Anwendung von modellbasiertem Management zur adaptiven Verwaltung von eingebetteten Dienstesystemen

Stefan Illner, Ingo Lück, Andre Pohl und Heiko Krumm
Universität Dortmund
{illner, pohl, krumm}@ls4.cs.uni-dortmund.de
ingo.lueck@materna.de

Zusammenfassung

Im Rahmen des europäischen SIRENA Projektes [1] arbeitete unsere Arbeitsgruppe an einem Konzept zur kontextsensitiven, adaptiven Verwaltung von eingebetteten Dienstesystemen. Das SIRENA Projekt hatte zum Ziel die Konzepte serviceorientierter Architekturen (SOAs) auf die Welt der eingebetteten Systeme anzuwenden. Da diese Systeme von Natur aus mit eher geringen Rechenressourcen versehen sind, sind intensive Rechenoperationen zur Laufzeit nicht wünschenswert. Der von unserer Arbeitsgruppe entwickelte Ansatz gliedert sich in daher zwei verschiedene Phasen. Zum Einen in eine Modellierungsphase, in der basierend auf den Konzepten des modellbasierten Managements ein Modell des zu verwaltenden Systems, z.B. mit den vorhandenen Diensten, Geräten und Nutzern, erstellt wird. Mit Hilfe von Ableitungsregeln können dann in einem (halb-)automatischen Prozess aus abstrakten Management-Zielen konkrete low-level Policies abgeleitet werden. Diese Policies werden von den in der zweiten Phase verwendeten leichtgewichtigen Management-Diensten zur Laufzeit durchgesetzt.

1. Modellbasiertes Management

Der Ansatz des Modellbasierten Managements ist im Feld des technischen Netz-, System- und Anwendungsmanagements angesiedelt. Managementanwendungen ergänzen das System, überwachen es selbständig und greifen bei Bedarf korrigierend ein [2,3]. Die Managementanwendungen sind aber ihrerseits nicht unproblematisch. Sie müssen sehr sorgfältig auf das Zielsystem zugeschnitten werden und den Kosteneinsparungen bei der Administration stehen schnell hohe Entwicklungs-, Anpassungs- und Pflegekosten für die Managementanwendungen gegenüber. Das modellbasierte Management verschiebt darum den Schwerpunkt der Managementsystementwicklung von der eigentlichen Anwendungsentwicklung hin zur Systemidentifikation. Sie erfolgt durch Modellierung. Das Zielsystem wird in seinem konkreten Aufbau, in den von ihm erbrachten Diensten und in den Managementvorgaben, den abstrakten Management-Policies, mit einem dreischichtigen Systemmodell dokumentiert. Das Modell wird danach um die Darstellung der benötigten Managementmechanismen erweitert, und die Konfiguration der Managementanwendungen sowie die zur Steuerung der Mechanismen verwendeten Low-Level-Policies werden aus dem erweiterten Modell generiert. Modellierung und Erweiterung werden durch ein interaktives graphisches Werkzeug, Modellklassenbibliotheken sowie Graphersetzungssysteme unterstützt.

Die in der obersten Modellschicht verwendeten abstrakten Policies stellen hier ausschließlich Anforderungen dar:

- *Funktional* notieren *Zugriffstripel* aus Rolle, Zugriffsmodus und Objekt, welche abstrakten Funktionen für welche Subjekte verfügbar sein sollen, und im Komplement, welche Funktionen im Sinne einer Zugriffskontrolle verwehrt werden müssen. Das dahinter stehende Konzept entspricht dem bekannten NSA-Modell der Rollenbasierten Zugriffskontrolle (RBAC).
- *Nicht-funktionale Anforderungen* werden ähnlich zu Service Level Agreements (SLAs) durch Attributierung von Zugriffsmodi, Objekten, Rollen und Zugriffstripeln durch beispielsweise einen Sicherheitsanforderungsvektor (bestehend aus Maßzahlen zur Stärke der geforderten Integrität, Vertraulichkeit, Verfügbarkeit und Nachvollziehbarkeit) sowie einen

Zuverlässigkeitsvektor und einen Leistungsvektor notiert.

- Die Anforderungen (sowohl funktional als auch nicht-funktional) können *modal* differenziert werden. Dazu wird das erweiterte RBAC-Modell, Generelized Role-based Access Control Model (GRBAC) verwandt. Es führt – zusätzlich zu den in RBAC vorhandenen Subjekt-Rollen – Umgebungs- und Systemrollen, mit deren Hilfe Umgebungs- und Systemzustände referenziert werden können ein (z.B. *zur Tagesarbeitszeit, vom Büroarbeitsplatz aus, bei Regenwetter, wenn die Akkureserve zur Neige geht*).

