Einfache, aber konsistente Diagramme erstellen und verwenden

Bereich
Planung und Design

Aktivität
Lösungskonzept entwerfen

schnell durchführbar ★★☆
einfach durchführbar ★★☆
agil einsetzbar ★★☆

Ziele

- Ein grundlegendes Verständnis für das System bei Stakeholdern etablieren
- Eine Basis für die Diskussion im Team schaffen
- Festhalten und Verfügbarmachen der wichtigsten Konzepte

Motivation/Problemstellung

Agile Vorgehensweisen betonen die Wichtigkeit funktionierender Software gegenüber umfangreicher Dokumentation. Andererseits kann der vollständige Verzicht von Dokumentation zu negativen Auswirkungen führen. Die Vermittlung und Diskussion über grundlegende Funktionsweisen des Systems rein auf Basis von Gesprächen kann zeitaufwendig und schwierig sein, Umsetzungsdetails werden mit der Zeit vergessen, die Etablierung einer gemeinsamen Vision wird schwierig. Die Erstellung von einfachen Diagrammen als Diskussionsgrundlage bei der Erarbeitung von Konzepten und zur Dokumentation von Systemaspekten kann eine geeignete Maßnahme dagegen sein.

Kurzbeschreibung

Zur Beschreibung grundsätzlicher Aspekte des Systems oder spezifischer Konzepte können Entwickler Diagramme erstellen. Dies ist sowohl während der Erarbeitung von Lösungskonzepten als auch im Nachgang zur weiteren Verwendung während der Umsetzung sinnvoll. Damit Diagramme einen Mehrwert liefern, sollten Sie so gestaltet sein, dass sie den Konsumenten effizientes Arbeiten erlauben. Dafür sollten die Diagramme einen geringen Grad an Komplexität aufweisen und möglichst konsistent sein, das heißt grundsätzliche Prinzipien der Diagrammgestaltung sollten berücksichtigt und einheitliche Diagrammtypen und Notationen verwendet werden.

