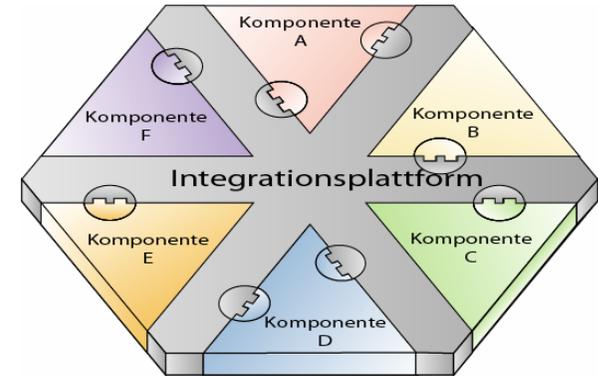
- Bei Eclipse gibt es die IDE-Plattform, auf deren Basis unterschiedliche IDEs implementiert werden können



Was bedeutet das?

Plattform-basierte Entwicklung

- Kernkonzepte werden in der Plattform implementiert
 - Gemeinsamkeiten heraus-faktorisieren
 - Kernkonzepte identifizieren und implementieren
- Spezialitäten werden außerhalb implementiert
 - Können auf den Kernkonzepten aufbauen, das fördert die fachliche Integration
 - Es stellt sich die Frage, wie die Plattform spezialisiert werden kann
- Kann nicht Up-Front passieren
 - Plattform-Entwicklung ist ein evolutionärer Prozess



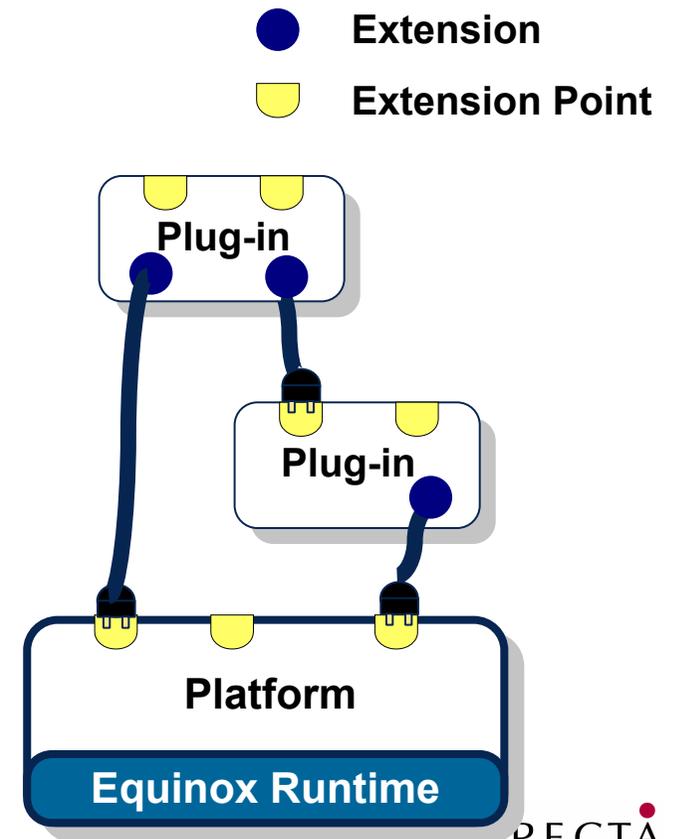
Erweiterbarkeit ist entscheidender Faktor

- Erweiterbarkeit ergibt sich nicht von selbst
 - Erweiterbare Teile der Plattform müssen sorgfältig entworfen und implementiert werden
 - *„Der erste Wurf ist fast immer falsch oder unpassend“*
- Möglichkeiten für Erweiterungs-Mechanismen
 - Benutzung und Vererbung
 - Dependency Injection
 - **Extension-Points**
- Wichtig:
 - **Klar definieren und dokumentieren, was erweiterbar ist und was nicht**

Extension-Points und Extensions

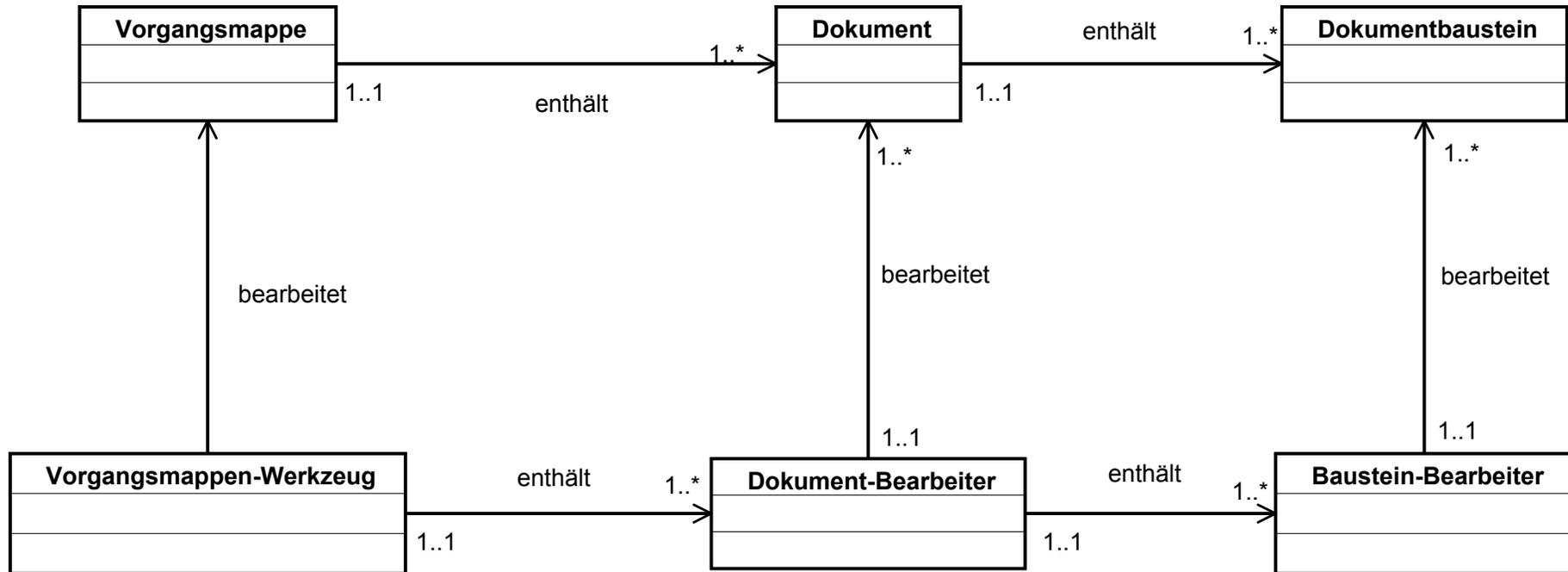
Der Extension-Point-Mechanismus von Eclipse eignet sich hervorragend, um eine eigene erweiterbare Plattform zu realisieren

