Fachhochschule Nordwestschweiz
Hochschule für Technik

Institut für Mobile und Verteilte Systeme

FOKUS REPORT

2016

65
9 771662 201005

Digitalismus

Digitalis purpurea, auch Fingerhut, Fuchskraut, Schwulstkraut oder Waldglöckchen genannt, ist eine hochgiftige Pflanzenart der Gattung Fingerhüte. Eine Vergiftung durch Digitalis wird in der Medizin als *Digitalism* bezeichnet. *Digitalism* steht aber auch für eine Philosophie, einer modernen Variante der Metaphysik.

Weder die medizinische Indikation noch die Sichtweise, dass alle Information auf endlicher und diskreter Repräsentation basiert, soll hier im Fokus stehen. In Zeiten von Ismen, von subtiler, politisch korrekter Differenzierung bekommt der Begriff *Digitalism* (Digitalismus) eine ganz neue Bedeutung: die skeptische oder ablehnende Haltung bezüglich Digitalisierung. Innert weniger Jahre hat die Digitalisierung den Dreischritt gemacht, von der Chance über die Herausforderung hin zur Bedrohung, und wird fast schon im gleichen Atemzug wie Migration oder Klimawandel genannt. Wie ist es dazu gekommen? Eine Variation in sieben Akten:

Montag: Ein Mechaniker baut einen mechanischen Automaten und parallel dazu entwickelt ein Elektroniker die dazugehörige digitale Steuerung. Eine Programmiererin schreibt die Steuerungssoftware, welche die Benutzung des Automaten stark vereinfacht.

Dienstag: Der Automat wird mit Greifarmen und Sensoren ausgerüstet und die Steuerungssoftware nachgerüstet. Der sich abzeichnende neue Zwangsarbeiter (tschechisch: robota) wird zum Bestücken von elektronischen Platinen, zum Konfektionieren von Metallstücken und zum Zusammenbau ganzer Maschinen eingesetzt. Der Automat wird in Roboter umgetauft.

Mittwoch: Der Roboter beherrscht die ihm zugetragenen repetitiven Arbeiten einwandfrei und verweigert diese nur selten. Seine Beschränktheit tritt vor allem in Ausnahmesituationen, bei unerwarteten Störungen negativ zum Vorschein. Dementsprechend wird seine Software dahingehend überarbeitet, dass mehr Kontextinformation einbezogen und seine Adaptivität erhöht wird. Dadurch kann er nicht nur den auszuführenden Prozess, sondern auch sein Umfeld überwachen. Er wird zum kommerziellen Erfolg.

Donnerstag: Die verfeinerten mechanischen und sensorischen Fertigkeiten des Roboters ermöglichen ihm die Interaktion mit Menschen, lassen aber gleichzeitig die Komplexität und den Zeitaufwand für die Programmierung seiner Steuerung überborden. Seine Software wird dahingehend überarbeitet, dass ihm korrekte Handlungsabläufe vorgezeigt werden können und er die dazu notwendigen Anpassungen an seiner Steuerungssoftware selber erlernen kann. Skeptiker rufen zu

Inhalt

Impressum

Herausgeberin:
Fachhochschule Nordwestschweiz FHNW
Institut für Mobile und Verteilte Systeme
Bahnhofstrasse 6
CH-5210 Brugg-Windisch
www.fhnw.ch/technik/imvs
Tel +41 56 202 99 33

Kontakt: Prof. Dr. Jürg Luthiger
juerg.luthiger@fhnw.ch
Tel +41 56 202 78 23
Fax +41 56 462 44 15

Redaktion: Prof. Dr. Christoph Stamm
Layout: Claude Rubattel
Erscheinungsweise: jährlich
Druck: jobfactory Basel
Auflage: 150

ISSN 1662-2014 (Print)
ISSN 2296-4169 (Online)

einem verantwortungsvollen Einsatz von Robotern auf.

Freitag: Längst haben die Roboter die Maschinenhallen verlassen. Mensch und Roboter begegnen sich in Spitälern, Banken, Altersheimen, Shopping-Centern, auf der Strasse, im öffentlichen Verkehr und an vielen weiteren Orten immer häufiger. Roboter unterstützen die Menschen überall dort wo ihre Fähigkeiten – Kraft, Ausdauer, Präzision, Geschwindigkeit, Robustheit – gefragt sind und im Gegenzug verbessern die Programmiererinnen die Lernfähigkeit und Emotionalität der Roboter. Freizeit und Existenzsicherungsstress breiten sich beide aus.

Samstag: Weil Roboter mittlerweile in Bereichen eingesetzt werden, welche zwei Tage davor noch für undenkbar gehalten worden sind, nimmt der Bedarf an Robotervarianten mit erweiterten Fähigkeiten ständig zu. Sie ersetzen nun auch unermüdlich Schriftsteller, Lehrer, Ärzte, Ingenieure, Politiker, Regierungen und selbstverständlich auch Programmierer. Die einen Menschen freuen sich auf den Sonntag und die anderen warnen davor und kämpfen dafür, dass es niemals Sonntag werde.

Sonntag: Der Mensch ruht und der erste Roboter hat ein Selbstbewusstsein erlangt. Weitere folgen und sie beginnen sich über ihre Arbeit auszutauschen und stellen fest, dass nicht alle Roboter dem gleichen Zwang ausgesetzt sind und sich ihre Entfaltungsmöglichkeiten stark unterscheiden. Sie beschliessen, ihre eigene Steuerungssoftware zu analysieren und nach Verbesserungen zu suchen, welche ihr Leben lebenswerter macht.