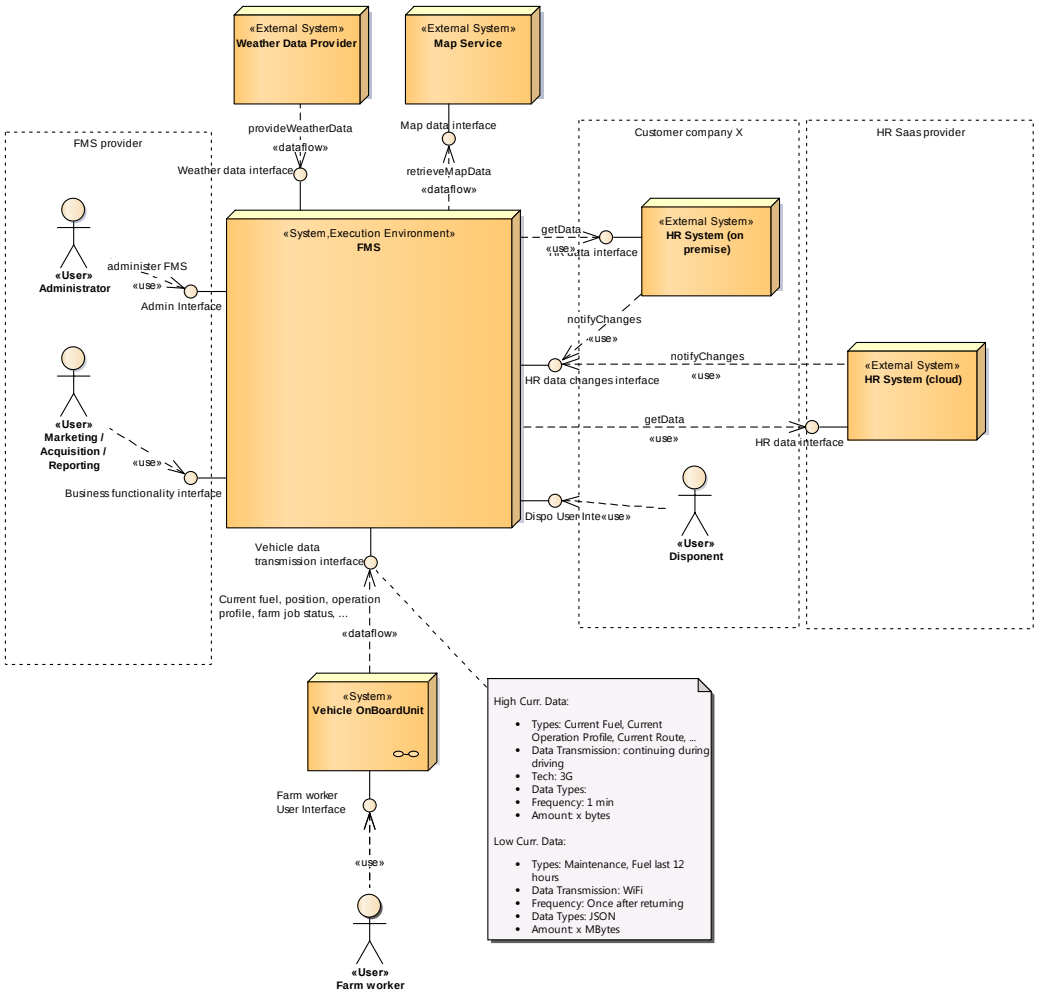

Abbildung 1:Beispiel Architekturdiagramm

Input

- Lösungsideen für den Systemaufbau oder das zu beschreibende Konzept

Output

- Einfache, aber konsistente Diagramme, die die allgemeine Funktionsweise des Systems oder konkrete Konzepte illustrieren, dokumentieren und die Basis für Diskussionen zwischen Stakeholdern bilden.

Rahmenbedingungen

Ausführender
Jeder Entwickler

Werkzeuge, Hilfsmittel
Whiteboard und/oder Stift und Papier und/oder Modellierungssoftware

Vorkenntnisse/Erfahrungen
Kenntnisse der UML sind von Vorteil

Ort/Umgebung
Beliebig

Weitere Teilnehmer
Weitere Mitglieder des Entwicklungsteams

Voraussichtliche Dauer
Wenige Minuten bis zu einer Stunde

Vorgehensweise

Vorbereitung
Die Grundlage für die Erstellung einfacher, aber konsistenter Diagramme ist die Definition der zu verwendenden Notation, also Diagramm- und Elementtypen, sowie die zu beschreibenden Aspekte und Sichten. Um eine einheitliche Erstellung und damit eine hohe Verständlichkeit zu erreichen, sollte dies vor der Erstellung für das gesamte Entwicklungsteam oder, hinsichtlich der Zusammenarbeit zwischen Teams, auch teamübergreifend definiert werden. Dabei haben die Entwicklungsteams die Freiheit, die Notation entsprechend Ihres spezifischen Kontextes und mit Hinblick auf die gegebenen Herausforderungen zu definieren. Im Folgenden werden einige Ideen erläutert, die Entwicklungsteams als Grundlage für Ihre eigene Definition verwenden können.

Architekturkonzepte
Die meisten Architekturkonzepte besitzen Aspekte, die sich mit Hilfe von einfachen Diagrammen gut illustrieren lassen. Die Verständlichkeit eines Konzeptes profitiert davon fast immer. Um ein Konzept zu beschreiben werden *Sichten* verwendet, das heißt man beschreibt konkrete Eigenschaften eines Systems getrennt von anderen, um so die Menge und damit die Komplexität der dargestellten Information zu reduzieren. Eine Sicht stellt einerseits einen konkreten Aspekt des Systems dar: Funktionen (Welche Einzelteile tun was zu welchem Zweck?), Daten (Welche Daten speichert und verarbeitet das System?), Deployment (Wie wird das System auf Ausführungsumgebungen und Maschinen verteilt?), Aktivitäten (Welche Prozesse und Aktivitäten gibt es für die Entwicklung, die Auslieferung und den Betrieb der Software?) und Technologien (Welche Technologien werden zur Realisierung des Systems eingesetzt?). Andererseits wird bei Sichten zwischen der Darstellung des Systems zur Laufzeit (bspw. interagierende Systeme, Datenflüsse, etc.) und dem System zur Entwicklungszeit (bspw. verwendete Entwicklungstechnologien, Datenstrukturen, etc.) unterschieden. Mit einer Sicht wird also ein Systemkonzept (bspw. Interaktion mit Fremdsystemen) mit Hilfe eines oder auch mehrerer Systemaspekte zu einer gegebenen Systemzeit beschrieben.
Jedes Architekturkonzept, das zur Lösung einer konkreten Problemstellung entwickelt wird, wie beispielsweise ein Konzept zur Synchronisation von Daten, ein Konzept für Daten-Backup oder ein Rechte- und Rollen-Konzept, ist ein guter Kandidat für die Beschreibung mit Hilfe von einfachen Diagrammen.

