Cockpit-basierte Management-Systeme

Robert Neumann, Fritz Zbrog, Reiner Dumke

Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik,
AG Softwaretechnik, Postfach 4120, 39106 Magdeburg
[robert.neumann,fritz.zbrog,reiner.dumke]@ovgu.de

Zusammenfassung

Der folgende Beitrag zeigt die verschiedenen technologisch-basierten Formen einer so genannten Cockpit-Anwendung für die verschiedensten Bereiche der Software-Entwicklung, -Wartung und dem (Qualitäts-) Management. Ausgehend von den allemeinen Grundlagen und Intentionen einer Cockpit-Anwendung wird eine dynamische Software-Infrastruktur vorgestellt, die insbesondere die operativen Informationsflüsse, deren mögliche Aggregationen, aber auch Präsentationen in flexibler Weise unterstützt und (für zunächst drei unabhängig ansteuerbare Bildschirme) implementiert.

1. Einleitung

Die Informationstechnologie (IT) befindet sich in einem komplexen, dynamischen Umfeld. Analysen haben gezeigt, dass sich die IT parallel zur wachsenden Globalisierung der Unternehmen entwickeln wird. Außerdem zeigen Trends, dass die Unternehmen zunehmenden Ansprüchen und Herausforderungen gewachsen sein müssen. Dazu gehört, dass IT-Unternehmen sowohl dem wachsenden Kostendruck als auch dem steigenden Anspruch an Qualität und Flexibilität nachkommen müssen. Um diesen Ansprüchen gerecht zu werden, benötigt jedes IT-Unternehmen ein sehr gut funktionierendes Management. Für ein erfolgreiches Software-Projekt sind gute Projektplanung, -beobachtung und -steuerung die wichtigsten Grundvoraussetzungen. Das Management von Software-Projekten verbindet unterschiedliche Geschäftsbereiche miteinander. Prozess- und Produktbeobachtungen sollten daher den jeweiligen Perspektiven angepasst präsentiert werden. Hierbei kann ein so genanntes „Software-Projektkontrollzentrum“ für das Management als eine kompakte Informationsquelle bezüglich des Projektstandes hilfreich sein.

In der aktuellen Fachliteratur hat sich dafür auch der Begriff des Leitstandes oder Cockpits etabliert. Diese sollen dem Manager die Möglichkeit bieten, Auswertungen des Projekt-Controllings an die jeweilige Fragestellung oder den individuellen Interessen des Betrachters anzupassen und relevante Steuerungsinformationen übersichtlich darzustellen. Die Idee des Cockpits stammt aus der Luft- und Raumfahrt: Der Flug stellt einen Prozess dar, der mit einem Entwicklungsprozess in der IT vergleichbar ist. Am Prozess sind Mitarbeiter aus unterschiedlichsten Bereichen beteiligt (Pilot, Flugsicherung, Passagier usw.). Jeder wünscht oder benötigt verschiedene Informationen über den Flug (Prozess) oder das Flugzeug (Produkt). Der Passagier beispielsweise erhält Informationen über den Fortschritt des Fluges. Der Pilot hingegen benötigt alle Informationen über die Technik, Funktionstüchtigkeit und Leistung des Flugzeuges. Die Flugsicherung wird mit Informationen über den Luftraum und allen darin befindlichen Flugzeugen versorgt, um diese zu koordinieren.

Abbildung 1: Leitstand für die Flugsicherung

In der Software-Entwicklung wird erst durch Cockpits die Messbarkeit des Projektstatus sowie die Erstellung von Trend-Analysen ermöglicht. Projektinformationen lassen sich transparent und standardisiert darstellen, die Bewertung und Einschätzung des Projektes wird vereinfacht und eine klare, einfache Berichterfassung möglich. Es sollen dabei keine neuen Informationswelten geschaffen werden, vielmehr werden bestehende Systeme (Kunde, Software-Entwicklung, Projektleitung usw.) vernetzt und in Zusammenhang gebracht. Mit Hilfe von Cockpits kann die Projektplanung realistischer gestaltet werden.

Aufwand und Kosten der Software-Entwicklung lassen sich auf ein Minimum reduzieren, da prozess- und produktrelevante Risiken schnell identifiziert werden können.