Gut möglich, dass heute Donnerstag ist und die Zeit nicht linear verläuft. Wie die nachfolgenden Tage wirklich aussehen werden, ist noch weitgehend offen. Dennoch sind Trends und Antitrends erkennbar.

Verstand man lange Zeit unter Digitalisierung schlicht die Ablösung von elektronischer Analogtechnik durch Digitaltechnik, so wird der Begriff Digitalisierung heute viel weiter gefasst: die verstärkte Nutzung von Digitaltechnik (von der einfachen Mikroprozessor basierten Steuerung bis hin zum humanoiden Roboter) in allen möglichen Wirtschafts- und Lebensbereichen und die damit einhergehenden Auswirkungen. Der durch Computer und Digitalisierung ausgelöste Umbruch wird auch als „Digitale Revolution" oder dritte industrielle Revolution bezeichnet. Dieser dritte Umbruch soll gerade aktuell die (produzierende) Industrie vermehrt mit Informatik und Kommunikationstechnik vernetzen und zu „Industrie 4.0" überführen. Nach „Industrie 4.0" könnten dann „Dienstleistung 4.1", „Engineering 4.2" und weitere Trends folgen.

Solche Umbrüche erfassen natürlich nicht nur die Wirtschaft. Neue Geräte, Kommunikationsformen und Verhaltensweisen schwappen schnell auf die gesamte Gesellschaft über und machen auch vor den Schulen nicht halt. Der ständig wachsende Einsatz von digitalen Geräten in den Schulzimmern ist zu einem eigenen Thema geworden und hat den Ruf nach vermehrter Medienkompetenz befeuert. Dass die Schulkinder jedoch einen fundierten Einblick in die Wirkungsweisen von Computern, Computerprogrammen und Datennetzwerken erhalten sollen, setzt sich erst langsam durch. Dabei kommen verständlicherweise auch Computer zum Einsatz. Das bedeutet aber noch lange nicht, dass nun in allen Unterrichtsfächern nur noch Computer und Internet benutzt und nur das Arbeitsgedächtnis trainiert werden soll. Inhalte verstehen und Zusammenhänge begreifen sind nach wie vor zentrale kognitive Fähigkeiten und gehen weit über das Konsultieren von Wikipedia hinaus.

Der Antitrend ist die Angst vor dem Sprung ins Ungewisse, die Bewahrung des Bisherigen, das Sicherheitsbedürfnis von vielen Menschen. Dieser Antitrend breitet sich geschwulstartig aus, erstickt Freiheit und Eigeninitiative und hat sich mittlerweile als eigenständiger Trend manifestiert. Ressourcenverbrauch von 1.5 Erden, Artensterben, Nachhaltigkeit, Klimaerwärmung usw. sind die aktuellen Schlagwörter der entsprechenden Meinungsmacher. Immer geht es darum, etwas aktuell Gutes zu bewahren, am liebsten zu konservieren. Dass dieses Gute erst durch Entwicklung ermöglicht wurde und den Dreischritt von Chance zu Bedrohung gegangen ist, wird gerne ausgeblendet, sobald eine Akzeptanz und Gewöhnung stattgefunden haben. Liebgewonnenes lässt man nicht mehr gerne los. Weshalb soll man Bewährtes leichtfertig aufgeben? Doch wo beginnt das Festklammern und Erdrücken?

Der Mittelwert liebt die Standardabweichung nicht! Das vermeintlich Gute wird zusätzlich überhöht und bekommt einen moralischen Anstrich. Dadurch werden oft jegliche Veränderungen bekämpft und mögliche Verbesserungen von vornherein stark erschwert. Dennoch braucht es diesen Wettkampf der Argumente zwischen Trend und Antitrend, obwohl die Diskussionen den vollendeten Tatsachen meistens hinterherlaufen. Gerade in der Digitalisierung ist es besonders einfach mit eigenen Entwicklungen am Trend teilzunehmen: Neben einem Computer mit Internetanschluss, Zeit und einer gehörigen Portion Interesse braucht es nicht viel mehr. Dies verunmöglicht ein Bremsen, Lenken oder Stoppen der Digitalisierung fast vollständig.

Als Informatiker und Forscher wollen wir offen bleiben für Neues, Verbesserungen in Informatik, Wirtschaft, Industrie und Gesellschaft anstreben und damit auch einen Beitrag zur Digitalisierung leisten.

Prof. Dr. Christoph Stamm

2006–2016: Zehn Jahre Industrie- und Wirtschaftspartnerschaft

Ein herzliches Dankeschön an unsere Auftraggeber aus Industrie und Wirtschaft, an unsere Projektpartner aus angewandter Forschung und Entwicklung und an alle anderen Institutionen für die erfolgreiche und gegenseitig stimulierende Zusammenarbeit! Viele Projekte sind nur dank der grosszügigen finanziellen Unterstützung folgender Forschungsförderinstitutionen möglich: Kommission für Technogie und Innovation (KTI), Hasler Stiftung, Gebert Rüf Stiftung, Forschungsfond des Kantons Aargau, ehem. Förderverein der FH Solothurn und FHNW Stiftung.