In der mittleren Modellschicht werden System und Policies auf einer Dienste-orientierten Ebene dargestellt. Jedem abstrakten Zugriffstripel entsprechen hier ein oder mehrere Möglichkeiten von Dienstnutzungen. Die abstrakten Umgebungsrollen werden auf dieser Ebene zu Formeln über Zuständen bestimmter, abstrakter Systemparameter verfeinert, die den Systemzustand konkreter beschreiben. Den abstrakten Systemparametern können auf dieser Ebene einschränkende Definitionen, wie z.B. obere und untere Schranken für numerische Wertebereiche, zugeordnet werden.

In der unteren Modellschicht wird dargestellt, dass die Dienste durch Server-Prozesse und die Subjekte durch Client-Prozesse implementiert werden, welche auf Netzknoten angesiedelt sind. Das System aus Knoten, Vernetzung, Prozessen und ihrer Kommunikationsinfrastruktur wird hier weiterhin um konkrete Management-Komponenten ergänzt (d.h., z.B. Firewalls, Verzeichnisse, VPN-Konzentratoren, Watchdog-Prozesse, Zugriffskontrollkomponenten). Die Management-Policies haben hier den Charakter von Low-Level-Policies, welche direkt den Steuerparametern der Enforcement-Komponenten entsprechen (z.B. Access Control Lists in Paketfilter-Knoten).

Durch die Modellierung von System und Anforderungen auf drei unterschiedlichen Schichten können die Policies auf mehreren Ebenen diskutiert und geprüft werden. Durch die automatisierte Ableitung der Low-Level-Policies werden Verfeinerungsfehler vermieden.

2. Laufzeit Management & Policy Durchsetzung

Die im ersten Schritt mit dem Modellierungstool erstellten Management-Policies werden von einer Menge von Management-Diensten zur Laufzeit durchgesetzt und überwacht. Die entwickelte Infrastruktur besteht aus fünf leichtgewichtigen Diensten (siehe auch Abbildung 1). Die dargestellte Aufteilung soll eine gute Skalierbarkeit und Wiederverwendbarkeit der Dienste ermöglichen.

Der *Deployment Service* wird zur Initialisierung und zur Verteilung der Konfigurationsdaten an die Management-Dienste verwendet. Der *Monitoring Service* überwacht ihm zugeteilte Dienste unter Nutzung von Polling oder Eventing Mechanismen. Sollten Änderungen an den überwachten Parametern auftreten, werden diese dem *Environment Element Activation Service* gemeldet. Dieser hält eine Liste der abstrakten Systemparameter , den *Environment Condition Elements,* vor, die auf der mittleren Modellebene zur Definition des Systemzustands verwendet wurden. Diese werden anhand der im Modell definierten Einschränkungen und den neuen Parameterwerten re-evalutiert. Ändert sich der Zustand von mindestens einem Element, leitet