- Der Anbieter eines Extension-Points definiert, was er von Extensions erwartet
 - Schema
 - Interface
 - Ggf. weitere Dokumentation
- Der Anbieter ermittelt zur Laufzeit, welche Extensions existieren und nutzt diese entsprechend
- Der Lieferant einer Extension erfüllt den Vertrag des Extension-Points und fügt seine Implementation auf diesem Wege dem Anbieter hinzu



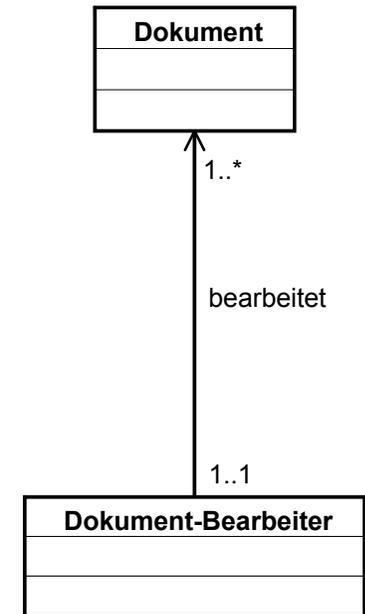
Beispiel: Vorgangsbearbeitung

Strukturübersicht



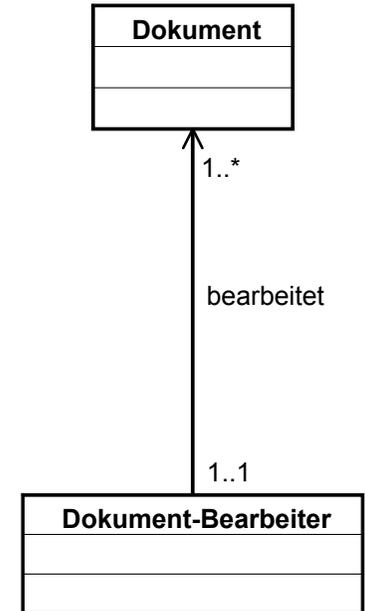
Beispiel: Vorgangsbearbeitung -> Dokument-Bearbeiter

- Dokument-Bearbeiter ist ein Java Interface und ein Extension Point
- Java-Klassen, die Dokument-Bearbeiter implementieren, enthalten den vollständigen Code, um Dokumente eines Typs darstellen und editieren zu können
- Jede Extension, die sich hier anmeldet, deklariert, für welchen Typ von Dokument sie zuständig ist
- Die Zuordnung von Dokument-Typen und ihren zuständigen Bearbeitern erfolgt ausschließlich über die Konfiguration (plugin.xml)



Beispiel: Vorgangsbearbeitung -> Dokument-Bearbeiter

- + Man kann flexibel neue Dokumente und ihre entsprechenden Bearbeiter dem System hinzufügen, ohne das Vorgangsmappen-Werkzeug selbst zu verändern
- + Je nachdem, welche Plugins man ausliefert, kann man Werkzeuge mit verschiedenen „Sichten“ auf dieselben Vorgangsmappen ausliefern
 - Es gibt bei uns beispielsweise einen vollständigen Antrags-Bearbeiter für die Hauptverwaltung und einen eingeschränkten Antrags-Viewer für die Geschäftsstellen



Dokument-Bearbeiter: Antrag

Vorgang: Neuantrag - 350117404 - 2373

Vorgang Antrag Brief Policierung Rückfrage Wiedervorlage **Entwicklung**

Antrag ablegen Antrag komplett prüfen Antrag policieren Antrag überführen Bestandssystem anzeigen Neuen Brief anlegen Policierung im VWS Rückfrage VD In Wiedervorlage Manuellen Termin anlegen

Policennr 350117404 Versicherungsnehmer [redacted] Vorgangsnummer 2373

Inhalt

- Antrag
 - Neuantrag - Scazzari, Ma
 - Allgemeines
 - Partner
 - Antrag (techn)**
 - Gesundheitserklärung
 - Antrag (ju)
 - Zahlweg, Legitimation
 - Votum
 - Scan
 - 11.08.06 Neuantrag
 - Notizblock (0)
 - Hinweisblatt
 - Laufzettel

Hinweisblatt

Es liegen 4 schwerwiegende Probleme, 3 Probleme und 3 Hinweise vor.

ACHTUNG !!!

Antrag wurde mit der allgemeinen Maske erfasst. Es fehlen wichtige Daten.

Allgemeine Hinweise

- Die Antragsdaten wurden noch nicht in die Antragsstatistik überführt. Bitte tragen Sie die fehlenden Daten ein (Beitragssumme, HDI-OB).
- Die Antragsdaten wurden noch nicht in das Antragstracking übernommen.

AVS

- Verbindung zum AVSService kann nicht hergestellt werden.

Der Schnellerfasser hat folgende Probleme gemeldet:

- Es sind Anlagen zu diesem Antrag vorhanden ok

Fehlende Angaben

- Die Legitimation ist unvollständig. Die fehlenden Angaben vom VN nachfordern. ok

Prüfungen

- Die Gesundheitserklärung ist nicht mehr gültig, da mehr als 6 Monate zwischen Antragsdatum und Versicherungsbeginn liegen ok
- Falsche Beitragssumme für Hauptversicherung [ersetze durch 24.000](#)
- Die Summe der Fondsanteile ergibt nicht 100%

Vorvertrag

- Es wurden folgende Vorverträge für [redacted] gefunden

Mandant: **Aspecta**

Versicherungsscheinr.	urspr.	Todesfallschutz	urspr.	Monatsprämie
315776351		irrelevant		50,00 €
aktuelle		irrelevant		100,00 €
350117672		irrelevant		-

Neuantrag - Antrag (techn)

