Domain Specific Software Engineering: Internet of Things

Seminar Domain Specific Software Engineering

Version 0.1

Zürcher Hochschule für Angewandte Wissenschaften

Daniel Brun

xx. Juni 2015

Eigenständigkeitserklärung

Hiermit bestätigt ich, dass vorliegende Seminararbeit zum Thema „Evaluation einer Mini ERP Lösung für einen Verein“ gemäss freigegebener Aufgabenstellung ohne jede fremde Hilfe und unter Benutzung der angegebenen Quellen im Rahmen der gültigen Reglemente selbständig verfasst wurde.

Thalwil, 11. Februar 2015 Daniel Brun

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Hintergrund

Im Rahmen meines Bachelor-Studiums in Informatik an der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) muss im 6. Semester eine Seminararbeit zu einem vorgegebenen Themenbereich erarbeitet werden. Ich habe mich für den Themenbereich „Domain Specific Software Engineering“ entschieden.

Aus einem Themenkatalog konnte ein spezifisches Thema im Bereich „Domain Specific Software Engineering“ ausgewählt werden. Ich habe mich für das Thema „Internet of Things“ entschieden.

Für die Arbeit sollen circa 50 Arbeitsstunden aufgewendet werden. Dies entspricht etwa einem Umfang von 15 bis 20 Seiten. Zusätzlich gelten die Rahmenbedingungen gemäss dem Reglement zur Verfassung einer Seminararbeit ([**ZHAW:2012:Seminararbeit:Reglemente**])

1.2 Aufgabenstellung

Es soll ein Dokument zum Thema Domain Specific Software Engineering im Bereich Internet of Things erstellt werden. Das Papier soll die Schwierigkeiten der Software-Entwicklung in diesem Bereich aufzeigen und einen groben Überblick über das Thema eben.

1.3 Abgrenzung

Abgrenzung

1.4 Motivation

motivation

1.5 Struktur

Diese Arbeit gliedert sich in folgende Hauptteile:

-

Struktur

Im ersten Kapitel werden die Details zur Ausgangslage und die Hintergründe der Arbeit aufgezeigt. Im zweiten Kapitel wird mit Hilfe einer Umfrage innerhalb des Turnvereins eine Analyse erstellt. Aus dieser Analyse gehen die Randbedingungen, Ziele und Anforderungen an das Mini System hervor. Diese Randbedingungen, Ziele und Anforderungen werden im Kapitel 'Evaluation' als Kriterien für die Vorselektion, Selektion und anschliessenden die Evaluation der Produkte verwendet. Im letzten Kapitel wird ein Fazit gezogen, eine Empfehlung an den abgegeben und über die gesamte Arbeit reflektiert.

KAPITEL 2

Ausgangslage

Mit dem laufenden Fortschritt in der Computer- und Kommunikationstechnik und der damit einhergehenden Miniaturisierung und Mobilisierung eröffnen sich immer wieder neue Bereiche in der Informatik. Eines dieser neuen Gebiete wird als „Internet of Things“ bezeichnet.

Mit dem Fortschritt in der Computertechnik und der fortlaufenden Miniaturisierung und Mobilisierung kommt das IOT

Allgemeins Gadgets, IOT

2.1 Internet of Things

Das Internet of Things (IOT), beziehungsweise das „Internet der Dinge“, Nächste Etappe in der durch das Internet ausgelösten REvolution wie Leute kommunizieren und miteinander arbeiten

Dinge Kommunikation mit realer Welt, Dinge müssen zusammenarbeiten, Geschwindigkeit, Skalierung und Fähigkeiten

Vernetzte Stadt –> Beispiel

Veränderung, womöglich mehr, als das human centric Internet

Anzahl Smartphones aktuell? entwicklung? Potenziall für IOT?