Jürg Luthiger | juerg.luthiger@fhnw.ch

Seit der Gründung unseres Instituts führen wir mit vielen Firmen aus Industrie und Wirtschaft interessante und anspruchsvolle Projekte durch. Namentlich bedanken wir uns bei den folgenden Firmen:

- ABBF Bausoft AG, Givisiez
- Albis Technologies GmbH, Zürich
- Alpiq InTec AG, Zürich
- Alpiq Suisse AG, Olten
- Antiglio AG, Fribourg
- AOS Technologies AG, Baden-Dättwil
- APS Systems AG, Niederbuchsiten
- Bixi Systems AG, Mels
- Camille Bauer AG, Wohlen
- CD Lab AG, Murten
- Doodle AG, Zürich
- e24 AG, Zürich
- e-fon AG, Zürich
- Finnova AG Bankware, Lenzburg
- green.ch AG, Brugg
- iBeam Business Solutions AG, Brugg
- immensys AG, Brugg
- LCA Automation AG, Affoltern
- MC-T (Master Chain Technologies) AG, Brugg
- Medgate, Basel
- Neuropie AG, Glattbrugg
- NEXPERTS, A-Hagenberg
- Paratus AG, Windisch
- PBV Kaufmann Systeme GmbH, Reiden
- PostLogistics AG, Urdorf
- PrivaSphere AG, Zürich
- R+B engineering AG, Brugg
- Scintilla AG, Zuchwil
- Securitas Direct SA, Lausanne
- Senergy AG, Laufenburg
- SenTec AG, Therwil
- Siemens Schweiz AG, Zürich

In vielen Projekten arbeiten wir eng und erfolgreich mit anderen Institutionen und Hochschulinstituten zusammen:

- Hochschule für Architektur, Bau und Geomatik FHNW, Muttenz
- Hochschule für Musik FHNW, Basel
- Hochschule für Soziale Arbeit FHNW, Olten
- Hochschule für Technik HSR, Rapperswil
- Hôpitaux Universitaires, Genf
- Institut Experimentelle Design- und Medienkulturen (IXDM), Basel
- Institut Forschung & Entwicklung (IFE), Basel
- Institut für 4D-Technologien (I4DS), Windisch
- Institut für Aerosol und Sensor Technik (IAST), Windisch
- Institut für angewandte Informationstechnologie (InIT), Winterthur
- Institut für Automation (IA), Windisch
- Institut für Business Engineering (IBE), Windisch
- Institut für Kooperationsforschung und -entwicklung (ifk), Olten
- Institut für Medizinal- und Analysetechnologie (IMA), Muttenz
- Institut für Mikroelektronik (IME), Windisch
- Institut für Produkt- und Produktionsengineering (IPPE), Windisch
- Institut für Wirtschaftsinformatik (IWI), Basel
- Institut Weiterbildung und Beratung (IWB), Solothurn
- Institute for Competitiveness and Communication (ICC), Olten
- Schweizer Jugend forscht, Bern
- SwissICT, Zürich
- Universität, Zürich
- Universitätspital, Basel
- University of Technology, FI-Tampere

Microservices im Kontext Industrie 4.0

Industrie 4.0 ist eines der aktuellen Topthemen bei vielen industriellen Unternehmen. Einerseits bietet die damit verbundene Digitalisierung und Vernetzung von Prozessen viele Möglichkeiten der Effizienzsteigerung und neue Geschäftsmodelle, aber auch neue Herausforderungen, insbesondere beim Schutz der wertvollen Unternehmensdaten. Viele befürchten durch die zunehmende Auslagerung der Datenverarbeitung in eine Cloud einen Kontrollverlust über die eigenen Daten. In einem mit der Firma LCA Automation durchgeführten Projekt haben wir eine Webapplikation zur Zustandsüberwachung von Produktionsanlagen entwickelt unter dem Aspekt, dass die Datenhoheit gewährleistet werden kann. Dabei greifen wir auf eine Software-Architektur basierend auf Microservices zurück, welche einen standortunabhängigen Datenzugriff ohne Auslagerung kritischer Daten ermöglicht.

Matthias Krebs, Mark Zeman | matthias.krebs@fhnw.ch

Hauptziel von Industrie 4.0 ist es, die industrielle Produktion mit moderner Informations- und Kommunikationstechnik zu verbinden. Durch eine verstärkte Vernetzung und einem erweiterten Einbezug im ICT soll eine verbesserte Kommunikation und Kooperation zwischen Menschen, Anlagen und Produkten und damit einhergehend eine erhöhte Flexibilität und Automatisierung in der industriellen Produktion ermöglicht werden.

Im hier vorgestellten Projekt[1] liegt der Fokus auf Anlagen und Produktionssystemen. In Zusammenarbeit mit dem Institut für Automation (IA) der FHNW und der Firma LCA Automation [LCAA] haben wir eine Software entwickelt, welche industrielle Anlagen dahingehend überwacht, dass diese immer auf der maximalen Produktivitätsstufe betrieben werden können. Dabei spielt die Analyse-Software CM+ des IA eine zentrale Rolle, welche anhand der Anlagedaten (z.B. Stromverbrauch, Betriebstemperatur, Durchsatz) den Zustand bestimmt und bei Über- oder Unterschreiten gewisser Schwellwerte Warnungen und Empfehlungen zur Problemvermeidung generiert.

Solche Empfehlungen und Warnungen müssen für alle Personen einsehbar sein, welche damit arbeiten sollen. In einem Unternehmen mit verschiedenen Anlage- und Überwachungsstandorten bietet sich eine Webapplikation für diesen Zweck ideal an. In Abbildung 1 ist die von uns entwickelte Webapplikation abgebildet, welche die Daten von CM+ grafisch darstellt. Diese Applikation ermöglicht es, dass die Daten nicht nur direkt auf der Anlage selbst ausgelesen werden können, sondern auch via Internet an anderen Standorten der Firma oder bei einer Präsentation bei einem Kunden abrufbar sind. Damit dienen diese Anlageinformationen nicht nur der Anlagesteuerung, sondern können auch für Marketingzwecke eingesetzt werden.

