Referenzarchitektur eines Ressourcen-Cockpits zur Unterstützung der Instandhaltung

Andreas Reidt[1], Markus Duchon[1], Helmut Krcmar[2]

[1]fortiss GmbH, München
[2]TU München, Institut für Wirtschaftsinformatik

Zusammenfassung

Durch die Trends der Digitalisierung und der Industrie 4.0 steigen die Anforderungen an die handelnden Personen im produzierenden Gewerbe. Unterstützende Prozesse wie die Instandhaltung trifft dieser Wandel besonders stark, da dort das Wissensmanagement auch wegen zunehmenden Alters der beteiligten Personen stark an Bedeutung zunimmt. Informationssysteme, wie ein Ressourcen-Cockpit, die den Instandhalter durch zielgerichtete und gebündelt dargestellte Informationen und Auswertungen unterstützen, stellen eine Lösung dieser Herausforderung dar. Die Entwicklung eines solchen Ressourcen-Cockpits ist jedoch kostspielig und erfordert ein tiefes Verständnis der technischen Möglichkeiten sowie der Domäne der Instandhaltung. Zusätzlich führen fehlende Standards und Architekturen zu dem Problem, dass unabhängig voneinander entwickelte Systeme inkompatibel zueinander sind. Dies erschwert eine branchen- und unternehmensübergreifende Zusammenarbeit, die in der Instandhaltung nötig ist. Um eine zukunftssichere Entwicklung zu ermöglichen, soll in diesem Beitrag die Referenzarchitektur eines Ressourcen-Cockpits zur Unterstützung der Instandhaltung vorgestellt werden, die die Entwicklung eines individuellen Ressourcen-Cockpits erleichtert.

1 Einleitung

RAen (RA) sind ein probates Mittel die Entwicklung von innovativen Softwarelösungen zu erleichtern. Sie stellen Domänenwissen, eine einheitliche Begriffswelt und eine Vorgehensweise für die Ausarbeitung konkreter Softwarearchitekturen bereit, wodurch die Basis für ein einheitliches Verständnis und Kompatibilität gelegt wird. Dies geschieht durch die Bereitstellung einer speziellen, abstrakten Architektur, welche die allgemeinen Richtlinien zur Spezifikation von konkreten Architekturen setzt (Angelov et al., 2009).

Für den Anwendungsfall des Ressourcen-Cockpit ist eine RA ein ideales Mittel um die Entwicklung zu steuern und zu vereinfachen. Nachdem in Reidt et al. (S. 23 ff.) das Vorgehen zur Erstellung einer RA anhand von spezifischen Anforderungen an ein Ressourcen-Cockpit und deren Bedeutung dargelegt

wurde, soll in diesem Beitrag die konkrete Referenzarchitektur für ein Ressourcen-Cockpit zur Unterstützung der Instandhaltung (RARC) vorgestellt werden.

Die in diesem Beitrag vorgestellten Elemente der RARC sind eine Zusammenfassung der Ergebnisse aus der Arbeit von (Reidt et al., 2016a). Zuerst wird daher der Hintergrund von RAen in Kapitel 2 erläutert. Darauffolgend wird die Darstellungsform in Kapitel 3 der RARC erklärt, bevor deren Inhalt in Kapitel 4 präsentiert wird. Ein zusätzlicher Aspekt bzgl. Sicherheit und Datenschutz wird der RARC in Kapitel 5 hinzugefügt. Abschließend werden die Erkenntnisse zusammengefasst.

2 Hintergrund Referenzarchitektur

RAen oder -modelle existieren in der (Wirtschafts-)Informatik und verwandten Domänen seit langer Zeit. Durch die Durchdringung von Informationstechnologie im produzierenden Gewerbe nimmt die Anzahl an Publikationen, welche als RAen bezeichnet werden, besonders im Kontext der Industrie 4.0 zu (vgl. Adolphs et al., 2015; Lin et al., 2015). Erdacht wurde dieses Konzept um u. a. die Entwicklung in komplexen Domänen zu erleichtern, indem Wissen und Konzepte bereitgestellt werden, oder um Aspekte zu vereinheitlichen. Beispiele für solche RAen wären AUTOSAR in der Automobilindustrie (AUTOSAR, 2015) oder die RA von Webbrowsern (Grosskurth & Godfrey, 2005). Demgemäß geben Referenzmodelle und -architekturen oft eine Grundstruktur eines Systems bzw. eine Vorgehensweise zur Erstellung vor oder beschreiben welche Teilelemente vorhanden sein müssen um ein solches System zu etablieren bzw. einen Prozess erfolgreich durchzuführen. Dem Referenzaspekt wird dadurch Ausdruck gegeben, dass die Architektur einen tendenziell allgemeingültigen Charakter haben muss. Dies bedeutet, dass eine RA über Unternehmensgrenzen und individuelle Fälle hinweg für eine bestimmte Domäne anwendbar sein sollte. Aus diesem Grund sind RAen ein beliebtes Mittel um Gemeinsamkeiten aufzuzeigen und die Kerne von Applikationen vergleichbar zu machen. Weiter wird in vielen Anwendungsfällen von RAen angestrebt durch eine RA eine Standardisierung bzw. Vereinheitlichung herbeizuführen oder schlicht Domänenwissen verfügbar zu machen.