Abbildung 2: Cockpit-Vision

Um Software-Entwicklungsprozesse erfolgreich zu unterstützen gibt es bereits unterschiedliche Ansätze. Einerseits werden hierbei vor allem Tools bzw. Tool-Infrastrukturen verwendet, die eine Prozessplanung und -kontrolle auf der Grundlage projektbezogener (Mess-) Datenhaltung realisieren (z. B. Knowledge Plan, MS Project, Function Point Workbench, COCOMO II Tool, Rational kombiniert mit Metrics One, SLIM Tool usw.). Die dabei erfassten Daten sind im Allgemeinen Messdaten zur Produktqualität (zumeist auf der Grundlage so genannter Codemetriken) und projektbezogene Daten aus Aufwandsschätzungen und/oder Prozessplanungsdaten (Ressourcen, zeitlicher Ablauf, Einsatzbedingungen).

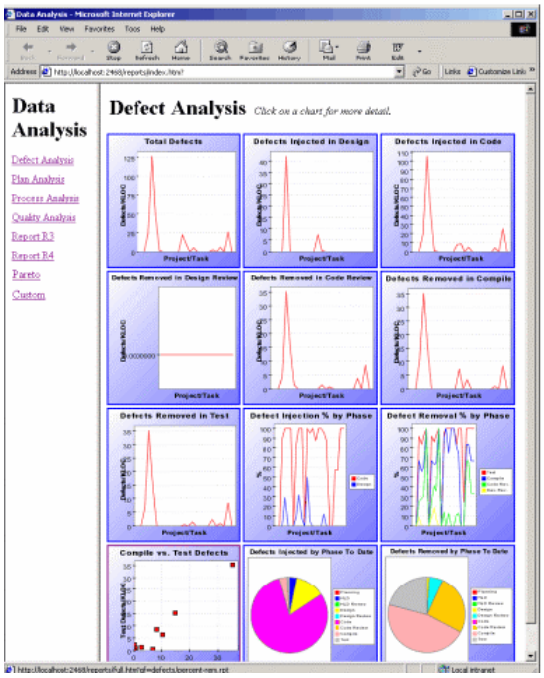

Abbildung 3: Firmenbeispiel zur Cockpit-Adaption

Andererseits werden insbesondere als Berichterstattung für das Firmenmanagement Projektdaten (zumeist manuell) aggregiert und aufbereitet. Abbildung 3 zeigt ein derartiges Beispiel aus dem Alcatel-Firmenbereich [Ebert 2007]. Neben diesen firmenbezogenen Lösungen gibt es auch Bestrebungen, Projekt-entscheidungen durch die Bereitstellung allgemeiner Erfahrungen zu unterstützen.

Für die allgemeine Struktur gilt beispielseise die in der folgenden Abbildung beschriebene Ebenen-Architektur für den Bereich des Projektmanagements [Ebert 2007].

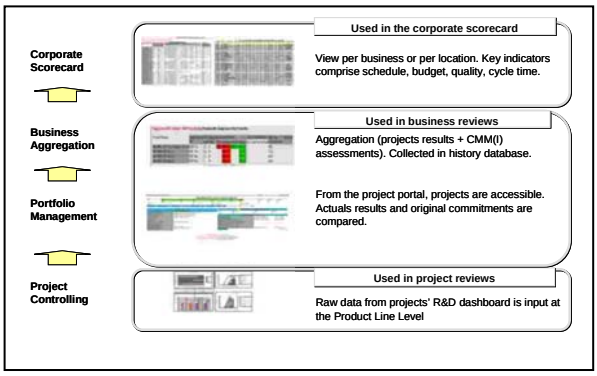

Abbildung 4: Cockpit-basierte Aggregationsformen von Projektmanagementdaten

2. Einfachste Cockpit-Lösungen

Ausgangspunkt für Cockpit-Lösungen ist es, die Möglichkeiten der Darstellung von Prozess- und Produktinformationen während des Software-Entwicklungsprozesses zu analysieren. Damit werden dem Software-Management alle relevanten Informationen gegeben, um ein Projekt optimal zu kontrollieren und steuern zu können. Für einen Projekt-Leitstand sollen die einzelnen Indikatoren auf separaten Bildschirmen mit unterschiedlichen Darstellungs-formen präsentiert werden. Die Messung, Auswertung und Darstellung der Metriken werden jeweils mit Hilfe unabhängiger Tools bzw. (Internet-) Diensten realisiert. Die einzelnen Cockpits sollen aktuelle und konsistente Informationen liefern, damit einen schnellen und übersichtlichen Zugang zu allen relevanten Ergebnissen ermöglichen und somit die Gesamtsituation klar und übersichtlich darstellen und somit Steuerungsbedarf leicht erkennbar machen.