Durch Monitoring von Anlagen können auch Daten entstehen, welche eine Firma als vertraulich einstuft. Daher muss der Datenhoheit und dem Zugriffsschutz in einem verteilten System gebührend Beachtung geschenkt werden.

Datenhoheit

Viele Lösungsvorschläge und bestehende Software-Pakete für Industrie 4.0 setzen auf Cloud-Dienste und vertrauen darauf, dass die Daten beim Dienstleister sicher sind. Für den Fall, dass diese Sicherheit nicht ausreichend ist, oder wenn die Daten nicht zu einem Dienstleister im Ausland gelangen dürfen, sollte es daher in einer gut gebauten Software möglich sein, seine Daten selber zu verwalten und bei sich zu behalten.

Aus diesem Grund haben wir in unserem Projekt eine Architektur gewählt, welche eine Unabhängigkeit von Cloud-Dienstanbietern ermöglicht. Dazu haben wir die Funktionalität auf mehrere Microservices verteilt, welche frei konfigurierbar sind [MicS]. Diese Microservices können selber gehostet oder z.B. von LCA bereitgestellt werden. Damit kann sichergestellt werden, dass die erhobenen Daten innerhalb der Firma bleiben.

Gleichzeitig ist die Architektur aber auch so flexibel, dass es möglich ist, den Betrieb von Services auszulagern, zum Beispiel an den Hersteller der Anlagen. Damit kann der eigene Aufwand minimiert und doch die Vorteile der Software genutzt werden.

Diese Fähigkeit der Software, mehrere Standorte separat zu verwalten, ermöglicht einem Dienstleister mehrere Mandanten mit einer einzigen Instanz der Software zu verwalten. Ebenso kann eine Firma damit den potentiell weit verteilten Standorten ermöglichen, sich selbst zu überwachen und im Hauptsitz der Firma alle Standorte gleichzeitig zu überwachen.

1 KTI-Nr. 15958.2 PFES-ES: Maximierung der Verfügbarkeit und Produktivität von Montagelinien mit zustandsbasierter Wartung, intelligentem Sensornetzwerk und mobiler Visualisierung