Für das tiefere Verständnis des Begriffes der RA ist wiederum der bereits genannte Begriff des Referenzmodells von entscheidender Bedeutung. In Wissenschaft und Praxis wird dieser Begriff bereits ausgiebig verwendet. Im wissenschaftlichen Kontext werden Referenzmodelle als Informationsmodelle definiert, die als Ausgangslösungen zur Entwicklung projektspezifischer Mo-

delle Verwendung finden (Becker et al., 2007). RAen hingegen sind Referenzmodelle, die auf Softwareelemente heruntergebrochen werden, die Funktionen der Modelle kooperativ implementieren und die Datenflüssen zwischen den Elementen skizzieren (Bass et al., 2013). Zusammengefasst stellt eine RA im Kontext dieser Publikation eine spezielle, abstrakte Architektur dar, welche die allgemeinen Richtlinien zur Spezifikation von konkreten Architekturen setzt (Angelov et al., 2009).

Die Darstellung des Zusammenhangs zwischen den Begriffen und einer konkreten Softwarearchitektur wird in Abbildung 1 weiter veranschaulicht. Die Veränderung des Abstraktionsniveaus über die verschiedenen Modelle hinweg wird so ersichtlich. Insbesondere im Hinblick auf Softwarearchitekturen wird deutlich, dass eine RA diese nicht ersetzt bzw. ersetzen soll sondern eine Grundlage für die Erstellung eben dieser darstellt.

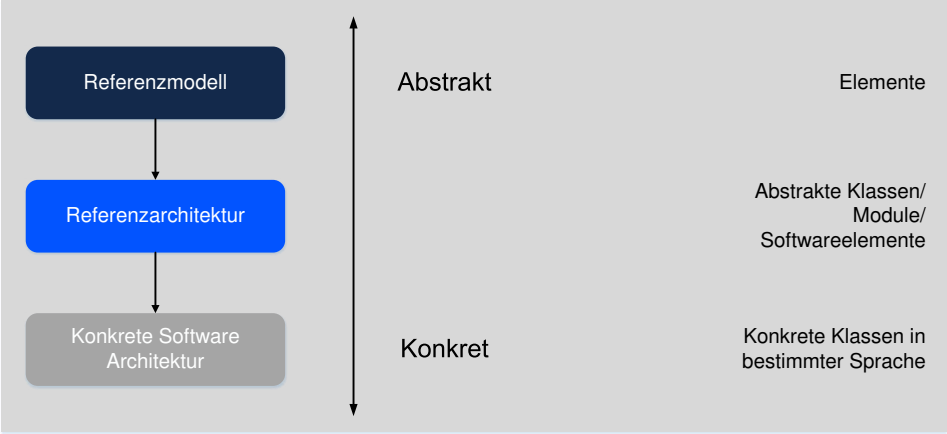

Abbildung 1 : Zusammenhang Referenzmodell, -architektur und Softwarearchitektur (Abbildung angelehnt an (Martínez-Fernández et al., 2013)

3 Darstellung der Referenzarchitektur

Eine verständlichen Darstellung ist nach (Cloutier et al., 2009) von entscheidener Bedeutung für den Erfolg einer RA. Für die RARC wurde daher eine Darstellungsart gewählt, welche sich an das 4+1-Sichten Softwarearchitekturmodell von (Kruchten, 1995) anlehnt. Diese Art von Architekturdarstellung, die auf verschiedenen Sichten bzw. Sichtweisen beruht, ist eine typische Darstellungsform zur Entwicklung komplexer IT-Systeme. Sie wurde neben

dem Aspekt der Verständlichkeit auch aus dem Grund gewählt, dass die inkludierten verschiedenen Sichten es ermöglichen zielgruppenspezifisch bestimmte Elemente einer Architektur mit unterschiedlicher Darstellungsform hervorzuheben.

Diese Art der Darstellung und der Grund für diese Entscheidung wird in Kapitel 3.1 erläutert. Zusätzlich werden In Kapitel 3.2 die Anpassung erklärt, die nötig sind, um anstelle von konkreten Softwarearchitekturmodellen eine RA für den angestrebten Zweck darzustellen.

3.1 Darstellung einer Softwarearchitektur nach Kruchten

Um die Architektur eines Software-intensiven Systems, wie des Ressourcen-Cockpits, darzustellen, müssen verschiedene Sichten auf die Architektur möglich sein bzw. diese dargestellt werden können. Eine der bekanntesten und verständlichsten Arten dies umzusetzen ist das 4+1-Sichten Softwarearchitekturmodell von Kruchten (1995). Aufgrund der einfachen Darstellung und Erweiterbarkeit im Hinblick auf die Nutzung für RAen, wurde diese Darstellung als Basis verwendet.

Das Modell unterscheidet ursprünglich zwischen 4+1-Sichten, welche jeweils einen Blickwinkel eines bestimmten Stakeholders (z. B. Endnutzer, Entwickler oder Projektmanager) auf das zu entwickelnde System darstellt. Dies erlaubt für jeden Projektbeteiligten eine übersichtliche Darstellung der benötigten Informationen. Darüber hinaus können, falls zusätzliche Blickwinkel wünschenswert sind, neue Sichten leicht eingefügt werden. Die Sichten selbst sind in ihrer Notation und ihrem Aufbau nicht festgelegt und die Auswahl der adäquatesten Darstellungsform wird an den jeweiligen Anwendungsfall angepasst. Nachfolgend werden die 4+1-Sichten beschrieben und anschließend wird auf Anpassungen hinsichtlich der RA eingegangen.