Allgemeines

Tarif: 50 Tarifschlüssel: ADE10500000
 Schlüsselkennzeichen: 1 Produktschlüssel: ADE10500000179B-
 Aktionskennzeichen: PLUS

Technische Daten

Vers.-beginn: 01.10.2006 Eintrittsalter der VP1 (Jahre): 42
 Beitragszahlungsdauer (Jahre): 20 Aufbauphase (Jahre): 20
 Zahlungsweise: 1/12 Zahlbeitrag €: 100
 Beitragssumme €: 0

Zuzahlung

Beitragsvorauszahlung

Dynamik

Dynamik: Keine Intervall (Monate):

Fonds

Fonds	ISIN	Anteil (%)
Entfernen Hinzufügen		

BUZ

Gesamtbeitrag

Gesamtbeitrag €: 100,00

HZV

Dokument-Bearbeiter: Scan

Vorgang: Neuantrag - 350117404 - 2373

Vorgang Antrag Brief Policierung Rückfrage Wiedervorlage **Entwicklung**

Antrag ablegen Antrag komplett prüfen Antrag polizieren Antrag überführen Bestandssystem anzeigen Neuen Brief anlegen Policierung im VWS Rückfrage VD In Wiedervorlage Manuellen Termin anlegen

Policennr 350117404 Versicherungsnummer Vorgangsnummer 2373

Inhalt

- Antrag
- Partner
- Antrag (techn)
- Gesundheitserklärung
- Antrag (jur)
- Zahlweg, Legitimation
- Votum
- Scan
- 11.08.06 Neuantrag
- Seite 1
- Seite 2
- Seite 3
- Notzettel (0)
- Hinweisblatt
- Lautzettel

11.08.06 Neuantrag - Seite 1

Antrag auf eine fondsgebundene Rentenversicherung

ASPECTA Lebensversicherung AG
 22399 Hamburg - Schwartstr. 119 (22399) | 22399 Hamburg - Lomstedter Str. 101 (22399) | 22399 Hamburg - Lomstedter Str. 101 (22399)
 Tel: (040) 6 57 8 3 31 - Fax: (040) 6 20 94 77 | www.aspecta.com

Origin. Nr. 085142 VSM
 VD ADE 10 500 000 PLUS Invest basic FRAU Invest basic GARANT Invest basic

Vorname: MANUELA Nach-Nr.: 209

PLZ: 65775 Wohnort: KELKHE: M
 Ort: Offenbach
 Beruf: Erzieherin
 Geburtsdatum: 12.06.1964
 Berufstätigkeit: Erzieherin

Technische Vertragsdaten

Beitragszahlungswise: laufende Beitrag
 Depot (siehe Depotvereinbarung)
 Dynamische Anpassung: 5% Dynamik
 Beitragsbetriebl: 1/12
 Beitragszahlungswise: 1/12
 Beitragszahlungswise: 24
 Beitragszahlungswise: 50,00
 Beitragszahlungswise: 100,00

Gesamteinlösungsbeitrag: EUR 50,00

Gesundheitserklärung (nur bei Einschluss der Berufsunfähigkeits-Zusatzversicherung)

Erklärungen der zu versichernden Person: Nur Beitragsbefreiung (bis zu einer Beitragssumme von max. EUR 100.000,-)...

Erklärungen zu den mit [X] beantworteten Fragen (weitere Erklärungen, insbesondere auch Namen und Anschriften weiterer Ärzte/Krankenkassen, ggf. auf gesondert unterschriebener Anlage geben):

Welche(r) Arzt/Ärztin (über Ihre Gesundheit am besten informiert) (Vollständige Adresse):

Bestimmte der zuzurechnenden Person: Ehepartner, Kind, Geburtsdatum: 21.07.1987

Wurden bei Antragstellung Besondere Vereinbarungen getroffen? Welcher Der Antrag gilt nur, wenn die Besonderen Vereinbarungen zustande kommen.

Einzugs-ermächtigung (Doppel einseitig zugrund erhaltend bei laufender Beitragszahlung): Citibank Düsseldorf

Legitimation des Versicherungsvertragsnehmers: gültiger Personalausweis, gültiger Reisepass

Wichtige Hinweise: Bitte bei diesem Antrag unterschreiben, lassen Sie bitte die Schlussklärung des Antragstellers und der zu versichernden Person auf der Rückseite, diese Erklärung enthält Ermächtigungen zur Entbindung von der Schweigepflicht und zur Datenverarbeitung und informiert Sie über Ihr Widerrufsrecht bei der Fondsbeteiligung...

Ort: Kelkheim Datum: 15.09.05 Unterschrift des Vermittlers: [Signature]

Neuantrag - Antrag (techn)

Allgemeines

Tarif: 50 Tarifschlüssel: ADE10500000
 Schichtkennzeichen: 1 Produktschlüssel: ADE1050000179B-PL
 Aktionskennzeichen: PLUS

Technische Daten

Vers.-beginn: 01.10.2006 Eintrittsalter der VP1 (Jahre): 42
 Beitragszahlungsdauer (Jahre): 20 Aufbauphase (Jahre): 20
 Zahlungsweise: 1/12 Zahlungsbeitrag €: 100
 Beitragssumme €: 0

Zahlung

Beitragsvorauszahlung

Dynamik

Fonds

ISIN Anteil (%)

Entfernen Hinzufügen

BUZ

Gesamtbeitrag

Gesamtbeitrag €: 100,00

HIV

PostkorbWerkzeug



Dokument-Bearbeiter: Laufzettel

The screenshot displays a software interface for document processing. The main window is titled 'Vorgang: Neuantrag - 350117404 - 2373'. The interface is divided into several sections:

- Top Bar:** Contains navigation tabs like 'Vorgang', 'Antrag', 'Brief', 'Policierung', 'Rückfrage', 'Wiedervorlage', and 'Entwicklung'. Below these are various action buttons such as 'Antrag ablegen', 'Antrag komplett prüfen', 'Antrag policieren', 'Antrag überführen', 'Bestandssystem anzeigen', 'Neuen Brief anlegen', 'Policierung im VWS', 'Rückfrage VD', 'In Wiedervorlage', and 'Manuellen Termin anlegen'.
- Left Sidebar (Inhalt):** Shows a tree view of the document structure. The 'Laufzettel' entry is highlighted in blue, and a black arrow points to it from the right.
- Table (Laufzettel):** A table with columns 'Status', 'Datum', and 'Bearbeiter'. It contains three entries:

Status	Datum	Bearbeiter
Eingang VD	28.09.2005 00:00	
Eingang HV	05.10.2005 00:00	
Der Vorgang wurde neu angelegt	11.08.2006 09:46	
- Form (Neuantrag - Antrag (techn)):** A form for entering technical data. It includes fields for 'Allgemeines' (Tarif, Schichtkennzeichen, Aktionskennzeichen), 'Technische Daten' (Vers.-beginn, Beitragszahldauer, Zahlungsweise, Beitragssumme, Eintrittsalter, Aufbauphase, Zahlbeitrag), 'Dynamik', and 'Fonds'. The 'Gesamtbeitrag €' field is set to 100,00.