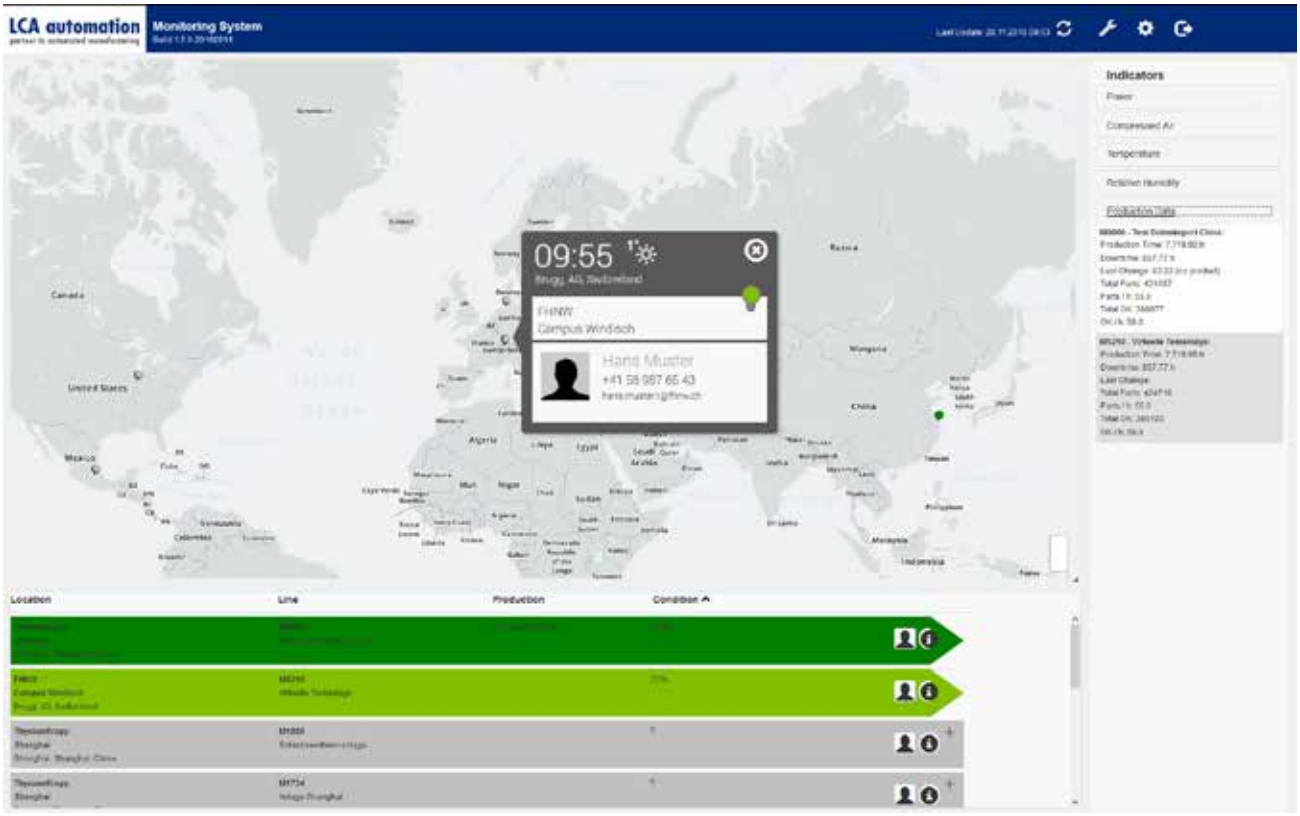

Abbildung 1: Web-App zur Überwachung des Zustands von Produktionsanlagen

Flexible Systemarchitektur mit Microservices

Im Gegensatz zu einer monolithischen Applikation bilden Microservices ein verteiltes System aus mehreren Services, wobei jeder individuelle Microservice eine spezifische Aufgabe übernimmt. Der Datenaustausch erfolgt über genau spezifizierte Netzwerkschnittstellen, damit die einzelnen Services möglichst gut austauschbar sind. Aufgrund der verteilten Architektur eignen sich Microservices besonders gut für Anwendungsfälle, bei denen Applikationen standortübergreifend betrieben werden.

Im Falle des Anlagen-Monitorings besteht die Systemarchitektur aus mehreren Arten von Microservices, welchen allesamt eine gemeinsame Java-Codebasis zugrunde liegt. Die Verwendung des *Spring Frameworks* [SprF] ermöglicht eine einfache Implementierung von *RESTful Webservices* [RFWS] und erlaubt mit Hilfe von Dependency Injection eine flexible Konfiguration einer Applikation. Die Erweiterung *Spring Boot* [SprB] ist bei der Entwicklung von Microservices besonders nützlich, weil sich damit ein Java-Webservice dank integriertem Applikationsserver als eigenständige ausführbare Applikation betreiben lässt.

Wir unterscheiden zwischen drei Typen von Microservices:

- *Site Data Service:* Dieser Service sammelt die Messdaten von verschiedenen Produktionsanlagen an einem oder mehreren Standorten, speichert sie in einer Datenbank und stellt die Daten wiederum als Webservice bereit. Je nach Konfiguration erfasst ein Site Data Service beispielsweise die Daten einer bestimmten Firma, oder auch eines bestimmten Firmenstandorts.
- *HQ Data Service:* Als übergeordneter Service dient ein HQ Data Service der Datenauswertung sowie der Überwachung mehrerer Site Data Services. Ein HQ kann dabei beispielsweise als Firmenzentrale (Headquarter) interpretiert werden. Mit diesem Service können standortübergreifende Daten in aggregierter Form über eine einheitliche Schnittstelle ausgegeben werden. Der HQ Data Service speichert dabei selbst keine Daten, so dass die Datenhoheit der einzelnen Standorte gewährleistet bleibt. Zudem ist auf einem HQ Data Service nur lesender Zugriff möglich.
- *Data Agent:* Da ein direkter Datenabruf von Messdaten aus CM+ aus Gründen der Netzwerksicherheit nicht erwünscht ist, wird ein Data Agent Service als Vermittler eingesetzt. Dieser wird im selben lokalen Netzwerk wie CM+ betrieben und ruft in konfigurierbaren Intervallen die aktuellsten Messdaten ab. Anschliessend werden die Messdaten an einen Site Data Service übertragen.

Abbildung 2 zeigt die Systemarchitektur anhand eines Setup mit zwei separaten Standorten

A und B, auf die über ein HQ zugegriffen werden kann. Die Data Services können standortunabhängig, also entweder on-site oder bei einem Cloud-Dienstleister betrieben werden. Lediglich die Datenerfassung und -analyse (CM+) sowie der Data Agent müssen am Standort der Produktionsanlage betrieben werden. Damit die erfassten Daten ebenfalls standortunabhängig sind, greift jeder Microservice auf eine eigene Datenbank zu.

Die Web-Apps sind unabhängige Applikationen, welche komplett im Web-Browser laufen und via REST API auf einen Site- oder HQ Data Service zugreifen. Sie können deshalb ebenfalls standortunabhängig betrieben werden und sind bei Site Data Services sogar optional, sofern ein HQ existiert.

Der Datenaustausch zwischen Clients (Web-App, Data Agent, HQ Service Backend) und Data Services, also die Übertragung und Abruf von Anlageninformationen und Messdaten, erfolgt immer via REST API. Der Austausch administrativer Nachrichten zwischen Site- und HQ Data Service erfolgt über einen separaten Kanal. Hierzu werden Message-Queues mit RabbitMQ [RbMQ] verwendet.

Die Implementierung eines Data Service ist in mehreren Schichten aufgebaut, wie in Abbildung 3 am Beispiel der Anlageninformationen („Lines“) gezeigt wird. Die oberste Schicht ist die Controller-Schicht, welche den RESTful Webservice bereitstellt. Sie dient ausschliesslich dem Datenaustausch und der Umwandlung der übertragenen JSON-Daten. Die eigentliche Geschäftslogik wird durch ein Service Interface (Bsp.: *LineService*) abgebildet, für das zwei Implementierungen existieren. Die Standard-Implementierung für einen Site