Basierend auf den so ermöglichten Mess- und Analyseaktivitäten gibt es unterschiedliche Darstel-lungsvarianten integrierter Messwertdarstellungen. Abbildung 5 zeigt eine Möglichkeit für eine derartige Kombination hinsichtlich struktureller und zeitlicher Aktivitätenbeschreibungen.

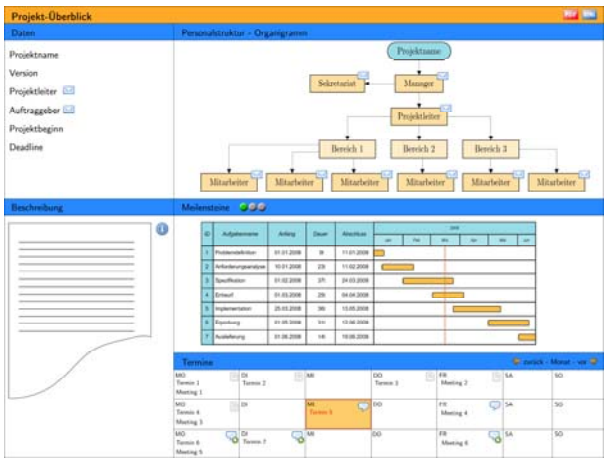

Abbildung 5: Entwurf einer Cockpit-Darstellung für die Projektübersicht

Mit Hilfe dieses Leitstandes können dann Projektkennzahlen aus den einzelnen Metriken aggregiert werden, um damit eine neuartige Projekt-Erfahrungsdatenbank aufzubauen. Diese kann dann die Analysemöglichkeiten und die Analysegenauigkeit signifikant erhöhen (siehe Abbildung 6).

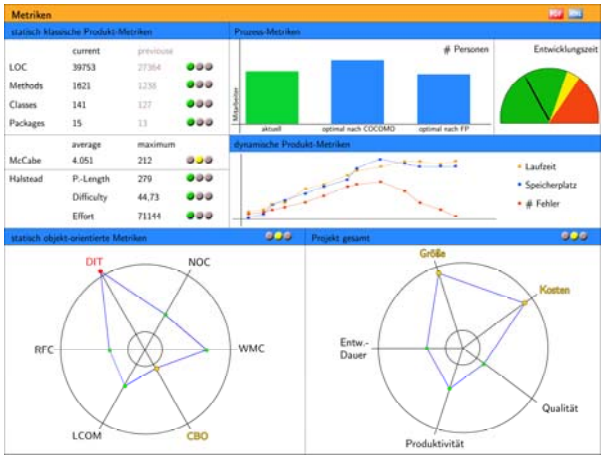

Abbildung 6: Entwurf für die detaillierte Darstellung von Metriken-Werten eines Produktes

Die Evaluierung der erreichten Ergebnisse für den intendierten Kontroll- oder gar Steuerungsprozess erfordert eine komplexe Form der zielgerichteten Auswertung bzw. Aggregation derartiger Auswertungsformen für den Managementbereich. Abbildung 7 deutet schließlich eine derartige Zusammenstellung für den unimttelbaren Projektmanagementbereich an und skizziert auch deren sinnvolle Verteilungsformen innerhalb eines Screens der Cockpit-Architekur.