Dokument-Bearbeiter: Notizblock

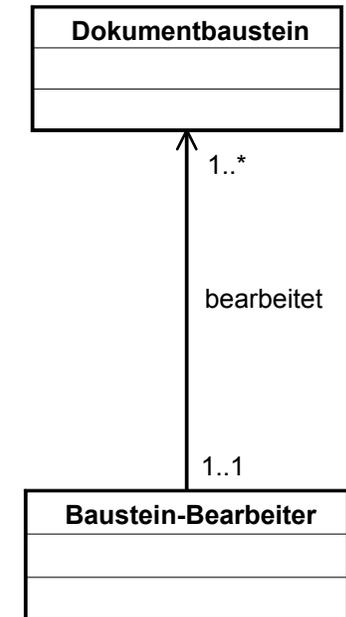
The screenshot displays a software interface for document management. At the top, there is a navigation bar with tabs for 'Vorgang', 'Antrag', 'Brief', 'Polisierung', 'Rückfrage', and 'Wiedervorlage'. Below this, a menu contains various actions like 'Antrag ablegen', 'Antrag komplett prüfen', and 'Antrag polizieren'. The main area is titled 'Entwicklung' and shows a table with columns for 'Betreff', 'Datum', and 'Autor'. A table entry is visible: 'Bitte Berufsgruppe prüfen', '15.09.2006 09:42', 'admin'. A 'Neue Notiz' dialog box is open, allowing the user to create a new note with fields for 'Autor', 'Datum', 'Betreff', and 'Text'. The 'Notizblock' is highlighted in the left sidebar, and a black arrow points to it. A red circle is drawn around the main content area.

Betreff	Datum	Autor
Bitte Berufsgruppe prüfen	15.09.2006 09:42	admin

Beispiel: Vorgangsbearbeitung -> Baustein-Bearbeiter

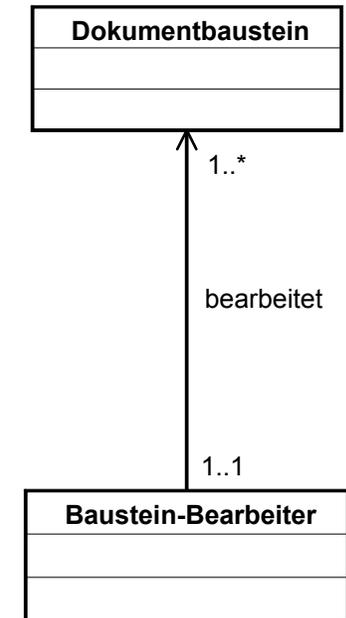
Gleiches Prinzip wie bei Dokument-Bearbeiter, eine Ebene tiefer:

- Baustein-Bearbeiter ist ein Java-Interface und ein Extension-Point
- Java-Klassen, die Baustein-Bearbeiter implementieren, enthalten den vollständigen Code, um Bausteine eines Typs darstellen und editieren zu können
- Insbesondere gibt es Methoden zum Übernehmen der UI-Felder in die Baustein-Properties und umgekehrt
- Jede Extension, die sich hier anmeldet, deklariert, für welchen Typ von Baustein sie zuständig ist
- Die Zuordnung von Baustein-Typen und ihren zuständigen Bearbeitern erfolgt ausschließlich über die Konfiguration (plugin.xml)



Beispiel: Vorgangsbearbeitung -> Baustein-Bearbeiter

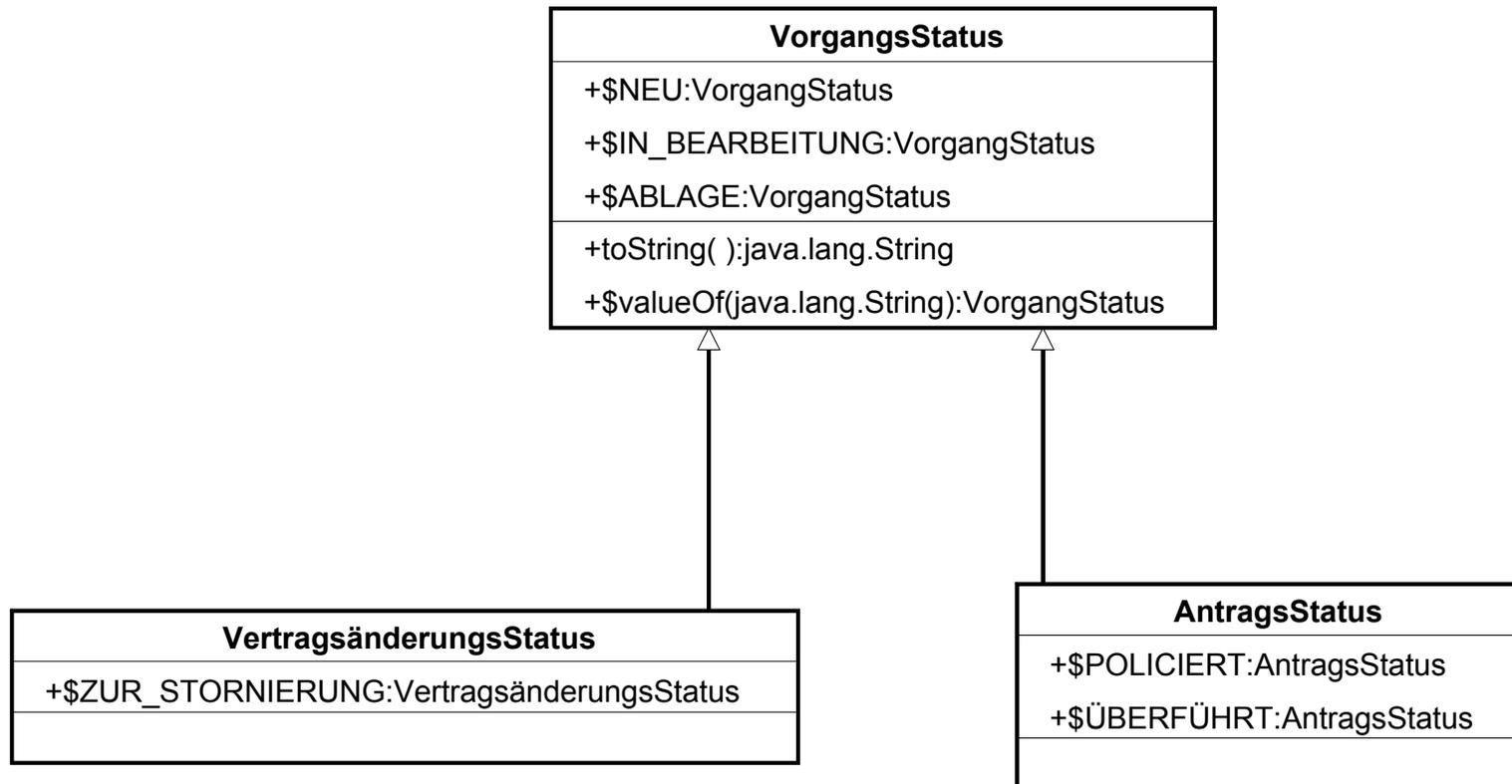
- + Dokumente lassen sich flexibel aus Bausteinen zusammensetzen (macht natürlich nur Sinn, wenn das Dokument selbst so eine Struktur nahe legt)
- + UIs lassen sich flexibel hinzufügen
- + Wenn man einen bestimmten Bausteintyp in einem weiteren Dokument verwendet, hat der automatisch schon einen passenden Bearbeiter