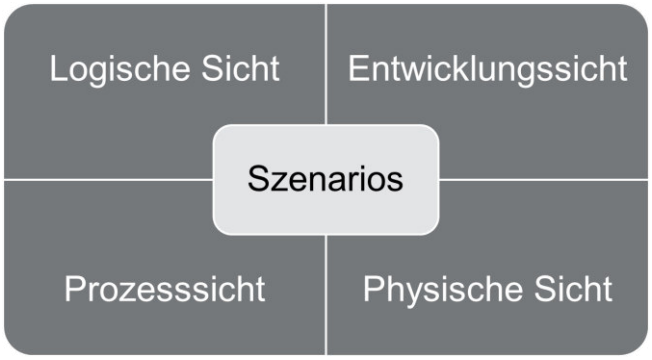

Abbildung 2 : 4+1 Sichten angelehnt an Kruchten (1995)

3.1.1 Logische Sicht

Diese Sicht beschreibt die Funktionalitäten und Services, welche dem Endnutzer zur Verfügung stehen. Die logische Sicht unterstützt vordergründig die funktionalen Anforderungen. Dem Endnutzer wird somit die logische Architektur dargestellt, womit er die zu erwartenden Services identifizieren kann. Um diese Ansicht zu erzeugen, wird das vorliegende System analysiert, in eine Menge von sogenannten „Schlüsselabstraktionen“ zerlegt und anschließend Gemeinsamkeiten und Designelemente zwischen diesen herausgearbeitet (Kruchten, 1995). Schlüsselabstraktionen bei Kruchten können dabei als Objekte oder Klassen angesehen werden. Die Informationen dieser Sicht werden meist per UML-Klassendiagramm, UML-Aktivitätsdiagrammen oder UML-Sequenzdiagrammen dargestellt.

3.1.2 Entwicklungssicht

Die Entwicklungssicht beschreibt das System vom Standpunkt eines Entwicklers und beschäftigt sich mit dem Softwaremanagement. Die logische Sicht bildet hierbei die Grundlage. Aus den einzelnen Objekten und Klassen der logischen Sicht werden in der Entwicklungssicht Softwarekomponenten erzeugt. Deren Verteilung auf verschiedene Subsysteme und Schichten wird bspw. durch UML-Komponenten- oder -Paketdiagramme abgebildet. Zusätzlich werden diese Komponenten in eine Hierarchie einsortiert, wodurch sich eine für den Entwickler hierarchische Darstellung der zu implementierenden Funktionalitäten ergibt. Diese sind über klare Interfaces voneinander getrennt. Hierdurch können Arbeitspakete einzelnen Entwicklern oder kleinen Teams zugewiesen werden und erleichtert auch die Wartung/Erweiterung der Software selbst.

3.1.3 Prozesssicht

In der Prozesssicht werden die dynamischen Aspekte des Systems verdeutlicht. Der Zusammenhang zwischen den Elementen aus der logischen Sicht, deren Zuordnung zu Kontrollflüssen, Kommunikationswegen und der notwendigen Synchronisation werden beschrieben. Dadurch wird das Laufzeitverhalten ersichtlich und zusätzlich nichtfunktionale Anforderungen wie Parallelität, Verteilung, Integration, Performance und Skalierbarkeit hervorgehoben.

3.1.4 Physische Sicht

Die physische Sicht beschäftigt sich mit der Systemtopologie, der Verteilung und Kommunikation der verschiedenen Komponenten auf physischer Ebene. Es ist eine Sicht für Systemarchitekten, die die Verteilung des zu entwickelnden Systems auf verschiedene Hardwarekomponenten und Netzwerkverbindungen plant.

3.1.5 Szenarien

Die der Architektur zugrundeliegenden Szenarien bilden die fünfte Sicht. Diese Szenarien stellen die wichtigsten Anwendungsfälle der Architektur bzw. der Anwendung dar. Die Szenarien sind dabei teils redundant mit den vorhergehenden Sichten (daher das „+1“), jedoch helfen die dort enthaltenen Abläufe Architekturelemente zu identifizieren, zu veranschaulichen und die Architektur mit all ihren Sichten zu überprüfen. Die Szenarien können in Form von bspw. Use Cases grafisch oder textuell beschrieben werden. Die beschriebenen Use Cases gehen schlussendlich vollständig in den beschriebenen Sichten auf und dienen als Implementierungsgrundlage für Anwendungen. Sie verbinden damit alle anderen Sichten.

3.2 Anpassungen für eine Referenzarchitektur

Die vorgestellten Sichten bilden in den meisten Fällen eine Grundlage zur Darstellung von konkreten Softwarearchitekturen. Im Falle einer abstrakteren RA, wie der RARC, werden jedoch andere Anforderungen an die Darstellung und die Architektur selbst gestellt. Sie muss als Vorlage benutzt werden können, um technische, konkrete Architekturen zu entwickeln. Aus diesem Grund wurden die Sichten von Kruchten dahingehend angepasst, dass die Modellierung einer abstrakten RA möglich ist. Das Ergebnis dieser Anpassung ist in Abbildung 3 zu erkennen. Dort sind die einzelnen Sichten und deren Verbindungen zueinander abgebildet. Besteht eine Verbindung zwischen zwei Sichten so sagt dies aus, dass eine Sicht auf den Informationen der anderen aufbaut. Nachfolgend werden die Sichten und deren Auswahl beschrieben.