Device2Device Kommunikation, Server2Server Kommunikation,

http://de.slideshare.net/RealTimeInnovations/io-34485340

http://www.internet-of-things.eu/

Die „Things“, Dinge, beziehungsweise Geräte -Embedded Devices / -Systems, Kern: Bau vernetzter Produkte,

Transformation von isolierten Systemen zu vernetzten dingen,

Ziel von IOT-Lösungen (z.B. mit Cloud Power): Vernetzung von Millionen Geräten, Teilen, Analyzieren, Schlüsse ziehen (Big Data),

IOT-Lösungen: Verbesserung medical outcome, schneller und bessere Produkte, Entwicklungskosten reduzieren, besseres Shopping vergüngen, Optimierung Energieerzeugung und -konsum - System: Verbessert für Datensicherheit , -datenschutz, mangement von Geräten, Data analyitcs

End-To-End Lösung: Orchestrierung der einzelnen komponenten, Grafik Seite 10 (Intel White Paper)

Eingebaute, sichere Kommunikation

IOT: Things - Gateway - Network and cloud, grosses ökosystem, regulatorische / rechtliche Schwierigkeiten

IOT: Thing - Local Network (Evtl. mit Gateway)- Internet - Back-End Services (Enterprise Data System, PC, Mobile Device)

Dinge: Autos, Geräte-Sensoren, Wearables, Smartphones, direkt verbunden über Mobiles Netzwerk mit Zugriff auf das Internet, IOT-Solution: entweder Dinge sind intelligent –> filtern / managed von daten lokal oder verbunden mit gateways, welche diese Fkt bieten.

Gateway: 85 % der Geräte nicht darauf ausgelegt, direkt mit Internet zu kommunizieren (Quelle: Intel: Developing Solutions for IOT White Paper), Gateway als intermediär zwischen legacy things und cloud und anbietung verbindung, sicherheit, management

Network and Cloud: klassisch, Datenanlyse der Rohdaten

2.1.1 Anwendungsbereiche

Unterschied: http://micrium.com/iot/iot-rtos/

Für das IOT gilt es zwei Anwendungsbereiche zu unterscheiden. Auf der einen Seite gibt es den Anwendungsbereich in der Industrie...(Verschiedenste Technologien, IP-Network -> internet), Wireless Sensor Network: Distributed Sensors, montiro physical / environmental conditions, WSN-Node (low cost, low powered, batterie): Einzelne Funktion

WSN Edge Nodes: WSN Node mit Internet Verbindung, Gateway WSN und IP-Netzwerk, lokale Verarbeitung, Speicher, evtl. User-Interface

Auf der anderen Seite existiert der Anwendungsbereich des Kommerziellen IoT (Bluetooth, Ethernet (Wired or Wireless), only with local devices.)

WSN-Technologien: Wi-Fi (hohe Verbreitung, hoher Stromverbrauch), Low-Power-Solutions (Low-Power and efficient radios, energy harvesting, mesh networking, long term operation, new protocols and data formats), IEEE 802.15.4: Radio Standard, base low power system, power reduction, 6LoWPAN: As small messages as possible, IPv6 over Low power Wireless Personal Area Networks, Kapselung und Kompressionsmechnismen für kürzere Übermittlungszeiten

2.1.2 Das „Ding“

Definition „Thing“ nich einfach, z.B. embedded computing device / system welches informationen über ein Netzwerk versendet und empfängt

Embedded Systems: Basis: Mikrocontroller, kleiner Memory foot print,

Stromsparen: Früher so rasch als möglich Aufgabe ausführen, anschliessend in Sleep mode wechseln, neue Prozessorarchitekturen: Fast kein Stromverbrauch, Nachteil: weniger Performance,

Plug And play, Paradigmenwechsel: Aktuell Server stellen Daten zur Verfügung,

-Low-Powered-Device -Embedded OS -Kommunikationsmöglichkeit (meist: wireless) mit 1:n Kommunikationsprotokollen -Direkt mit Internet oder Via Gateway -Netzwerk -Server / Infrastruktur -Analytics -End-User-Tier

2.2 Domain Specific Software Engineering

2.3 Domain Specific Languages

Eine „Domain Specific Language (DSL)“ bezeichnet eine Programmier-, Modellierungs- oder Metasprache, welche für eine spezifische Domäne entworfen und entwickelt wurde. Eine DSL addressiert dabei spezifische Schwächen und Problemstellungen der angesprochenen Domäne.