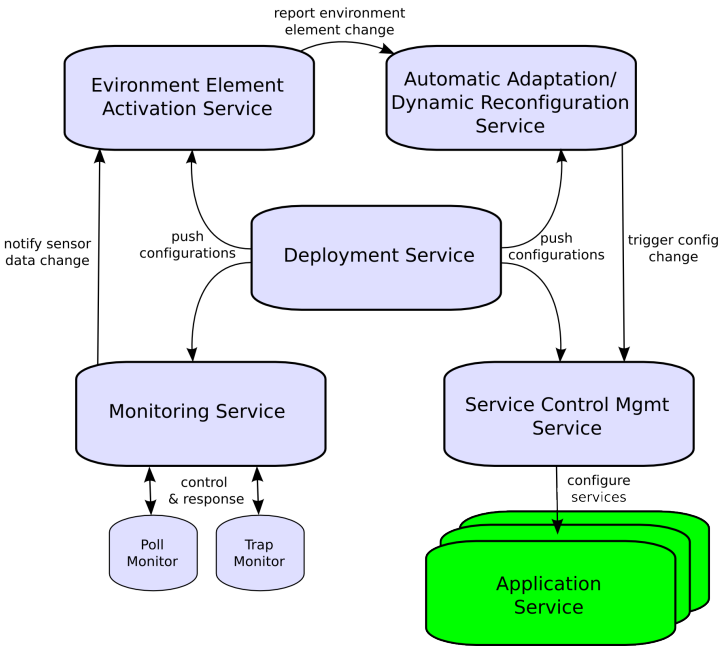

Abbildung 1: Management-Dienste

der *Environment Element Activation Service* die Liste aller aktiven *Environment Condition Elements* an den *Automatic Adaptation/Dynamic Reconfiguration Service* weiter. Dieser berechnet anhand der aktiven Systemparameter den aktuellen Zustand des Systems bzw. der Umgebung und leitet im Fall eines Zustandswechsels eine Rekonfiguration der verwalteten Applikationdienste ein. Die Definition, welche Zustände bzw. Szenarien für das verwaltete System von Belang sind, legt der Administrator in der Modellierungsphase fest. Die eigentliche Rekonfiguration nimmt der eine bestimmte Menge von Applikationsdiensten verantwortliche *Service Control Management Service* vor.

Dieses System wurde im Rahmen des SIRENA Projektes anhand zweier Demonstratoren erprobt und getestet. Die Implementierung des Tools und der Dienste erfolgte in Java, als SOA wurde *Universal Plug and Play* (UPnP) [5] eingesetzt.

3. Anwendungsbeispiel

In diesem Abschnitt soll ein kurzes Anwendungsbeispiel vorgestellt werden, dass im Rahmen des