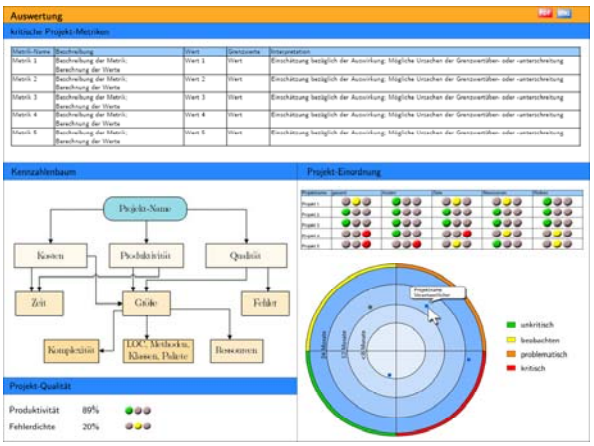

Abbildung 7: Entwurf für die Analyse einzelner Projektgebiete

Die Anwendung dieser Prinzipien wurden bereits in einer ersten prototypischen Lösung eines Cockpits zur Unterstützung von grundlegenden Projektmanagement-aktivitäten implementiert [Hansen 2008].

Abbildung 8: Cockpit-Beispiel für das Projetkmanagement

Dabei wurden folgende technologischen Infrastruk-turformen implementiert:

- Für die Ansteuerung des Bildschirms einerseits und des Multi-Screens andererseits wurden zwei Rechner verwendet.
- Die Applikationen sind jeweils spearate Desktop-Prozesse eines Rechners.

3. Der Magdeburger Cockpit-Ansatz

3.1 Architekturbezogene Grundlagen

Für das Magdeburger Cockpit-Framework gelten die folgenden Anforderungen:

1. Herausarbeitung der wesentlichen Grundsätze bei der Konstruktion von Cockpits in den verschiedensten IT-Bereichen,
2. Schaffung einer erweiterbaren, komponenten-basierten Cockpit-Architektur,
3. Bewertung verschiedener, technologischer Ansätze zur effizienten Umsetzung,
4. Orientierung (zunächst) auf eine messwerter-zeugende, -verarbeitende und –darstellende Prozessform,
5. Schaffung weiterer experimenteller Prototypen.

Für das eingegrenzte Ziel (in 4.) gilt beispielsweise das folgende Szenario:

> *Agenten- bzw. Sensorsoftware akquiriert die darzustellenden Informationen am Entstehungsort, woraufhin ein Transportsystem das Zusammenführen von Informationen (z.B. Datensammlungen zur Traffic-Verteilung, Belegungs- und Verbrauchsverteilung) am Präsentationsort organisiert. Nach entsprechender grafischer Aufbereitung wird die Darstellung der Informationen vorgenommen. Durch manuelle Interaktionen werden semiautomatische Sub- bzw. Assistenzsysteme ergänzt, wobei Prozess-Log-Daten sowie Transaktionskontrolldaten vom System erfasst werden. Im Anschluss erfolgt eine Rückverteilung der vom User eingegebenen Steuerdaten über das Transportsystem zu den entsprechenden Aktoren/Agenten. Der Informationskreislauf schließt sich.*

Die allgemeine Cockpit-Architektur lautet daher wie folgt.

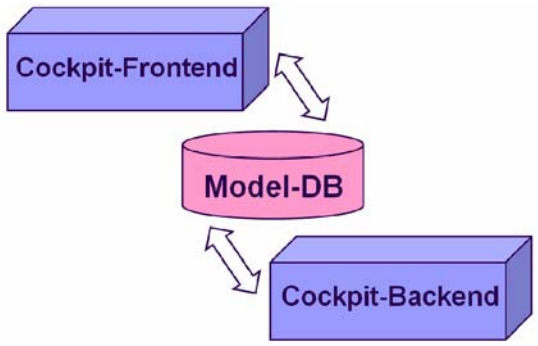

Abbildung 9: Allgeimene Cockpit-Architektur

Die einzelnen Frotend-Komponenten zeigt die folgende Abbildung.

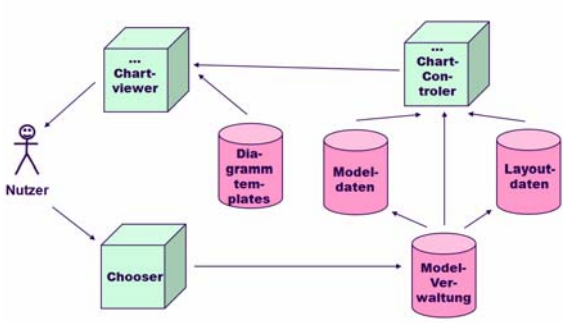

Abbildung 10: Cockpit-Frontend-Komponenten

Die entsprechenden Komponenten des Cockpit-Backend zeigt schließlich die folgende Abbildung.