KAPITEL 3

Die Domäne „Internet of Things“

3.1 Einführung

Beschreibung der Domäne

–

3.2 Anforderungen an Software

Stark von Einsatzgebiet abhängig (Kühlschrank vs. Low Energy ohne Stromversorgung)

3.3 Anforderungen an die Softwarearchitektur

3.4 Anforderungen an die Programmiersprache

—-

3.5 Anforderungen / Unterschiede zur „Standard-Domäne“

In diesem Kapitel werden für die einzelnen Abschnitte des Software Engineering Prozesses die Anforderungen, beziehungsweise die Unterschiede, zum Software Engeineering Prozess in der „Standard-Domäne“ aufgezeigt. Mit „Standard-Domäne“ wird in dieser Arbeit Software Engineering im Bereich von Enterprise-Anwendungen bezeichnet.

http://ercim-news.ercim.eu/en98/special/internet-of-things-a-challenge-for-software-engineering
http://link.springer.com/chapter/10.1007

Das der Begriff „Internet of Things“ eigentlich nur ein Oberbegriff ist.... deckt dieser Begriff auch ein sehr groses Spektrum an verschiedensten Anwendungsmöglichkeiten ab.

3.5.1 Software Requirements

eindeutig, Komplexität schwierig, nachhaltig (schwieriger Update)

non-consumer-market: regulatorisches Schwierigkeiten, Remote Tracking, Kabellose Implantate, –> Requirements-Phase: Verfolgbarkeit und Prüfbarkeit, Geschickte Verisonierung, Reviews –> Sicherstellung Compliance und Erfolg Wetter, Physische Einflüsse, Durch das Gerät erzeugte Hitze, Lange Lebensdauer

SW Muss: Skalierbar für breite Palette an verschiedenen Gerätekategorien Modular sein –> nur Auswahl (RAM-Footprint) Verbunden (Daten rein/raus) Verlässlich:Zertifizierung für kritische Applikationen

3.5.2 Software Design

Schlank, Energieeffizient, einfach wartbar?

Berücksichtigung: Protokolle, Standards, Last-Anforderungen

Machine To Machine Connectivity, Wireless, API-Evolution in MInd

3.5.3 Software Construction

Je nach Anwendungsbereich: Hardwarenahe, Optimierung Für die Entwicklung von Anwendungen für IOT-Geräte können https://www.rti.com/company/careers/software-engineer.html grundsätzlich

Firmware für Spezifische Hardware, Adressierung Netzwerk- und Verbindungsprobleme, Security

3.5.4 Software Testing

Wenn bei Consumer: Update evtl. schwierig, Fehlfunktion schwerwiegend,

Test: Nachbildung physische Umgebung, komplexe Szenarien, Netzwerk-Anforderungen,

3.5.5 Software Maintenance

Schwierig,... Immer connected –> regular updates, wenn nicht regelmässig upgedatet –> Verlust kritischer Funktionalität, Continuous Delivery

Nicht überall möglich, Low-Bandwith, Hardware-Near Devices

3.5.6 Software Configuration Management

3.5.7 Software Engineering Management

3.5.8 Software Engineering Process

Defect Tracking, small scale projects, fast product turnaroudn,

3.5.9 Software Engineering Tools and Methods

3.5.10 Software Quality

Höhere Qualität notwendig

Layer: Transport-Layer: TCP zum Teil overkill für IOT-Device –> UDP UDP: beser geeignet für relatime-data, TCPS Acknowledgment and retransmission unnützer overhad für solche Anwendungen (Stück sprache nicht rechtzeitig übermittelt -> Retransmission sinnlos),

Klassische Ansätze nicht alle geeignet für IoT,

Service-based Application: composition and orchestration of Services

KAPITEL 4

Software Engineering in der Domäne „Internet of Things“

Evtl. Struktur nach kategorie: Connectivity, Management, Security, API, ...