Beispiel: Initialisierung erweiterbarer Wertetypen

Erweiterbare Wertetypen sind keine Java-Enums!

Prinzip: Sobald die abgeleiteten Klassen geladen und initialisiert werden, registrieren sich ihre statischen Member **automatisch** als Instanzen ihrer Oberklasse.



Beispiel: Initialisierung erweiterbarer Wertetypen

Speicherung in der Datenbank über String-Mapping.

Der Hinweg über `toString`:

```
VorgangStatus status = vorgangsmappe.getStatus();  
VorgangsmappenBean bean = new VorgangsmappenBean();  
bean.setStatus(status.toString());  
persistenceManager.makePersistent(bean);
```

Beispiel: Initialisierung erweiterbarer Wertetypen

Speicherung in der Datenbank über String-Mapping.

Der Rückweg über `valueOf`:

```
VorgangsmappenBean bean =
```

```
    persistenceManager.getObjectById(vorgangsId);
```

```
Vorgangsmappe mappe = new Vorgangsmappe();
```

```
mappe.setStatus(VorgangsStatus.valueOf(bean.getStatus()));
```

Beispiel: Initialisierung erweiterbarer Wertetypen

Das Problem: Woher soll `VorgangsStatus.valueOf` etwas von den Konstanten in den abgeleiteten Klassen wissen?

Zur Erinnerung: sobald die abgeleiteten Klassen geladen und initialisiert werden, registrieren sich ihre statischen Member automatisch als Instanzen der Oberklasse `VorgangsStatus`.

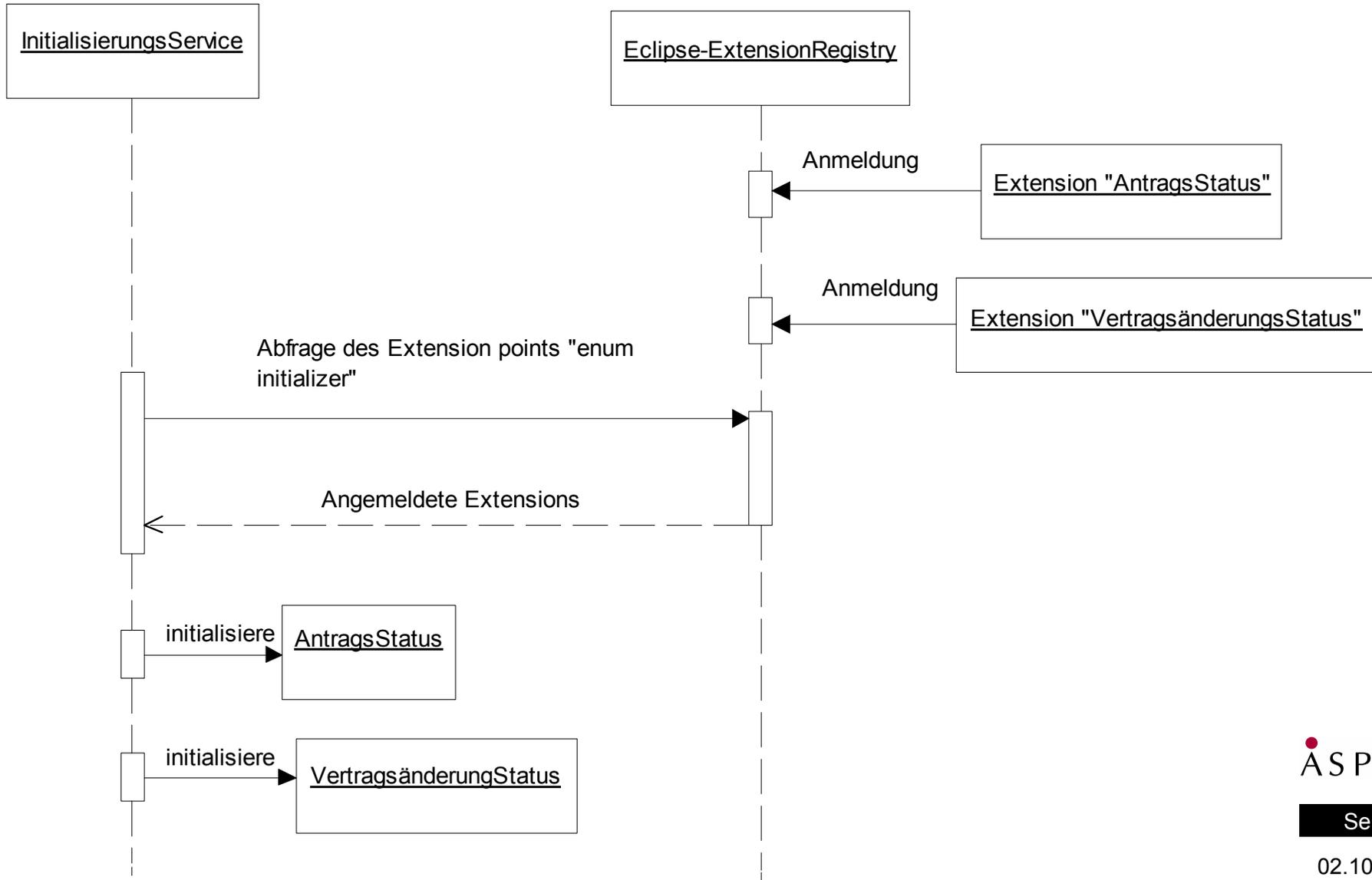
Wir benötigen also eine verlässliche Initialisierung der Klassen erweiterbarer Wertetypen zum Programmstart!