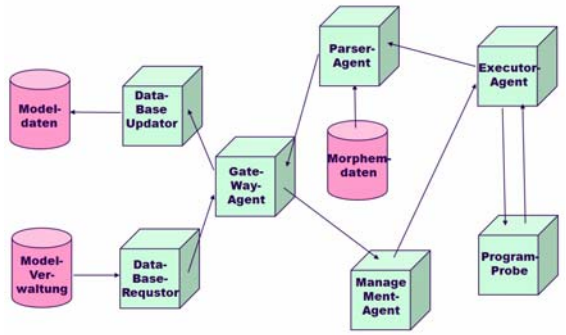

Abbildung 11: Cockpit-Backend-Komponenten

3.2 Realisierungsvarianten der Cockpit-Schichtenarchitektur

Aus dem Informationskreislauf des oben genannten Szenarios lässt sich die folgende allgemeine Schichtenstruktur für Cockpits ableiten.

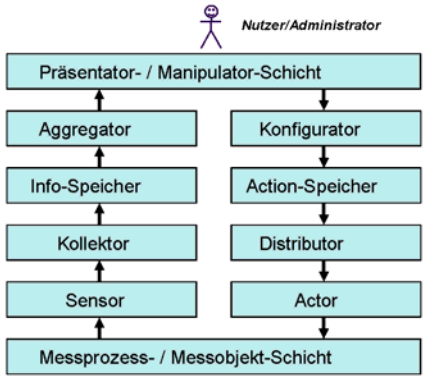

Abbildung 12: Cockpit-Schichtenstruktur

Im Folgenden werden zunächst Umsetzungsvarianten zu den einzelnen Schichten erläutert.

Darstellungsschicht

Durch Nutzung von großen Monitoren oder von Monitorgruppen zur Darstellung paralleler Informationsströme zu einem zusammengehörigen Sachverhalt entsteht intuitiv ein Software-Cockpit. Beispielsweise könnte ein Programmierer seine Entwicklungsumgebung auf einem Schirm, die Syntaxbeschreibung der von ihm verwendeten Programmiersprache auf einem zweiten und ein Web-Browser-basiertes Programmier-Tutorial auf einem dritten vor sich haben. Dahingehende Ansätze zur Organisation mehrerer Betriebssystem-Desktops durch so genannte Desktop-Manager tragen dieser Idee nur bedingt Rechnung. Spontane Cockpits, die entstehen, wenn User durch Selbstorganisation von Fenstern auf dem eigenen Desktop versuchen, mehre parallele Informationsströme gleichzeitig zu visualisieren werden durch eine sogenannte „Desktop Application Integration API“ unterstützt. Offene bzw. konfigurierbare Cockpits repräsentieren ein Rahmen-Framework zur Integration von Anwendungen in einen Subrahmen. Geschlossene Frameworks hingegen können nur vom Vertreiber modifiziert und erweitert werden. Offene Cockpits sind jedoch eher selten, meist werden die Informations-darstellung von einer Rahmenapplikation (z.B. Web-Browser, Application-Frame) gehostet.

Ein Web-Browser hat den Vorteil, plattformunabhängig und für die meisten Firewalls durchlässig zu sein. In Web-Browsern kann die Visualisierung von Informationen durch komplexe Applets, Portlets, AJAX-Scriptlets, GUI-Widgets oder Mashups umgesetzt werden. Entscheidet man sich für eine solche Variante, sind bestimmte Infrastrukturkomponenten der tieferen Schichten bereits festgelegt (Web-Server, JSP/Java-Container, CMS, Portalserver, Mapserver). Eine Cockpit-Applikation kann auch selbständig und plattformneutral sein (siehe Eclipse). In einem solchen Fall werden die parallelen Darstellungen durch Sub-GUIs von Teil- oder Plugin-Applikationen erzeugt.