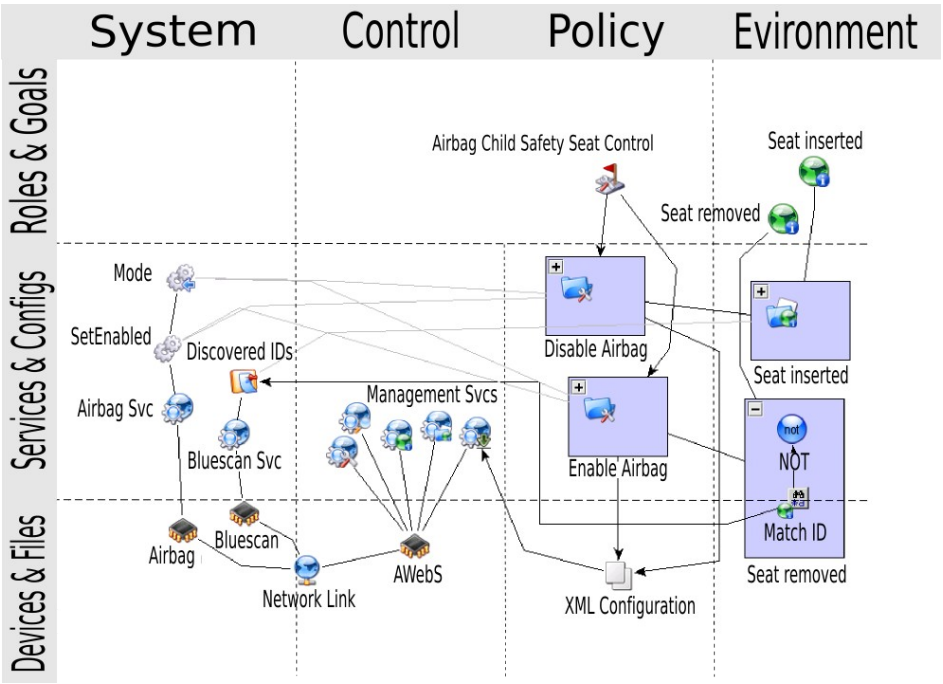

Abbildung 2: Kindersitz Beispielmodell

SIRENA Projektes realisiert wurde. Die Ausrichtung auf serviceorientierte Architekturen spiegelt sich auch im nun vorgestellten Modell wider. In Abbildung 2 ist ein Modellbeispiel zu sehen, wie es in einem Demonstrator implementiert wurde. Dieses Szenario stammt aus der Automotive Domäne. Mit Hilfe eines Bluetooth Scanners und einem mit einem Bluetooth Tag ausgestatteten Kindersitz wird die automatische Abschaltung des Beifahrerairbags bei Einbau des Kindersitzes auf dem Beifahrersitz simuliert. Der Bluetooth Scanner befindet sich im Fahrzeug, ebenso wie die eingebettete Plattform [4], auf der unsere Management-Dienste installiert sind. Das Modell in Abbildung 2 definiert zwei verschiedene Szenarien: Ein Szenario, in dem der Kindersitz eingebaut ist, und der Airbag deaktiviert werden muss, und das entsprechende Gegenstück, die Aktivierung des Airbags bei Abwesenheit des Kindersitzes. Auf der obersten Ebene des Modells sind im Beispiel nur insgesamt drei Elemente zu finden: Ein abstraktes Managementziel „Airbag Child Safety Seat Control“ und zwei Umgebungsrollen, die die unterschiedlichen Umgebungen (Sitz eingebaut, Sitz nicht eingebaut) repräsentieren. Auf der mittleren Ebene sind auf der rechten Seite des Modells die Verfeinerungen der Umgebungsrollen zu sehen. Bluetooth Ids sind ähnlich wie Mac-Adressen aufgebaut und bestehen aus einem Tupel von sechs Bytes. Die Entscheidung, ob der Kindersitz eingebaut ist oder nicht, hängt davon ab, ob der Bluetooth Scanner die Id des Kindersitz Tags scannen kann. Die Verfeinerungen der Umgebungsrollen definieren in Abhängigkeit von

dieser Tatsache, die beiden notwendigen Szenarien. Die Umgebungen sind mit den Konfigurationsanweisungen in der „Policy“ Spalte verbunden, die in der speziellen Situation aktiviert werden müssen. Auf der linken Seite des Modells ist die Systemmodellierung zu finden. Da keinerlei Policies für eine Zugriffskontrolle definiert werden mussten, ist in der „System“ Spalte die oberste Schicht leer. In der mittleren Schicht finden sich die Anwendungsdienste, der „Airbag Service“ für die Kontrolle des Airbags, und der „BlueScan Service“ des Bluetooth Id Scanners. Die Dienste laufen auf verschiedenen Geräten, die durch eine Netzwerkverbindung miteinander verbunden sind. Die für das Laufzeit Management verantwortlichen Dienste sind in der Mitte des Modells dargestellt. Diese laufen auf der AWebS Plattform und sind über eine Netzwerkverbindung mit den Applikationsdiensten verbunden. Aus dem Modell werden schließlich XML Konfigurationsbeschreibungen für die Management-Dienste generiert, die dann über den *Deployment Service* installiert werden.

Das vorgestellte Anwendungsbeispiel ist sehr einfach und speziell. In der endgültigen Anwendung müssten zudem weitere Sensoren in das Auto integriert werden, da das Scannen von Bluetooth Ids zu unzuverlässig ist und der Airbag auch bei einem im Fond installierten Sitz deaktiviert werden würde.

4. Ausblick

Zum Einen wollen wir unser Konzept in erheblich komplexeren Situationen testen und evaluieren. Zum Anderen ist unser Ziel für die Zukunft die Realisierung eines Modells, mit dem man im Bereich serviceorientierter Architekturen in der Lage ist, die Planung und Verwaltung von allgemeinen Management-Anforderungen (z.B. Verfügbarkeit, Durchsatz) durch allgemeine *Management Pattern* zu ermöglichen. Diese Pattern sollen so konzipiert werden, dass nach Modellierung des bestehenden Systems abstrakte Anforderungen eingefügt werden können und dann durch definierte Ableitungsregeln in einem (halb-)automatischen Prozess die Realisierung der Anforderungen geplant werden kann. Die Planung umfasst nicht nur die Ableitung der Konfigurationsbeschreibungen für die Laufzeit-Dienste, sondern auch die Erweiterung des Modells um für die Durchsetzung der abstrakten Anforderungen notwendige Elemente, wie z.B. Watchdog Prozesse oder zusätzliche, redundante Instanzen von Diensten.

Referenzen

[1] SIRENA, *Service Infrastructure for Realtime Embedded Networked Applications*, http://www.sirena-itea.org, 2006.

[2] S. Illner, A. Pohl, H. Krumm: *Automated Runtime Management of Embedded Service Systems Based on Design-Time Modeling and Model Transformation*. in Proc. 3rd IEEE Int. Conf. on Industrial Informatics (INDIN05), Perth, Australia, IEEE Computer Society Press, IEEE Catalogue Number: 05EX1057C, Paper Number PD-001854, 2005.

[3] João Porto de Albuquerque, Heiko Krumm, Paulo Licio de Geus: *Policy Modeling and Refinement for Network Security Systems*. in Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, 2005, Stockholm. Los Alamitos, California : IEEE, 2005. p. 24-33.

[4] AwebS, *Automobiles Web System*, http://www.awebs.de/, 2006.

[5] UPnP Forum, http://www.upnp.org, 2006.