Die Lösung: Ein Initialisierungs-Service und ein Extension point „enum initializer“.

Die Klassen melden sich über diesen Extension Point selber zur Initialisierung an.

Die Applikation muss nur noch beim Programmstart den Service aufrufen, der dann alle angemeldeten Klassen initialisiert.

Beispiel: Initialisierung erweiterbarer Wertetypen



Wie zeigt man Prüfergebnisse und andere Hinweise an?

- Das Marker-Konzept:
 - Jedes Dokument in einer Mappe kann so genannte „Marker“ liefern
 - Ein Marker kann einen Hinweis geben oder Problemstellen aufzeigen
 - Ein Marker enthält einen Text, der den Hinweis oder das Problem beschreibt
 - Ein Marker kann zu einer thematischen Kategorie gehören
- Marker können generisch am UI dargestellt werden
 - Marker erscheinen automatisch auf dem übergreifenden Hinweisblatt der Vorgangsmappe
 - Marker erscheinen zusätzlich automatisch an der entsprechenden Problemstelle im Bearbeiter
 - Marker werden per HTML dargestellt
- Woher die Marker kommen, hängt von der konkreten Konstruktion ab
 - Prüfungen hängen Marker als Prüfergebnisse an Dokumente an
 - Dokumente können auch je nach ihrem Zustand selbst Marker liefern

Beispiel: Marker am UI

Hinweisblatt

Es liegen 4 schwerwiegende Probleme , 3 Probleme und 3 Hinweise vor.

ACHTUNG !!!

- Antrag wurde mit der allgemeinen Maske erfasst. Es fehlen wichtige Daten.

Allgemeine Hinweise

- Die Antragsdaten wurden noch nicht in die Antragsstatistik überführt. Bitte tragen Sie die fehlenden Daten nach (Beitragssumme,HDI-OB).
- Die Antragsdaten wurden noch nicht in das Antragstracking übernommen.

AVS

- Verbindung zum AVSService kann nicht hergestellt werden. >>>

Der Schnellerfasser hat folgende Probleme gemeldet:

- Es sind Anlagen zu diesem Antrag vorhanden **ok**

Fehlende Angaben

- Die Legitimation ist unvollständig. Die fehlenden Angaben vom VN nachfordern. **ok** >>>

Prüfungen

- Die Gesundheitserklärung ist nicht mehr gültig, da mehr als 6 Monate zwischen Antragsdatum und Versicherungsbeginn liegen **ok** >>>
- Falsche Beitragssumme für Hauptversicherung [ersetze durch 24.000](#) >>>
- Die Summe der Fondsanteile ergibt nicht 100% >>>

Vorvertrag

- Es wurden folgende Vorverträge für [redacted] gefunden

Mandant: Aspecta		urspr. Todesfallschutz	urspr. Monatsprämie
Versicherungsscheinr.			
315776351		irrelevant	50,00 €
aktuelle		irrelevant	100,00 €
350117672		irrelevant	-

Beispiel: Marker-Fixes

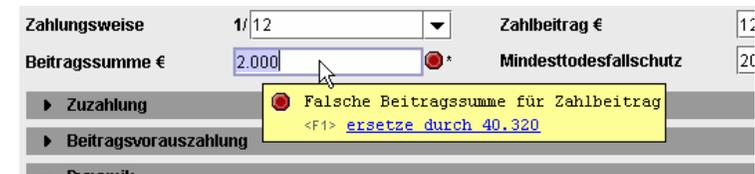
Bei einigen Problemen kann das System direkt helfen:

- Per Extension-Point können so genannte Marker-Fixes hinzugesteckt werden:
 - Orientiert sich an den Quick-Fixes in der Eclipse-IDE
 - Ein Marker-Fix bietet eine Lösung für ein Problem an
 - Wird am UI dargestellt und kann per Hyperlink aktiviert werden
 - Das System ermittelt zu allen Markern passende Marker-Fixes
 - Für ein Problem können auch mehrere unterschiedliche Marker-Fixes existieren

Auf dem Hinweisblatt:



Im Antrags-Bearbeiter:



Generelle Frage:

- Was flexibilisiert man per Konfiguration (aus der DB) und was per Extension-Point inkl. Code?

Unser Weg: Kein komplettes Meta-Modell

- Neue Tarife, Annahme-Grenzen und ähnliches wird per Datenbank konfiguriert
- Neue Produkte können per Extension hinzugefügt werden

Neue Produkte per Extensions:

- Produkt-Definition als Business Object (Dokument)
- Produkt-Persistenz als Service (Dokument-Verwalter)
- Produkt-spezifisches UI als UI-Komponenten (Dokument-Bearbeiter, etc.)
- Neue Prüfungen als Services
- Passende Marker-Fixes
- Spezielle Actions

Beispiel: Verteilte Services

Ausgangssituation:

- Services sind als POJOs realisiert

Die Idee:

- Services mittels Spring-Remoting remote-fähig machen und in das System einklinken