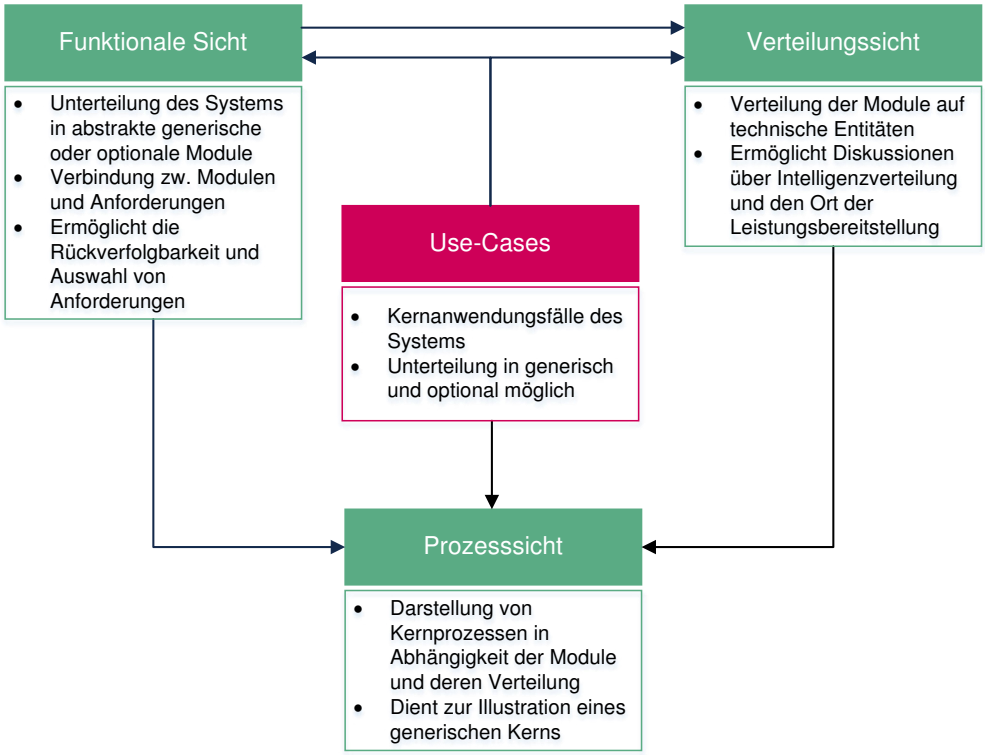

Abbildung 3 : Angepasste Sichten der RA (Eigene Darstellung)

Wie im vorherigen Abschnitt dargelegt, ist die logische Sicht im ursprünglichen Modell von Kruchten sehr detailliert. Dort werden die funktionellen Anforderungen in Klassen oder Objekte heruntergebrochen und miteinander verbunden und/oder in Vererbungshierarchien gebracht. Die Entwicklungssicht teilt die Klassen in Module, Subsysteme und verschiedene Schichten ein, die über spezifische Schnittstellen miteinander kommunizieren müssen. Diese Darstellung ist für eine RA zu spezifisch, da konkrete Klassen, Schnittstellen und Subsysteme gegenläufig zu einer hohen Abstraktion und einer generischen Sichtweise wären. Eine individuelle Adaptierung auf die eigenen technischen Gegebenheiten und Systeme wäre demzufolge kaum möglich.

Daher wurde die ursprüngliche logische Sicht dahingehend angepasst, dass sie optionale und generische Module einschließlich verbundener Anforderungen darstellt. Module sind in dieser Sicht, nicht wie bei Kruchten, Elemente, die verschiedene Klassen oder Objekte enthalten. Module stellen im Kontext der RARC eine logische Funktionseinheit dar und entsprechen eher einem

Modul im Sinne der ursprünglichen Entwicklungssicht. Jedoch sind sie auf einem deutlich höheren Abstraktionsniveau und enthalten daher keine konkreten Klassen, Verweise auf technische Bibliotheken oder Vererbungshierarchien. Sie stellen vielmehr eine abstrakte Funktionsbeschreibung einer logischen Einheit mitsamt Anforderungen, Unterfunktionen und bidirektionale Verbindungen zu anderen Modulen dar. Dies ermöglicht eine einfache Darstellung von nötigen und optionalen Funktionen in Verbindung mit generischen und optionalen Anforderungen. Durch diese nachvollziehbare Verbindung von Modul und Anforderung werden die individuelle Auswahl der Module anhand eigener Anforderungen, sowie eine transparente Rückverfolgbarkeit dieser bei Änderungen, erreicht. Weiterhin bieten die Module eine Grundlage für die Implementierung, ohne dass entscheidende Freiheitsgrade bei dieser eingeschränkt werden. Der Name dieser Sicht ist demgemäß Funktionale Sicht und auf eine isolierte Entwicklungssicht wurde komplett verzichtet.

Weitergehend ist die physische Sicht stark angepasst worden. Sie wurde ursprünglich genutzt, um den physischen Aufbau mitsamt Hardwareallokation so zu beschreiben, dass die Architektur nicht-funktionale Anforderungen wie Performance oder die Verfügbarkeit erfüllen konnte. Da der physische Aufbau von der Wahl der Module, deren Implementierung und der Größe der Applikationen im Allgemeinen abhängt, wäre eine solche Darstellung nicht zielführend. Sie ist im Kontext der RA daher zu einer Verteilungssicht umgewandelt worden und zeigt auf, welche Module aus der Funktionalen Sicht auf welchen Entitäten (Backend, mobiles Endgerät, Maschine) verfügbar sein müssen. Der Fokus liegt also auf der abstrakten Verteilung von Komponenten und Funktionen. Dies hat den Vorteil, dass die Stakeholder der Entwicklung einfach erkennen können, welche Funktionen zentral, dezentral oder mobil verfügbar sind. Hierdurch lässt sich die Intelligenzverteilung zwischen Backend, mobilem Endgerät und Anlage klar strukturieren und festlegen. Die Frage nach der Intelligenzverteilung stellt einen enorm wichtigen Punkt bei der Entwicklung von Industrie 4.0 Applikationen dar, die die Fähigkeiten von CPS effizient ausnutzen wollen.