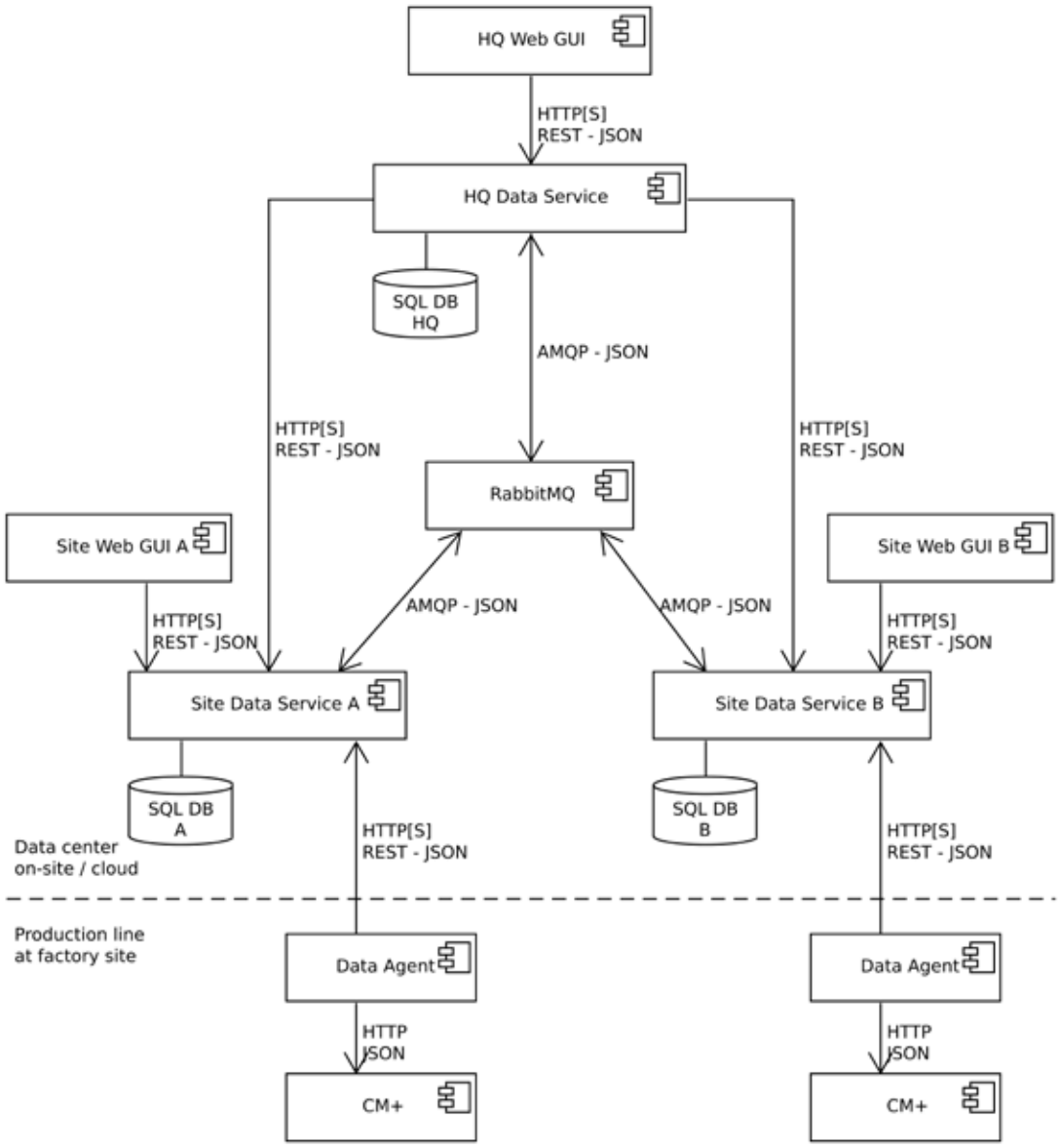

Abbildung 2: Systemarchitektur mit zwei Standorten und Headquarter HQ

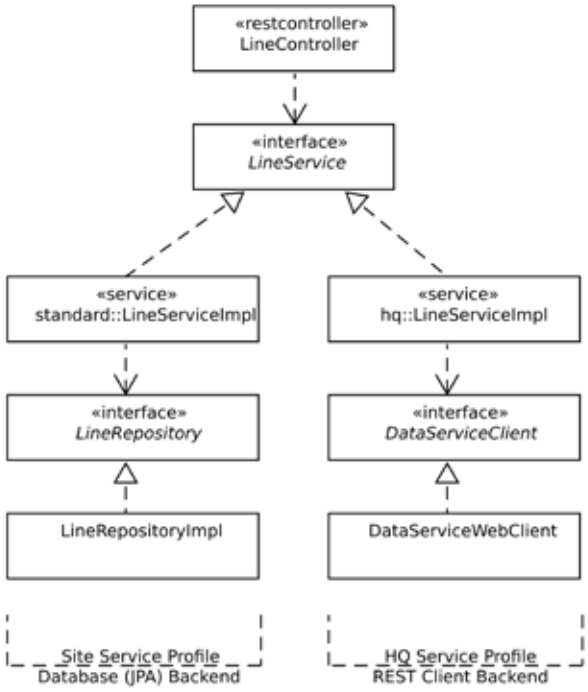

Abbildung 3: Service zum Abruf von Anlageninformation mit zwei Implementierungen

Data Service enthält Methoden zur Datenverarbeitung und bindet via JPA eine SQL-Datenbank an, um die verarbeiteten Daten zu speichern. Die Implementierung für einen HQ Data Service speichert keine Daten, sondern enthält als Backend einen Webservice Client, welcher Daten von einem entfernten Site Data Service abfragt. Welche Implementierung verwendet werden soll, wird durch Angabe des korrekten Spring-Profils in der Applikationskonfiguration festgelegt. Wird das Profil „hq" angegeben, wird die HQ-Implementierung verwendet, ansonsten die Standard-Implementierung.