Real-Time schwierig, trade off datenverlust, oft TCP als Datenbasis -> messung mit Quality of service besser

http://postscapes.com/internet-of-things-software-guide https://www.silabs.com/Supporthttp://www.datamati
source/35-open-source-tools-for-the-internet-of-things-1.html https://blog.profitbricks.com/top-
49-tools-internet-of-things/ http://www.businesscloudnews.com/2015/05/14/samsung-announces-
open-internet-of-things-platform/ https://www.rti.com/company/news/iot-connectivity-
webinar.html http://www.iotsworldcongress.com/documents/4643185/4c3cc80c-03b0-41be-
baf0-6a1a0fa7db07

SWE: https://hal.inria.fr/hal-01064075/document http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7
http://www.cio.com/article/2843814/developer/how-to-develop-applications-for-the-internet-
of-things.html http://www.appdevelopersalliance.org/internet-of-things/ http://www.intel.com/content/www/u
of-things/white-papers/developing-solutions-for-iot.html http://link.springer.com/chapter/10.1007

4.1 Programmiersprachen

C, C++, Zum Teil: Java –> OS benötigt, von daher C,C++ und Java Java benötigt nicht zu vernachlässigendd Ressourcen, z.B. Oracle Java ME Embedded: Memory: 130 KB - 700 KB, ROM: 350 KB - 2000 KB, + Netzwerkstack, Kernel, nicht mehr ein embedded system, Java nur auf fähigeren, teurerern Systemen

4.1.1 Assembler

Eine Implementation in Assembler kann unter bestimmten Voraussetzungen die beste Lösung für ein bestimmtes Problem sein. In der Regel wird der Assembler-Ansatz gewählt, wenn das Programm so effizient und sparsam wie möglich ablaufen und mit so wenig Ressourcen als möglich auskommen soll.

4.1.2 C / C++

4.1.3 Java / .NET

Für die Implementation kann es durchaus sinnvoll sein eine Hochsprache, wie Java oder .NET C# einzusetzen.

4.2 Protokolle und Standards

Grafik: http://electronicdesign.com/embedded/understanding-protocols-behind-internet-things

Viele Implementationen: MQTT: Collect device data, send to server, D2S CoAP XBEE XMPP: Access device data, connect device to people, D2S DDS: Distribute Device Data, fast bus for integrating intelligent machines (D2D) HTTP AMQP (IOT-Client-Seitig?), Queuing system to connect servers to servers (S2S)

https://www.sparkfun.com/news/1705 http://micrium.com/iot/internet-protocols/

4.2.1 MQTT

Message Queue Telemetry Transport, Device Data Collection, Haupfaufgabe: Fernmessung, Remote Monitoring, Datensammeln und an Infrastruktur ausliefern, Anwendungszweck: Grosse Netzwerke von kleinen Geräte, welche überwacht und kontrolliert werden müssen. Basis: TCP (kein Datenverlust)

Publish / Subscribe, central Server, Subscribe to Topics, MQTT-Broker, Notify all connected devices, Hub and Spoke System: Geräte –> Data Verbindung zu Server, System ist designt, um daten an enterprise technologien weiter zu geben (z.B. ESB)

Kein Protokoll für D2D, nicht mehrere Empfänger der Daten, wenige Kontrollmöglichkeiten, muss nicht schnell sein –> kein rela time (sekunden)

Ziel: Conserve Power and Management Bsp: ÖL Pipeline, Energieverbrauch Überwachung, Licht-Kontrolle

Lightweight packet structure, dokumentation auf einer Seite

4.2.2 XMPP

Ursprünglich: Jabber, entwickelt für Instant messaging Extensible Messaging and Presence Protocol

Text-Kommunikation zwischen Punkten

XML, über TCP (oder HTTP over TcP?)