Für das Cockpit-Konzept ist vor allem wichtig, dass Prozessetappen oder situationsbezogen eine parallele Darstellung relevanter Daten gezeigt wird. Das Eclipse-Framwork z.B. zeigt je nach Entwicklungs-phase verschiedene Fenster zugehöriger Tools. Tendenzen, die Offenheit von Cockpits zu unterstützen (im Sinne der Anpassungsfähigkeit), bestehen darin, eigene Workflows durch zusammenstellbare Darstel-lungsgruppen, kommunizierende Applikationen (siehe OLE-Verbunde) durch Story-Generatoren, Chart-Template-Generatoren, Desktop-Integratoren, Portlet- und Mashup-Konfiguratoren als Zusatz-Tools zu ergänzen.

Aggregator- und Konfiguratorschicht

Die Realisierung von Aggregatoren umfasst ein Spektrum von Shared-Memory-basierten, direkten Speicheraggragationen. Dabei hat ein Collector zuvor Daten in den Shared-Memory geladen und ein Aggregator liest diesen dann aus. Für stark Performance-abhängige Anwendungen könnte man sich auch die Eigenschaften moderner Grafikhardware zu Nutzen machen und den Shared-Memory à la GPGPU[1] auf den Grafikspeicher verlegen.

Die Darstellung selbst kann über eigenständige Komponenten bis zu einem modularen Multi-MVC-Konzept mit steuerbarer Kontrollfunktion zur Modellauswahl und Template-basierten Chart-Views erfolgen. Die Konfiguratoren nehmen manuelle Interaktionen vom User entgegen und lösen gegebenenfalls eine (Remote-) Reaktion aus bzw. stoßen einen parametrisierten Handlungsvorgang an, der entsprechend zeitlich und aktivitätsmäßig untersetzt wird.

Informations- und Aktionsspeicherschicht

Informationsspeicher, meist in Form von relationalen Datenbanken, sind bei komplexen Systemen zur funktionalen und zeitlichen Entkopplung der Vielzahl von parallel auszuführenden Präsentations- und Akquisetransaktionen unerlässlich. Neben der Darstellung revisionsfester Log-Einträge (ähnlich einer Flugzeug-Blackbox) haben Aktionsspeicher die Aufgabe der Aufbewahrung flexibler, austauschbarer bzw. modifizierbarer Ablauf- und Handlungspläne (ähnlich der Speicher von SPS), um entsprechende Verarbeitungen auslösen zu können.

Kollektions- und Distributorschicht

Kollektoren sind bestimmten Präsentationen zugeordnete, programmierbare Einheiten. Im passiven Modus arbeiten sie wie Event-Listener, die gleichzeitig das Modell der MVC-Architektur aktualisieren und damit für den Präsentator (Controller) eine Notifikation über ein Update erzeugen können – sie arbeiten so im Wesentlichen asynchron. Im aktiven Modus wird entsprechend der Akquisestrategie eine bestimmte Gruppe von verteilten Objekten mit einer vorgegebenen Frequenz abgefragt und die Ergebnisse für den Modellspeicher aufbereitet oder direkt weitergegeben.

Den von interaktiven Elementen der Präsentatoren eingetragenen Konfiguratoranweisungen entsprechen einzelne oder kompakte Anweisungen im Workflow-Stil. Distributoren übernehmen diese direkt und leiten sie an den Aktor weiter oder entnehmen die Aktionen entsprechend einer bestimmten Strategie (einmalig oder mehrmalig mit entsprechender Frequenz) dem

Speicher. Weiterhin übernehmen sie die richtige zeitliche und örtliche Verteilung sowie deren Ausführungsbestätigung.

---

[1] General Purpose Graphcis Processing Unit

Transportschicht

Denkbar sind hier IP, Industrie-IP oder datagrammbasierte Verbindungstechniken. In lokalen Systemen genügen meist Shared-Memory oder Message-Queue-Techniken. Für ein Cockpit speziell für Management von Computernetzwerken nutzt man meist auch Standard-Stacks und Protokolle (CMI, SMI, SNMP2), sowie deren MIB[2]-Datenbasis einschließlich der Agenten und den nutzerspezifischen Erweiterungen.

Eigene Lösungen lassen sich auch mit Unicast- oder Message-Queue-Systemen erreichen. Im Fall der Sockets wird die Kommunikationsverwaltung schwieriger und muss selbst implementiert werden. Die MQ-Systeme stellen einen zentralen neurologischen Punkt dar, der zudem noch einen Message-Router benötigt (ähnlich einem Integration Server in J2EE-Systemen bei EAI-Integration). Eine weitere flexible Lösung mit einigem Komfort stellt ein Ansatz mit Hilfe eines MAS (entsprechend des FIPA[3]-Standards) dar.