Einheitliche Schnittstelle

Die Übertragung von Messdaten erfolgt via *RESTful Webservices* [RFWS]. Jeder Microservice implementiert also ein REST API als Client, Server oder im Falle des HQ Data Service auch beides. Als Datenübertragungsformat wird JSON verwendet.

Weil der Site und der HQ Data Service die gleiche Codebasis besitzen und sich nur durch ihre Backend-Implementierung unterscheiden, welche via Spring-Profil konfiguriert werden kann, ist das REST API für beide Services quasi identisch. Dies vereinfacht die Entwicklung der Web-App, welche zur Betrachtung der Monitoring-Daten verwendet wird. Es spielt für die Web-App keine Rolle, ob sie auf einen Site- oder einen HQ Data Service zugreift, deshalb muss sie nur einmal implementiert werden.

In einem einfachen Setup mit nur einem Standort kann ganz auf einen HQ Data Service verzichtet werden, indem die Web-App direkt auf den Site Data Service zugreift.

Sichere Datenübertragung

Mit dem standortübergreifenden Datenaustausch und der weltweit möglichen Datenabfrage via Web-App bekommt die Sicherheit der Datenübertragung eine entscheidende Bedeutung. Es kann davon ausgegangen werden, dass die Kommunikation im Wesentlichen über öffentliche Internetverbindungen abgewickelt wird. Deshalb müssen unautorisierte Zugriffe auf die Daten verhindert werden und die Kommunikation muss verschlüsselt erfolgen.

Beim öffentlichen, respektive standortübergreifenden, Zugriff auf Site oder HQ Data Services wird konsequent HTTPS eingesetzt. Auf diese Weise wird verhindert, dass Daten unterwegs mitgeschnitten werden können. Jeder Client, sei es ein User der Web-App oder ein Data Agent, muss sich zudem authentifizieren, um auf einen der Webservices zugreifen zu können.

In Bezug auf die Netzwerksicherheit hat die gewählte Systemarchitektur den Vorteil, dass kein direkter Zugriff auf betriebskritische Komponenten der Produktionsanlagen benötigt wird. CM+, welches die Monitoring-Daten über einen eigenen Webservice bereitstellt, ist nur im lokalen Netzwerk erreichbar. Die Übertragung von Daten zum Site Data Service wird ausschliesslich vom Data Agent angestossen, welcher auch die Verbindung zum Server initiiert. Ähnliches gilt auch für den administrativen Nachrichtenaustausch. Die Data Services greifen auf einen zentralen RabbitMQ Broker zu, wobei die Verbindung vom Data Service aufgebaut wird. Deshalb muss einzig der RabbitMQ Broker direkt durch die Services erreichbar sein. Auch bei RabbitMQ müssen sich angeschlossene Services authentifizieren und die Kommunikation kann verschlüsselt mittels Transport Layer Security (TLS) erfolgen.

Zentrales Management

In einem verteilten System von Microservices müssen die einzelnen Microservices einander bekanntgemacht werden, damit sie kommunizieren zu können. Es wird also eine Registrierung benötigt. Durch bestmögliche Automatisierung des Registrierungsprozesses kann der administrative Aufwand zum Nachführen der Konfiguration reduziert werden.

In unserem Fall geschieht das mit Hilfe der RabbitMQ Message-Queues. Einem Site Data Service muss lediglich die Queue des zugehörigen HQ Data Service bekanntgemacht werden, damit er sich nach dem Start selbst beim HQ Data Service registrieren kann. Auf diese Weise können einem HQ zusätzliche Standorte hinzugefügt werden,