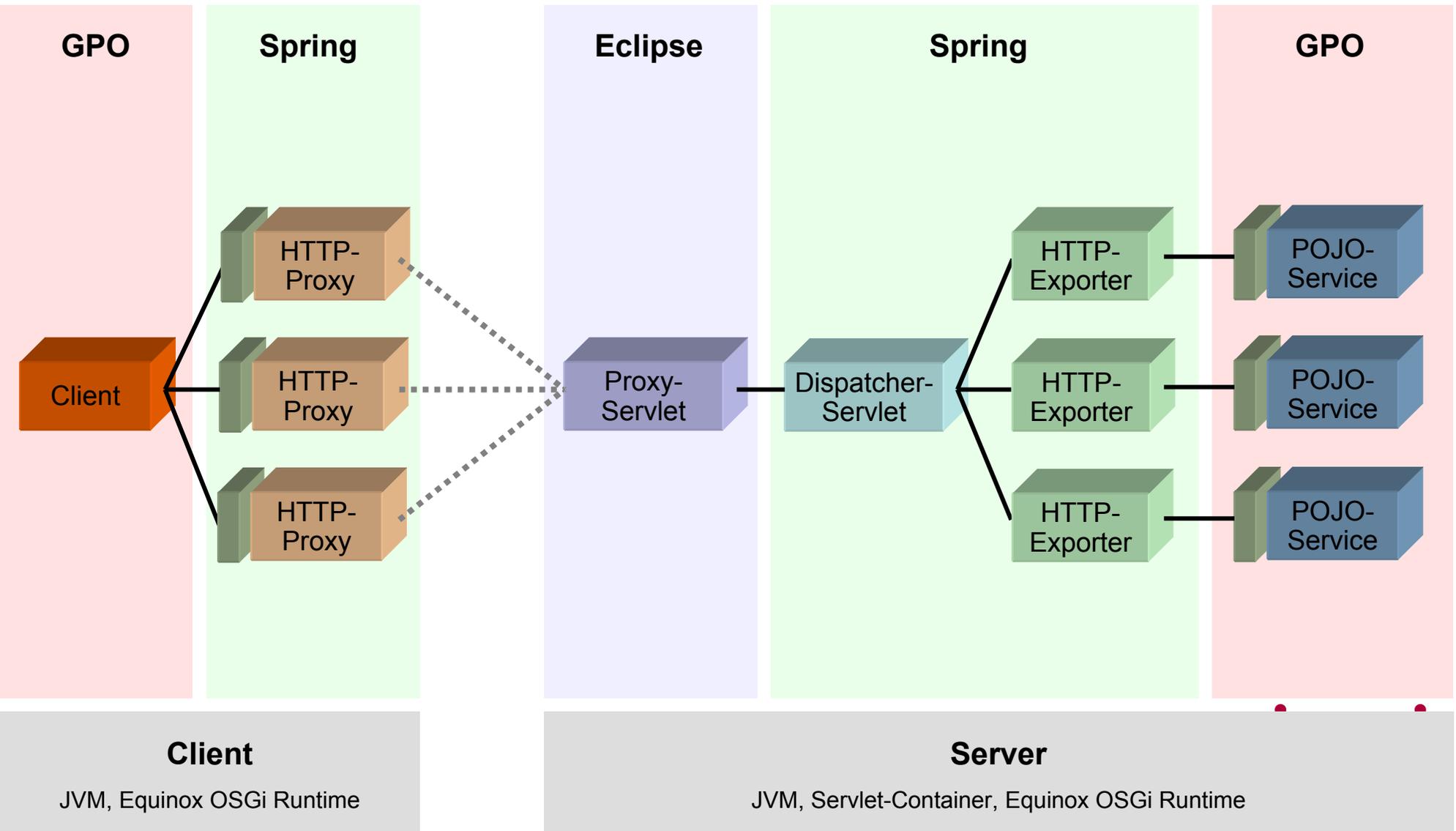
Das Spring Framework:

- Spring wird als ein Plugin dem System hinzugefügt
- Mit der Spring-OSGi-Integration (in Spring 2.1) können alle Plugins das Spring Framework nutzen

Verteilte Services:

- Werden per Spring-Definition remote-fähig gemacht
- Und über den Service-Extension-Point wird beispielsweise der Remote-Proxy angezogen

Verteilte Services mit Spring und Eclipse



Lessons Learned

Wir fassen zusammen:

Was haben wir gelernt?

Was würden wir wieder so machen?

Was würden wir anders machen?

Was sind die Schwierigkeiten?

- Getrennte Module können schwierig werden, wenn...
 - Bibliotheken Java-Reflection verwenden (müssen), z.B. Persistenz-Frameworks
 - Bibliotheken implizit dynamische Abhängigkeiten erzeugen (beispielsweise Hibernate mit generierten Proxies)
 - Bibliotheken merkwürdiges Classloading machen
 - Libraries eigene Versionen von Open-Source-Bibliotheken mitbringen
- *Manchmal ist das eigentliche Problem schwer zu identifizieren*
- **Aber:** Es gibt immer eine Lösung!
 - Buddy-Loading
 - Wrapper
- Eclipse 3.2 ist da um vieles freundlicher geworden... ☺

Lessons Learned: Stabile Plattform

Wie ist unser Stand?

- + Wir haben getrennte Builds für die Plattform und die Säulen, aber:
- Eine stabile Plattform ergibt sich nicht von allein, nur weil man getrennte Builds hat

Was haben wir gelernt? Was würden wir anders machen?

- ! Mit einer kleinen, schlanken Plattform starten
- ! Diese kleine Plattform schrittweise erweitern
- ! Explizit API-Design als Aufgabe verstehen und entsprechende Reviews bei Aufnahme in die Plattform durchführen
- ! Getrennte Teams unterstützen den Plattform-Gedanken, können aber auch hinderlich sein

Erweiterbarkeit auf Basis von Extension-Points:

- + Der Extension-Point-Mechanismus ist einfach zu verstehen und einfach zu verwenden (kann selbst ohne OSGi-Runtime genutzt werden)
- + Erlaubt es, sehr flexible Architekturen zu realisieren
- + Erlaubt relativ einfach „späte Erzeugung“ und dadurch gute Skalierbarkeit

- ! Bei variablen Teilen immer überlegen, ob die Flexibilität nicht durch einen Extension-Point herausgezogen werden sollte
- ! „Always have a client“ (YAGNI, keine EP auf Vorrat)

- Zu viele Extension-Points erschweren die Übersicht, Dokumentation notwendig

Was hilft, mit „published APIs“ umzugehen?

- Interfaces:
 - + Definieren die eigentlichen Begriffe und Services des Plugins
 - + Sind für Extension-Points elementar wichtig
 - + „Programming to an interface, not an implementation“
- Abstrakte Klassen:
 - + Können als Default-Implementationen von Interfaces helfen, API-Änderungen abzufedern

Eine immer wiederkehrende Frage:

Wie schneidet man Plugins?