Stärke: Adressierungs-Schema: name@domain.com, Security, Skalierbarkeit –> Ideal für Consumer-Oriented applications

Einfacher Weg Gerät zu adressieren, nicht schnell –> Polling oder Check for Updates on demand,

Einsatz: z.B. Heimautoation –> Thermostat mit Web verbinden

4.2.3 DDS

Data Distribution Service, Geräte welche direkt Geräte-Daten verwenden, Verteilung zu anderen Geräten, Interaktion mit Infrastruktur unterstützt, Daten-zentrierter Middleware-Standard, Wurzeln: High-Performance Verteidigung, Industrie, Embeddedd Applications, effektiv: millionen von nachrichten pro sekunde zu mehreren gleichzeitigen empfängern

Publish, Subscribe Architecture

Unterschied: Daten an Infrastruktur oder an anderes Gerät, Geräte sind schnell, mit vielen Geräten kommunizieren, TCP Point to Point to restriktiv, DDS: Detailierte qualitiy of service conrol, multicast, konfigurierbare verfügbarkeit, Redundanz

Filter und Selektion von Daten, bzw. Bestimmung was wohin geht

Direct Device-to-device bus, relational data model, ähnlich Datenbank,

Z.B.: Militär, Windparks, Asset-Tracking, Fahrzeug Test und Sicherheit

4.2.4 AMQP

Advanced Message Queuing Protocol, ab und zu: IOT-Protokoll, transaktionsbasierte Nachrichen zwischen Servern, message-centric-middleware, proces thousands of reliable queued transactions, Fokus: kein Nachrichtenverlust, Publishers –> Exchanges, queues to subscribers: TCP, acknowledge acceptance of message, optional transaction mode with formal multiphase commit sequence

Hauptsächlich: Business messaging, device - back-office data centers

IOT: appropriate for control plane oder server-basierte analyse funktionen

4.2.5 HTTP

Basis für Client-Server-Model im Web, Sichere Methode: auf Device nur Client implementiersen

4.2.6 WebSocket

Protokoll, Full-Duplex Communication über einzelne TCP-Verbindung zwischen Client und Server, Zeilt HTML5 Spec

4.2.7 CoAP

Web-Protokolle z.T. zu schwer für IOT-Geräte, Constrained Application Protocol for low-power and constrained networks, RESTful protocol, semantisch an HTTP ausgerichtet, 1:1 Mapping von / zu HTTP

UDP, + einige nachgebildete TCP-Funktionen, Request / Responses: Asynchron via CoAP Messages, Header, Method, Status Codes binary encoded, Reduktion Overhead, Caching abhängig von Response-Code (HTTP: Request Method)

Comparison of Protocol-Stacks: http://micrium.com/iot/internet-protocols/

Leichtgewichtige Alternative zu HTTP, CoAP Packets: all around bitmapping

4.2.8 Thread

Protokolle: Thread, Netzwerk-Protokoll, Fokus: Security, Low Energy, notwendiger Chip, schon in vielen Geräten vorhanden, gestützt auf 6LoWPAN, IPv6 over Low power Wireless Personal Area Network,

4.2.9 AllJoyn

Qualcomm entwickelt, anschliessend: Linux Foundation, AllSeen Alliance (Cisco, Microsoft, LG, HTC, ...) Verbindung, Wartung Geräte in WLAN-Netzwerk, Kontrolle, Benachrichtungs Service,

4.3 Frameworks

In diesem Kapitel werden einige Frameworks vorgestellt, welche ....

Frameworks –> mobile / web –> Vereinfachung Entwiclung, Abstraktion Implementations-Details, apache ISIS? http://iot.eclipse.org/java/open-iot-stack-for-java.html

4.3.1 Kommunikation zwischen IOT-Geräten

AllSeen IOTIVITY Z-Wave

6LoWPAN, ANT, Bluetooth, DASH7, ISA100, Wireless HART, Wireless M-Bus, Z-Wave, Zigbee / Zigbee IP, EnOcean, EtherCAT, Modbus, Profinet, HomePlug, HomeGrid

Voraussetzung Zugriff Internet: IPv6 Pflicht, Non-IP Geräte; Gateway

Middleware / Platforms / OS: http://postscapes.com/internet-of-things-software-guide

Weitere-Protokolle: http://postscapes.com/internet-of-things-protocols Tools: http://www.datamation.com/source/35-open-source-tools-for-the-internet-of-things-1.html

KAPITEL 5

Case-Study „iotivity

5.1 Aufsetzen der Entwicklungsumgebung

5.2 Entwicklung

5.3 Deployment und Betrieb

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der noch zu erledigenden Punkte