Sensor- und Aktorschicht

Bei technischen Prozessen oder Systemen sind Signalgeber und Fernwirktechnik mit entsprechender Treibersoftware Quelle und Ziel der Informationen. Die Netzwerke von Computern, Versorgungsunternehmen und Verteilungen besitzen meist Firmware-Komponenten (Programme zur Zustandsüberwachung und -manipulation) mit entsprechendem Interface, die ihre Integration in Cockpit-Systeme erleichtern. Eigenständige Software mit eigenem GUI lässt sich nur über Tunnellung (SSH, VPN) und Zuweisung von Präsentation-Frames integrieren.

Eine Vielzahl von Administrations-Tools oder Open-Source-Lösungen ist nicht speziell für Cockpits konzipiert; ihre Ausgaben müssen in einem Zwischenverarbeitungsschritt speziell aufgearbeitet werden (gegebenenfalls einen flexiblen Parser mit Morphemsteuerung).

Die obersten Schichten werden durch ein Java-basiertes Multi-MVC-Konzept mit ausgewählten Jfree-Charts realisiert. Der Controller übernimmt die Geräte-, Layout- und Data-Set-Zuordnung mit Hilfe einer MySQL-Datenbank, die gleichzeitig der Entkopplung von Frontend und Backend (obere und untere Schichten) dient. Das Backend-System besteht aus der MySQL-Verbindungssoftware, die über Gateway-Agenten mit einem JADE-basierten Multi-Agenten-System (MAS) zusammenwirkt. Die Agenten sind in Management-, Parse-, und Execution-Agenten auf verschiedenen Plattformen (Hosts) organisiert und nutzen entsprechend der Cockpit-Spezialisierung zur Zeit Open-Source-Sicherheits- und Management-Software als Sensor- bzw. Aktorschicht. Das gesamte System arbeitet in parallelen Tasks für die Teile Frontend, Datenbank, Gateway-Agent und die einzelnen Plattformen. Damit sind die Skalierbarkeit und eine gewisse Stabilität des Systems gesichert. In laufenden Arbeiten wird mit mobilen Agenten experimentiert. Abschließende Ergebnisse liegen gegenwärtig noch nicht vor.

3.3 Erste Implementierungsformen

Eine erste Implementation der Messwertverarbeitung und -darstellung wurde mit dem ***JADE-Framework*** prototypisch realisiert und für bereits vorhandene Messwertreihen präsentiert.
Darüber hinaus sind folgende Ergänzungen bzw. Erweiterungen vorgesehen:

- Ablösung der Agentengruppen durch mobile Agenten
- Schaffung eines Konfigurations- und Management-Interfaces
- Entwicklung von anpassbaren Verbund-Frames zur Individualisierung der Interaktion
- Parallelisieren des Frontends

4. Zusammenfassung

Wie gezeigt wurde, lassen sich Lösungen für eine Mess-Cockpit-Architektur finden, die an unterschiedliche Rollen, Vorgehensmodelle und Systemarchitekturen anpassbar und gleichzeitig hinreichend einfach sind. Cockpit-Architekturen könnten durch ein Framework iterativ an jeweilige spezielle Umgebungen anpassbar gemacht werden. Bei Messwert-Instrumentierung und –anzeige sollten in Ermangelung von Standards und Empfehlungen auch persönliche Präferenzen installierbar sein.

5. Literatur

[Dumke 2005] Dumke et al.: *An Agent-Based Measurement Infrastructure.* In: Abran et al.: Innovation in Software Mesurement, Shaker-Verlag 2005

[Ebert 2007] Ebert, C.; Dumke, R.: *Software Measurement.* SpringerVerlag, 2007

[Hansen 2008] Hansen, A.: *Konzeption und prototypische Realisierung eines Metriken-Cockpits für das Software-Projektmanagement.* Diplomarbeit, Universität Magdeburg, 2008

[Kunz 2008] Kunz et al.: *Ontology-Based Design of Architectures.* INMIC 2008, pp. 339-344

[2] Management Information Base
[3] Foundation for Intelligent Physical Agents