Die Use Case Sicht ist im Kontext der RARC ähnlich wie die Szenarien von Kruchten umgesetzt worden. Die dort enthaltenen Anwendungsfälle beschreiben die wichtigsten Aspekte der Architektur und dienen dazu, Architekturelemente der anderen Sichten zu identifizieren. Für die hier erstellte RA ist es jedoch von Nöten, unterschiedliche Use Cases von unterschiedlichen Unternehmenstypen abzubilden, um alle Aspekte eines Ressourcen-Cockpits hinsichtlich generischen und optionalen Elementen darzustellen. Daher sind z. T. ähnliche Use Cases verschiedener Unternehmen vorhanden.

Diese Use Cases bilden auch die Grundlage der Modellierung der Prozesssicht. In dieser werden die Use Cases per UML-Aktivitätsdiagramm so modelliert, dass eine Verbindung zwischen den Use Cases, Funktionaler Sicht und Verteilungssicht einfach ersichtlich ist. Dies geschieht für alle Use Cases. Anhand dieser Informationen werden weitere generische und optionale Teile von Prozessen der RARC identifiziert. Im Falle der RARC wird ebenso eine Grundaktivität identifiziert, welche die grundlegende Funktionsweise eines Ressourcen-Cockpits umspannt und die Use Cases vereinheitlicht. In dieser Grundaktivität werden die identifizierten generischen und optionalen Teile der Prozesse aggregiert und mit den übrigen Sichten verbunden. Mit dieser Form der Darstellung unterscheidet sich die Prozesssicht stark von der bei Kruchten verwendeten, da ein höherer Abstraktionsgrad und eine zweite Form der Verknüpfung der übrigen Sichten erreicht werden. Sie stellt damit die generische Überführung der Use Cases mitsamt Verknüpfung zu den restlichen Sichten dar und ist die Basis für eine iterative Weiterentwicklung der Architektur. Weiterhin wird anhand dieser Sicht das Wissen über die Gesamtarchitektur punktgenau dargelegt, sodass eine Evaluation der restlichen Sichten durch das Wechselspiel erreicht werden kann. Im Kontext der RARC wird insbesondere durch diese Sicht die Kommunikation zwischen allen Stakeholdern ermöglicht.

4 RARC

Nachdem die Darstellungsart skizziert wurde, werden die einzelnen Sichten und ihr konkreter Inhalt beschrieben. Genauere Ausführung mitsamt aller Details der einzelnen Sichten sind ausführlich bei (Reidt et al., 2016a) zu finden.

4.1 Use Case View

Die RARC enthält insgesamt neun spezifische Use Cases, welche Referenzprozesse für die jeweiligen Unternehmen und Instandhaltungsarten darstellen. Diese Use Cases werden ausgiebig bei (Reidt et al., 2016a) diskutiert. Aus den vorgestellten Use Cases ist ersichtlich, dass diese in sechs Kategorien eingeteilt werden können. Diese wären:

- Instandsetzung von Anlagen
- Wartung/Inspektion von Anlagen durch Bediener
- Wartung/Inspektion von Anlagen durch Instandhalter
- Zustandsabfrage und Wartungsbedarfsanzeige von Anlagen
- Auswertung über die Verfügbarkeit der Anlagen sowie die
- Auswertung über die Fehlerhistorien

Die Use Case Sicht beinhaltet daher den Grundstock an nötigen Use Cases zur Benutzung eines Ressourcen-Cockpits.

4.2 Funktionale Sicht

Die funktionale Sicht umfasst insgesamt 40 Module und deren Verbindung zueinander. Nachfolgend sind die enthaltenen Module zusammengefasst dargestellt. Die hell dargestellten Module sind hierbei optionale Module und die in dunkel hinterlegten Module sind generisch. Generische Module sind Module, die sich in jedem Ressourcen-Cockpit befinden müssen, optionale sind abhängig vom konkreten Anwendungsfall des Ressourcen-Cockpits (z. B. Kundendienst) und können, müssen jedoch nicht, implementiert werden.

M.1. Handlungsleitfäden/ Checklisten/Prüflisten	M.2. Dokumenten-management	M.3. Wiki	M.4. Anlagen-informationen
M.5. Anlagenübersicht und -auswertung	M.6. Berechnung/ Zugriff auf Produktionsplanung	M.7. Interne Navigation	M.8. Kataster für Hilfs- und Betriebsstoffe der Anlagen
M.9. Auftragsverwaltung/ Priorisierung	M.10. Mitarbeiter-management	M.11. Wartungs-management	M.12. Auftragsmanagement in Verbindung mit ERP
M.13. Schichtbuch-funktionalität/ Synchronisation	M.14. Work and People Tracking	M.15. Arbeitszeiterfassung	M.16. Fehlererkennung & Condition Monitoring
M.17. Fehlerdatenbank	M.18. Fehlermeldung und -darstellung	M.19. Predictive Maintenance	M.20. Synchronisation Hersteller und Betreiber
M.21. Navigation	M.22. Reisekosten-abrechnung	M.23. Serviceprotokolle	M.24. Vertragsmanagement
M.25. Signierfunktion	M.26. Kommunikation	M.27. Konnektivität/ Intranetzugang	M.28. Ersatzteilmanagement
M.29. QR-CODE auslesen	M.30. Remote Zugriff/ Fernwartung auf Anlagen/Maschinen	M.31. Technisches Nutzermanagement	M.32. Verarbeitung audiovisueller Medien
M.33. Funktionalitäten mobiles Endgerät	M.34. Synchronisation zw. Geräten und Systemen	M.35. Maschinenspezifika	M.36. Pluginintegration
M.37. Wetterdaten	M.38. Semantische Suche	M.39. Personalisierung des Systems	M.40. Telefonersatz