- Unsere Plugins (insgesamt ~260) sind relativ klein geschnitten
 - + Feine Granularität, angebundene Systeme beispielsweise separat in einzelnen Plugins gekapselt
 - + Dadurch einzelne Funktionsbereiche gut voneinander getrennt und separat nutzbar
 - Viele Projekte, viele Abhängigkeiten
 - ! Wir würden in Zukunft zu weniger Plugins tendieren

- Im Eclipse-SDK sind die Plugins recht groß geschnitten
 - + Wenige Projekte, bessere Übersicht
 - Größere Granularität, daher viele Teile nur „als Ganzes“ verwendbar

Verpackt man Libraries in eigene Plugins oder direkt in das Plugin, welches die Library verwendet?

- In jeweils eigene Plugins:
 - + Klar definierte Abhängigkeiten
 - + Auch Versions-Abhängigkeiten können dediziert deklariert und damit behandelt werden
 - Es werden immer mehr Plugins
- In das verwendende Plugin:
 - + Weniger Projekte, einfaches Handling, auch bei Updates der Library
 - Teilweise die gleiche Library dutzendfach im Workspace
 - Man muss sehr vorsichtig mit den Exports sein, sonst gibt es Versions-Konflikte und ClassCast-Exceptions

Lessons Learned: Persistenz mit JDO

Persistenz ist überwiegend mit JDO realisiert
(einer kommerziellen Implementation)

Was haben wir gelernt?

- ! Verwende niemals ein Framework entgegen seines intendierten Einsatz-Szenarios
 - ! JDO ist deutlich langsamer als direkter JDBC-Zugriff (Faktor 5-10)
 - ! Entscheide dich nie für ein so grundlegendes technisches Element aufgrund von politischen oder strategischen Überlegungen
 - ! Implementiere immer einen Real-World-Prototypen mit der gewählten Technologie, bevor sie in der Breite eingesetzt wird, auch unter dem Performance-Gesichtspunkt
-
- Closed-Source-Implementation erweist sich immer wieder als hinderlich im Projektalltag
 - Enhancement des Bytecodes beim Kompilieren verzögert den Auto-Build enorm, kein flüssiges Arbeiten in Eclipse mehr
-
- Spätes Ersetzen des Persistenz-Frameworks kann aufwendig sein, selbst wenn der Code gut gekapselt ist

Lessons Learned: Refactoring

Je größer das System wird und je länger es weiterentwickelt wird (oder werden soll), desto wichtiger sind Refactorings!

Unsere Erfahrungen:

- ! Refactorings müssen kontinuierlich durchgeführt werden
- ! Refactorings müssen dann durchgeführt werden, wenn der Code-Smell bemerkt wird
- ! Aufgeschobene Refactorings führen zu immer größer werdenden Aufwänden

- Published-APIs machen Refactorings teilweise schwieriger
- Die Sicht auf Softwareentwicklung als Projekt erweist sich gerne mal als hinderlich in diesem Zusammenhang:
 - Das Projektziel selbst ist in der Regel **nicht**, ein wartbares, gut strukturiertes System zu implementieren
 - Produkt-Sicht oftmals hilfreicher

- ? Immer wieder offener Punkt:
 - Wie plane ich Refactorings ein? Verstecke ich Refactoring-Aufwände in den normalen Features?

Es gibt häufig Diskussionen darüber, wie hilfreich Frameworks wirklich sind.

- Unsere Erfahrungen:
 - + Wir haben gute Erfahrungen mit kleinen und leichtgewichtigen Frameworks gemacht.
 - z. B. für Dokument-Bausteine und Baustein-Bearbeiter
 - + Lassen sich einfach implementieren
 - + Erlauben schnelle Entwicklung in die Breite
 - Aber: API-Problematik kommt sehr schnell ins Spiel

Automatisierte Tests sind überlebensnotwendig, vor allem, je größer ein System wird und je länger es entwickelt wird

- Unsere Erfahrungen:
 - ! JUnit-Tests sind gut, müssen aber schnell ausführbar sein und dürfen den eigenen Testdaten-Bestand nicht zerstören
 - ! Man muss frühzeitig mit automatisierten Integrations- bzw. Akzeptanztests (beispielsweise mit FIT) beginnen
 - Später sind solche Tests schwer und aufwendig zu implementieren
 - ! Testplan bzw. Drehbuch für manuelle Tests durch die Anwender notwendig
 - ! Für die Entwicklung ist ein hoher Grad an Automatisierung zwingend erforderlich
 - Der manuelle Test-Aufwand erstickt sonst die kurzen Release-Zyklen
 - Die Qualität des Systems sinkt kontinuierlich
 - ! Und nicht zuletzt: Alle Entwickler müssen an einem Strang ziehen

Kann ich die UIs „einfach“ so runterprogrammieren?

- **Achtung!!!**

- ! Man muss sehr genau darauf achten, was man im UI-Thread macht und was nicht
 - Siehe auch diverse Bücher zu Swing
- ! Eine Richtlinie: Möglichst wenig im UI-Thread tun
 - Damit macht das UI einen „flüssigen“ Eindruck
- ! Die Folge: Man muss sehr sauber mit dem Swing-Thread-Modell umgehen und auf Multithreading achten

- Es hilft, wenn man passende Richtlinien und/oder passende kleine Frameworks zur Hand hat, die einem die wesentlichen UI-Threading-Aufgaben abnehmen

Wie und wo implementiert man fachliche Prüfungen?

- Unser erster Ansatz: Sehr starke Modularisierung
 - Führte zu Schwierigkeiten bei abhängigen Prüfungen und Prüf-Reihenfolgen
 - Viele Dinge wurden doppelt und dreifach geprüft (kostet Performance und erschwert die Wartung)
- Marker haben sich bewährt!
 - + Prüf-Ergebnisse als Marker sind sehr flexibel

Fragen ???

Fragen jederzeit gerne!

Folien online auf der Seite des Arbeitskreises oder unter:

<http://www.it-agile.de/newsfeed.html>

Ivo Eitner: ieitner@aspecta.com
Markus Herzog: maherzog@aspecta.com
Gernot Neppert: geneppert@aspecta.com
Martin Lippert: martin.lippert@akquinet.de