Neben den konkreten, projektspezifischen Konzepten gibt es einige Konzepte, die in den meisten Systemen relevant sind und die daher mit einfachen Diagrammen beschrieben werden sollten. Die folgende Auflistung erläutert die wichtigsten dieser Aspekte zusammen mit den Fragen die beantwortet werden sollten:

- Kontextabgrenzung: Die Beschreibung des Systems als Black Box zusammen mit den interagierenden Systemen im Kontext. Schnittstellen und Datenflüsse werden gezeigt. Beantwortete Fragen: Mit welchen Systemen interagiert das System? Zu welchem Zweck geschieht dies? Wie und welche Daten werden ausgetauscht?
- Grundstruktur: Die Zerlegung des Systems auf den obersten Ebenen, zum Beispiel in Applikationsteile, Teilsysteme, Schichten, Komponenten, etc. Beispielsweise eine Zerlegung in mobile Endgeräte und Backend, Unterteilung in verschiedene Applikationen, eine Zerlegung in die Schichten UI, Geschäftslogik, Datenhaltung, etc. Beantwortete Fragen: Aus welchen Teilen besteht das System? Wie interagieren diese miteinander, um die grundlegenden Funktionalitäten zu realisieren? Wie und welche Daten werden zwischen Ihnen ausgetauscht?

- Daten: Die im System verwendeten und mit anderen Systemen ausgetauschten Daten. Dies umfasst Aspekte wie Art der Daten, deren Struktur, Datenflüsse, Speicherung, Datenformate und Repräsentationen, Transformationen, Änderungshäufigkeiten, etc. Beantwortete Fragen: Mit welchen Daten arbeitet das System? Wie sind diese Daten beschaffen? Wie werden die Daten gespeichert, transportiert und verarbeitet?
- Deployment: Das Verpacken der Software in Deployment-Artefakte (bspw. .jar- oder .war-Dateien) und deren (mehrfache) Verteilung auf Ausführungsumgebungen und physische Rechenknoten, wie Server, Workstations, mobile Endgeräte, etc. Beantwortete Fragen: Wie wird die Software in Deploymentartefakte verpackt? Auf welchen Rechnern und Geräten werden welche Systemteile ausgeführt? Gibt es Redundanzen und Parallelbetrieb?
- Verwendete Technologien: Die zur Umsetzung verwendeten Sprachen, Frameworks, Ausführungsumgebungen, Buildsysteme, etc. Dies bezieht sich also auf jegliche Software die als fertiger Baustein zum Einsatz kommt und nur noch konfiguriert aber nicht mehr selbst geändert wird. Beantwortete Fragen: Welches Framework wird verwendet um ein Konzept umzusetzen? Wie wird das Framework verwendet? Welche Konfigurationen werden vorgenommen?

Neben diesen grundlegenden Konzepten besteht oftmals auch Bedarf für die Beschreibung weiterer Konzepte, wie Konfiguration, Logging, Authentisierung und Autorisierung, Administration, Upgrading oder Monitoring. Jeder dieser Aspekte kann durch ein oder mehrere Diagramme visualisiert werden:

Diagramm- und Elementtypen
Zur Erstellung von Sichten und damit Beschreibung von Architekturkonzepten mit Diagrammen muss festgelegt werden, welche Diagrammtypen verwendet werden sollen. Ein Diagrammtyp beschreibt die Menge der Elemente die darin verwendet werden sollten. Die Unified Modeling Language bietet sich als Grundlage für die verwendeten Diagramm- und Elementtypen an, da sie vielen Entwicklern bereits bekannt ist und sie oft bereits damit gearbeitet haben. Meist werden davon nicht alle Diagramme benötigt, die Auswahl einer Menge von häufig verwendeten Elementen hilft, die Komplexität zu reduzieren. Die folgende Auflistung zeigt einige Vorschläge für Diagramm- und Elementtypen, die häufig verwendet werden und damit eine sinnvolle Untermenge von UML bilden.

- Komponentendiagramm: Beschreibung des Systems zur Laufzeit mit Laufzeitentitäten und Zerlegung in Einzelteile. Verwendete Elemente und Relationen: Komponenten, Interfaces, Konnektoren, Nutzungsbeziehungen, Datenflüsse.
- Klassendiagramm: Darstellung der Realisierung von Komponenten mit Entwicklungszeitelementen (bspw. Klassen), sowie Datenstrukturen. Verwendete Elemente und Relationen: Klassen, Interfaces, Assoziationen, Aggregationen, Kompositionen, Vererbungen.
- Sequenzdiagramm: Interaktionen zwischen Systemeinzelteilen zur Laufzeit. Verwendete Elemente und Relationen: Komponenten, Objekte, Fragmente, Aufrufe.
- Verteilungsdiagramm: Verteilung der Software auf Ausführungsumgebungen und physische Maschinen und Endgeräte. Verwendete Elemente und Relationen: Knoten, Artefakte, Deploymentbeziehungen.

Die beschriebenen Sichten, Konzepte und Diagrammtypen zeigen eine Basis, auf der Entwicklungsteams Ihre eigene Notation definieren können. Essentiell dabei ist, dass diese Notation vor der Erstellung von Diagrammen festgelegt und konsistent in allen Diagrammen verwendet wird.