Abbildung 4: Darstellung der funktionalen Module der RARC

Die Module werden bei (Reidt et al., 2016a) nach dem in Tabelle 1 dargestellten Schema beschrieben. Hierdurch wird gewährleistet, dass die technische Umsetzung, die Verbindung zu anderen Modulen, wie auch die Verbindung zu anderen Sichten und Anforderungen einfach ersichtlich ist.

Tabelle 1: Vorlage und Aufbau der Modulbeschreibungen

Name der Komponente	**In dieser Sektion wird der Name der Komponente eingetragen**
Darstellung in Referenzarchitektur	In dieser Sektion wird das Modul mitsamt den enthaltenen Anforderungen in einem Block dargestellt. Hierdurch ist die Zusammensetzung des Moduls aus optionalen wie auch generischen Anforderungen ersichtlich.
Zweck/Ziel	Hier wird das Ziel bzw. der Zweck des Moduls beschrieben.
Generisch/ optional	Dieses Feld beschreibt, ob das Modul generisch oder optional ist.
Funktions-beschreibung	Die Funktionsbeschreibung des Moduls wird in dieser Sektion gegeben. Unter Funktionsbeschreibung wird hier das allgemeine Vorgehen zur Erfüllung des Ziels bzw. des Zweckes verstanden. Es können ebenso auf spezifische Herausforderungen während der Implementierung hingewiesen und gegebenenfalls deren Lösung skizziert werden.
Mögliche Unterfunktionen	Eine Aufteilung des Moduls in Unterfunktionen wird hier anhand der Funktionsnennung aufgelistet.
Verbindungen zu anderen Modulen	Hier werden die Module genannt mit welchen das vorliegende Modul im Austausch steht.
Voraussetzungen zur Funktions-tüchtigkeit	Technische, organisatorische Voraussetzungen zur Funktionstüchtigkeit des Moduls werden hier erläutert.
Erfüllte Anforderungen	In diesem Abschnitt werden die jeweiligen Anforderungen genannt, die durch das Modul abgedeckt werden.
Verbundene Prozesse	Hier werden die jeweiligen Prozesse genannt, in denen das Modul vorkommt.
Mögliche Technologien	Mögliche Technologien zur Umsetzung des Moduls werden hier genannt.
Aufteilung nach Entität	Die Aufteilung des Moduls auf Entitäten wird hier erklärt. Eine komplette Darstellung mit allen Modulen und deren Aufteilung ist darüber hinaus in Kapitel 4.3 zu finden.

4.3 Verteilungssicht

Die Verteilungssicht enthält alle Module aus der Funktionalen Sicht und stellt dar, auf welcher Entität das jeweilige Modul vorhanden sein muss. Als Entitäten werden drei Systeme, auf denen sich Module befinden, bezeichnet und mit „E“ abgekürzt. Die drei Entitäten sind dabei:

- E1. Maschine/Anlage - Hiermit sind maschinennahe Systeme gemeint, die sich entweder in der Maschine selbst, oder an einem direkt an der Maschine lokal angeschlossenen System, befinden.
- E2. Mobiles Endgerät - Hiermit ist ein mobiles Endgerät gemeint, welches der Instandhalter, Bediener oder eine dem Instandhaltungsmanagement zugeordnete Person zur Unterstützung von Instandhaltungsaufgaben nutzt. Im Kontext der untersuchten Unternehmen wurde ein Tablet als mobiles Endgerät erwünscht.
- E3. Backend - Das Backend stellt den zentralen Server dar, auf denen die meisten Informationen zusammenlaufen. Es ist nicht spezifiziert welche Leistung oder welche Art von Rechner das Backend haben bzw. sein sollte. Der Zugang zu dem Backend wird in der Regel durch einen Desktop PC oder einen Laptop erlangt. Hierfür ist i. d. R. ein Inter- bzw. Intranet Zugang notwendig.

Anhand dieser Übersicht lässt sich die Intelligenzverteilung des Ressourcen-Cockpits und von bestimmten Modulen erläutern und planen. Der Trend zu einem stärker dezentralen System lässt sich ebenso hier erkennen. Fokussierter auf den dezentralen Aspekt eines CPS sind die einzelnen Teilmodule bei (Reidt & Krcmar, 2016) diskutiert worden.

4.4 Prozesssicht

Die Prozesssicht der RARC beinhaltet die wichtigsten Prozesse basierend auf den Use Cases. Diese werden mithilfe von abgewandelten UML-Aktivitätsdiagrammen so dargestellt, dass klar ersichtlich ist, welche Aktivität auf welchem Modul und auf welcher Entität abläuft. Zusätzlich sind generische, wie auch optionale Aktivitäten und Abzweigungen eingezeichnet. Neben diesen Prozessen wurde auch eine Grundaktivität abgeleitet, welche den Kern des Systems darstellt. Durch diese Sammlung an Prozessen ist klar ersichtlich welche Prozesse durch ein Ressourcen-Cockpit wie zu unterstützen sind und an welchen Punkten Entwicklungspotential vorhanden ist. Aufgrund der Komplexität der Prozesse und der Darstellung wird hier auf die Arbeit von (Reidt et al., 2016a) verwiesen.

5 Sicherheitsaspekte

Die RA wurde im Laufe des Projektes durch Sicherheitsaspekte erweitert. Diese sind zwar kein fester Bestandteil der RARC, erweitern diese jedoch um essentielle Faktoren hinsichtlich Sicherheit und Datenschutz. Im Zusammenhang mit der Sicherheit muss bedacht werden, dass sich der ursprüngliche Datenschutzgedanke der Datenvermeidung nur sehr schwer mit den digitalen Geschäftsmodellen in Einklang bringen lässt. Wie auch in der Studie im Rahmen des Projekte „IKT Wandel“ (Gonzalez et al., 2016) festgestellt worden ist, setzt die Nutzung von Daten durch Dritte – sowohl bei der privaten, aber speziell auch bei der industriellen Nutzung – ein ausreichendes Vertrauen in einen Umgang mit den Daten voraus. Dabei gewinnt der Begriff der Datenhoheit, also welche Entitäten unter welchen Umständen und Voraussetzungen Zugriff auf Daten erhalten, immer mehr an Bedeutung. Davon unberührt bleiben natürlich Maßnahmen zur Reduzierung der Angriffsfläche. Gerade dieser Punkt stellt bei einer Vielzahl an verbundenen Geräten (Maschinen, Sensoren, etc.) im Industrie 4.0 Umfeld eine besondere Herausforderung dar.

Um einen Überblick über die bestehenden Herausforderungen und Bedrohungen, Sicherheitsziele sowie geeignete Maßnahmen für deren Umsetzung zu erhalten wurde die aktuelle Wissensbasis aus Literatur und Praxis zusammengetragen und analysiert. Anhand dieser Literaturrecherche ergaben sich unter Berücksichtigung des Industrie 4.0 Kontextes elf relevante Schutzziele. Die drei am häufigsten genannten waren:

- Verfügbarkeit (Availability),
- Integrität (Integrity) und
- Vertraulichkeit (Confidentiality),

Diese entsprechen dabei den sogenannten Schutzzielen der Informationssicherheit (Krcmar, 2015). Die identifizierten Herausforderungen lassen sich in die folgenden vier Ebenen unterteilen:

- Endpunktsicherheit,
- Kommunikationssicherheit,
- Datenverteilung
- sichere Speicherung sowie
- Steuerung der Überwachung.

Die einzelnen Schutzziele zusammen mit den technischen Möglichkeiten zur Umsetzung lassen sich verschiedenen Abstraktionsebenen zuordnen. Diese Ebenen wären im Kontext der Industrie 4.0:

- Anwendungsebene,
- Wahrnehmungsebene,
- Vermittlungsebene und
- physikalische Ebene.

Für eine detailliertere Beschreibung der Ziele, Herausforderungen und Bedrohungen sei an dieser Stelle auf (Reidt et al., 2016b)) verwiesen.

Basierend auf diesen generischen Kategorisierungen können unterschiedliche Bedrohungsszenarien analysiert und geeignete Maßnahmen zusammen mit den technischen Möglichkeiten abgeleitet werden. Bei dem Beispiel einer sicheren Fernwartung der RARC, bei der eine Kommunikationsverbindung von einem entfernten Rechner zu einer Anlage aufgebaut werden soll, ist eine der betroffenen Bedrohungsebenen die Vermittlungsebene. Ausgehend von dieser Ebene lassen sich Herausforderungen für die Kommunikationssicherheit, die Endpunktsicherheit und die Steuerung und Überwachung ableiten, zusammen mit technischen Möglichkeiten, um die damit verbundenen Schutzziele zu erreichen. Beispiele dieser Möglichkeiten umfassen in diesem Fall die Verschlüsselung oder die Nutzung sicherer Protokolle (Kommunikationssicherheit), die Authentifizierung und Autorisierung oder Whitelisting und Zugriffskontrolle von bestimmten Anwendungen (Endpunktsicherheit) sowie eine Initiierung der Fernwartung ausgehend von der Anlage (Steuerung und Überwachung), um dadurch nach außen offene Ports zu vermeiden und somit die Angriffsfläche zu reduzieren.

6 Zusammenfassung

In diesem Beitrag wurde die Referenzarchitektur eines Ressourcen-Cockpits vorgestellt. Zuerst wurden aufbauend auf den Grundlagen der RARC in Reidt et al. (S. 23 ff.) der Hintergrund von Referenzarchitekturen beleuchtet. Anschließend wurde die Darstellungsart der Referenzarchitektur dargelegt, um anhand dieser Darstellung den Inhalt auszuarbeiten. Eine Erweiterung durch Sicherheitsaspekte wurde als abschließender Teil der Referenzarchitektur vorgestellt. Anhand dieser Informationen ist es möglich, Ressourcen-Cockpits schneller, zielgerichteter und kompatibler zu entwickeln.

7 Literaturverzeichnis

Adolphs, P., Bedenbender, H., Dirzus, D., Ehlich, M., Epple, U., Hankel, M., Zusammenspiel Datensammlung

Wollschlaeger, M. (2015). *Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)*. *VDI /VDE Statusreport*.