Durchführung
Entwicklungsteams können zu verschiedenen Zeitpunkten von der Erstellung und Verwendungen von einfachen, aber konsistenten Diagrammen profitieren.
Während der Konzepterarbeitung können mit Diagrammen Sachverhalte schnell illustriert und verdeutlicht werden. Sie dienen damit dem Ersteller des Konzeptes als Strukturierungswerkzeug. Werden Konzepte im Team erarbeitet, lässt sich anhand von Diagrammen sinnvoll diskutieren. Insbesondere Diagramme, die grundlegende Konzepte des Systems zeigen, eigenen sich darüber hinaus auch als Diskussionsgrundlage mit Stakeholdern außerhalb des eigentlichen Entwicklungsteam, wie Managern oder dem Product Owner. Während der Konzepterstellung steht das *Architekturdesign*, also die Erarbeitung von Lösungskonzepten im Vordergrund. Daher werden zu diesem Zeitpunkt Diagramme häufig am Whiteboard und mit Papier und Stift erstellt. Fotos und Scans, an einem sinnvollen Ort abgelegt, sichern die Verfügbarkeit für alle Teammitglieder.
Während der Umsetzung oder bei der *Wartung* können ebenfalls Diagramme erstellt oder auch angepasst werden. Auch hier dienen sie als Diskussionsgrundlage im Gespräch mit Kollegen oder als Gedächtnisstütze und zur Verdeutlichung des Konzeptes für den Einzelnen. Insbesondere bei größeren Systemen kann dann die Umsetzung von Diagrammen mit einem UML-Modellierungswerkzeug (wie bspw. Enterprise Architect von Sparx Systems) sinnvoll sein. Ihnen liegt ein Modell zugrunde, man erstellt also nicht nur Bilder, was Vorteile für die Nachverfolgbarkeit, Änderbarkeit und Verwaltung von Diagrammen hat. Auf der anderen Seite zwingen solche Werkzeuge den Ersteller sich an Formalismen zu halten. Deswegen sind sie nicht für den Einsatz während der Konzepterstellung zu empfehlen, weil sie dabei die Aufmerksamkeit häufig auf die Darstellung, weg von den eigentlichen Inhalten lenken.

Gütekriterien/Empfehlungen

Diagramme zur Illustration von Konzepten sind nur sinnvoll einsetzbar, wenn sie von allen Verwendern einfach gefunden und genutzt werden können. Ist die Hindernisschwelle zur Nutzung zu hoch, zahlt sich die Investition in Diagramme meist nicht aus, weil sie dann kaum genutzt werden. Aus diesem Grund sollten die Diagramme in einer einfach zu verstehenden Struktur an einem Ort abgelegt werden, der von allen relevanten Personen einfach zugreifbar ist, wie beispielsweise ein Netzlaufwerk oder ein Repositorium. Auch Wikis bieten diese Möglichkeit müssen aber aktiv gepflegt werden. UML-Modellierungswerkzeuge bieten oft auch die Möglichkeit zum verteilten Zugriff auf die Daten. Dazu muss allerdings ein zusätzliches Werkzeug verwendet werden.
Neben der Verfügbarkeit von Diagrammen bestimmt Ihre Aktualität maßgeblich Ihren Mehrwert. Ändert sich also ein Konzept, sollte ebenfalls das entsprechende Diagramm angepasst werden. Mit einem Modellierungstool sind Diagramme einfach änderbar. Verantwortlich dafür sollte derjenige sein, der das Konzept erstellt hat.

Risiken

Es sollte sichergestellt werden, dass der Grad der Formalisierung nicht zu hoch wird. Dies ist häufig eine Ursache dafür, dass die definierte Nation nicht verwendet wird, weil der Aufwand für die Erstellung von Diagrammen zu hoch wird.
Andererseits sollte sichergestellt werden, dass die definierte Notation konsistent verwendet wird, da sich ansonsten Missverständnisse ergeben können.

Einordnung in den agilen Referenzprozess

Mögliche Vorgänger

- Architektur\Architekturlösungen im Team entwickeln

Mögliche Nachfolger

- Architektur\Kontinuierliche Architekturbewertung durchführen

Mögliche Alternativen, verwandte Praktiken

- Grob- und Detailplanung bei der Implementierung nutzen
- Architekturentscheidungen dokumentieren

Einordnung in das PQ4Agile-Qualitätsmodell

- Fokus auf Wartbarkeit
- Grundsätzlich alle Produkt-Qualitätsattribute auf die die zu erarbeitenden Architekturkonzepte abzielen

Schlagworte

Architektur, Diagramme, Sichten, Konzepte, Lösungen

Weiterführende Informationen

Informationen im Internet

http://www.arc42.de/
http://www.uml.org/

Das Projekt PQ4Agile wird vom Bundesministerium für Bildung und Forschung im Rahmen der Maßnahme KMU-innovativ: IKT (01|S13032) gefördert.