Angelov, S., Grefen, P., & Greefhorst, D. (2009). A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness. In *2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA 2009*. Cambridge, UK: IEEE. http://doi.org/10.1109/WICSA.2009.5290800

AUTOSAR. (2015). AUTOSAR - AUTomotive Open System ARchitecture. Abgerufen 1. September 2015, von http://www.autosar.org/specifications/

Bass, L., Clements, P., & Kazman, R. (2013). *Software Architecture in Practice* (3. Aufl.). Upper Saddle River, New Jersey: Addison Wesley. http://doi.org/10.1024/0301-1526.32.1.54

Becker, J., Delfmann, P., Knackstedt, R., & Kuropka, D. (2007). *Wissensmanagement mit Bordmitteln*. (J. Becker & R. Knackstedt, Hrsg.), *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung*. Berlin Heidelberg: Springer. http://doi.org/10.1007/978-3-642-52449-3

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., & Bone, M. (2009). The Concept of Reference Architectures. *Systems Engineering*, *13*(1). http://doi.org/10.1002/sys.20129

Gonzalez, A. A., Becker, K., Cheng, C.-H., Döricht, V., Duchon, M., Fehling, M., … Zoitl, A. (2016). *Digitale Transformation - Wie Informations- und Kommunikationstechnologie etablierte Branchen grundlegend verändern*. *Abschlussbericht des vom Bundesministerium für Wirtschaft und Technologie geförderten Verbundvorhabens „IKT-Wandel“*. München.

Grosskurth, A. J., & Godfrey, M. W. W. (2005). A reference architecture for web browsers. In *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)*. Budapest, Hungary: IEEE. http://doi.org/10.1109/ICSM.2005.13

Krcmar, H. (2015). *Informationsmanagement*. http://doi.org/10.1007/978-3-662-45863-1

Kruchten, P. (1995). Architecture Blueprints - the „4+1“ View Model of Software Architecture. *IEEE Software*, *12*(November). http://doi.org/10.1145/216591.216611

Lin, S.-W., Miller, B., Durand, J., Joshi, R., Didier, P., Chigani, A., Witten, B. (2015). *Industrial Internet Reference Architecture*. *Technical Report*. Industrial Internet Consortium.

Martínez-Fernández, S., Ayala, C., Franch, X., Martins, H., & Ameller, D. (2013). A Framework for Software Reference Architecture Analysis and Review. In M. Solari Buela & A. C. Dias Neto (Hrsg.), *10th Experimental Software Engineering Track Workshop (ESELAW)*. Montevideo, Uruguay.

Reidt, A., Duchon, M., & Krcmar, H. (2016a). *Referenzarchitektur eines Ressourcen-Cockpits zur Unterstützung der Instandhaltung* (1. Aufl.). München: Fortiss GmbH. http://doi.org/10.13140/RG.2.2.16563.84003

Reidt, A., Duchon, M., & Krcmar, H. (2016b). Sicherheitsaspekte von Industrie 4.0. In E. Müller & A. Bullinger-Hoffmann (Hrsg.), *Tagungsband „Smarte Fabrik & Smarte Arbeit – Industrie 4.0 gewinnt Kontur“ VPP2016 – Vernetzt planen und produzieren*. Chemnitz: TU Chemnitz.

Reidt, A., & Krcmar, H. (2016). Referenzarchitektur für Cyber-physische Systeme zur Unterstützung der Instandhaltung. In V. Nissen, D. Stelzer, S. Straßburger, & D. Fischer (Hrsg.), *Multikonferenz Wirtschaftsinformatik (MKWI) 2016*. Ilmenau: Universitätsverlag Ilmenau.

Autoren

Andreas Reidt

Andreas Reidt studierte Wirtschaftsinformatik an der TU Darmstadt. Bevor er als wissenschaftlicher Mitarbeiter bei der fortiss GmbH forschte, arbeitete er für die Commerzbank und SAP. Bei fortiss beschäftigt er sich in dem Geschäftsbereich Business Model und Service Engineering mit der Digitalisierung im Kontext der Industrie 4.0, berät Unternehmen und arbeitet und forscht an neuartigen Softwarearchitekturen und Plattformentwicklungen. Zusätzlich betreut er ein vom fortiss entwickeltes IT-Benchmarking.

Markus Duchon

Markus Duchon studierte Informatik und promovierte 2013 im Bereich der mobilen und verteilen Systeme an der Ludwig-Maximilians-Universität München in Kooperation mit der Siemens Corporate Technology. Dort war er in der Abteilung für Software Architekturen und Plattformen tätig. Am fortiss leitet er die Entwicklung einer Software zur intelligenten und energieeffizienten Steuerung von Gebäuden. Seine Forschungsschwerpunkte umfassen Verfahren zur Optimierung verteilter Systeme, kontextsensitive Systeme sowie Software Architekturen u.a. für den Smart Grid.

Helmut Krcmar

Helmut Krcmar ist Inhaber des Lehrstuhls für Wirtschaftsinformatik an der TU München. Er ist wiss. Geschäftsführer der fortiss gGmbH - bayerisches Landesinstitut und An-Institut der TU München. Von 2010 bis 2013 war er Dekan der Fakultät für Informatik der TU München. Er forscht auf dem Gebiet des Informationsmanagements, der IT-ermöglichten Wertschöpfungsnetze, dem Dienstleistungsmanagement, dem Computer Supported Cooperative Work und der Informationssysteme für IT-Service Provider, im Gesundheitswesen und im öffentlichen Bereich.