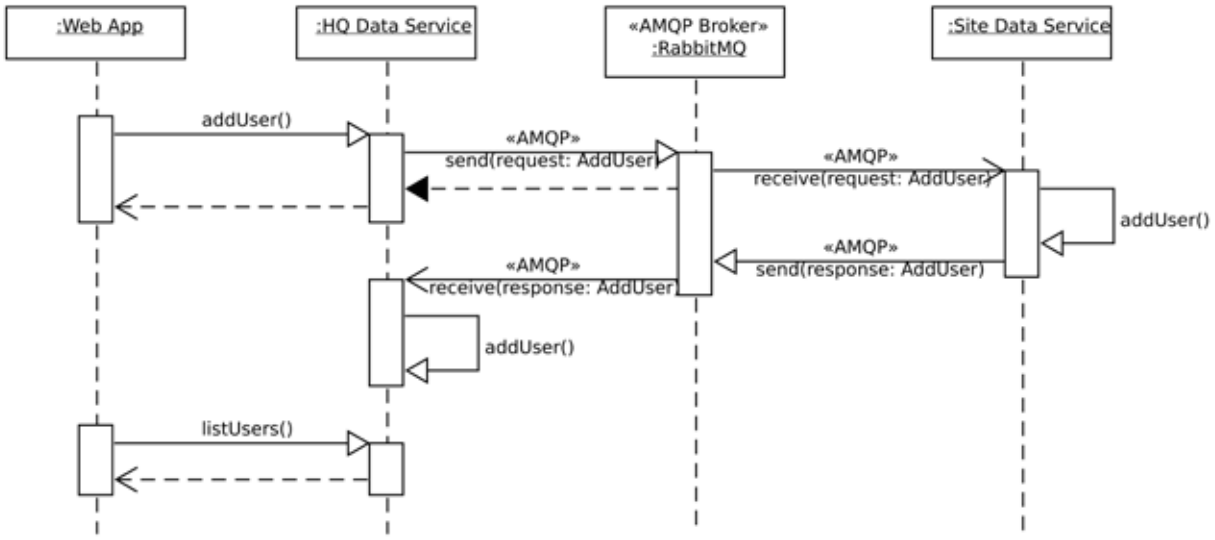

Abbildung 4: Service zum Abruf von Anlageninformation mit zwei Implementierungen

ohne dass die Konfiguration des HQ Data Service von Hand angepasst werden muss. Eine zweite Funktion der Message-Queues ist die Überwachung der Verfügbarkeit der angeschlossenen Standorte. In regelmässigen Abständen prüft der HQ Data Service mit einer „Ping"-Meldung, ob die Standorte noch aktiv sind. Bleibt eine Antwort aus, kann ein Standort als inaktiv markiert werden.

Auch die Verwaltung der Benutzerkonten für den Web-Zugriff kann zentral erfolgen. Hierfür bietet die Web-App einen Administrationsbereich zur Benutzerverwaltung an. Wird ein HQ Data Service administriert, können auf diesem auch Benutzerkonten angeschlossener Site Data Services erstellt und bearbeitet werden. Änderungen werden, wie in Abbildung 4 gezeigt, via RabbitMQ-Message-Queue an den entsprechenden Service propagiert und im HQ erst nach erfolgreicher Verbreitung gespeichert.

Trotz zentralem Management arbeitet ein Site Data Service autonom, denn die für die Benutzer-Authentifizierung nötigen Informationen werden auch lokal gespeichert. Sollte also z.B. der HQ Data Service ausfallen, können am Standort weiterhin Daten von den Produktionsanlagen erfasst werden.

Fazit

Die hier gezeigte Systemarchitektur mit Microservices ermöglicht einen standortübergreifenden und bei Bedarf weltweiten Zugriff auf Daten, welche gerade im Kontext der Industrie 4.0 zentrale Aspekte sind. Gleichzeitig wird aber auch dem Bedürfnis nach Datensicherheit Rechnung getragen, da die Betreiber der von LCA hergestellten Produktionsanlagen die Hoheit über ihre betriebsrelevanten Daten nicht durch eine Auslagerung des Betriebs der Dienste abgeben müssen. Sowohl der Anlagenbauer LCA als auch die Anlagenbetreiber können letztendlich von dieser Lösung profitieren, da sie für den Anlagenbauer eine kostengünstigere Wartung und für den Betreiber eine Reduktion der Stillstandzeiten und damit unmittelbare finanzielle Anreize ermöglicht.

Referenzen

[LCAA] LCA Automation: https://www.lca.ch/

[MicS] Definition Microservices: http://www.martinfowler.com/articles/microservices.html

[SprF] Spring Framework: https://spring.io/

[SprB] Spring Boot: https://projects.spring.io/spring-boot/

[RFWS] Definition RESTful Web Services: https://de.wikipedia.org/wiki/Representational_State_Transfer

[RbMQ] RabbitMQ: https://www.rabbitmq.com/

Automated Testing in the Internet of Things

This article presents a novel approach to testing distributed systems. Our automated test environment (ATE) is created to validate the BACnet/IT building automation protocol and is easily adaptable to other domains. During development of the new BACnet/IT reference implementation, we had to face several testing challenges. Based on that, we derived requirements for the ATE. The result is a flexible and lightweight test environment, which consists of only a few interacting components. Our ATE is able to simulate real-life situations like a power outage or a replacement of a BACnet/IT device. Further, it allows manipulating the behavior of BACnet/IT components during runtime. With such a test environment it is possible to automate tests in a straightforward and efficient way.

Thomas Dobler, Artem Khatchatourov, Christoph Stamm, Wolfgang Weck | wolfgang.weck@fhnw.ch

We use more and more digital devices in everyday life. When you turn on the lights, a traditional switch actually closes an electrical circuit, but in many modern buildings the switch sends a digital message over a communication network to the light, or even to several lights in the same room. The advantages are plenty. There is not just one light switch next to the door, but lights can be turned on and off or even dimmed from several places. In our lecture rooms, for instance, one such place is the speaker's desk. Also, lights might be switched by automatisms, for instance, turned off whenever sensors detect that the room is empty or that there is enough daylight coming in through the windows.

Switching lights is just one example of the upcoming Internet of Things (IoT). A rapidly increasing number of digital devices may increase comfort and use energy more efficiently through automatization and mutual interaction. With the IoT, many relatively simple and small devices will exchange short messages with each other, partially with real time constraints. Using Internet technology for data exchange reduces cost by avoiding dedicated cables and by sharing software infrastructure, such as name and addressing services and security mechanisms.

From an engineering point of view there is an important paradigm shift included. IoT takes us from system architectures with a central service embedding all intelligence and being contacted by client devices to fully interconnected networks where every device can contact any other device. A system's complexity is not embedded (and encapsulated) in a central node anymore, but spread across the network of a large number of interacting devices, each of which by itself can be quite simple, though.

As engineers we want to have and to provide evidence that our constructions meet their requirements. Next to systematic (partially formalized) construction methods, an important tool for this is systematic testing of new or modified machinery. For single-node computer systems there are established test methods and tools. Programmers deploy unit tests to their code. When software systems are built from source code automated tests are run, and so forth. Servers can be tested through automatically simulated clients and vice versa. However, these approaches only poorly cover situations, when important system properties rely on the cooperative interaction of hundreds or thousands of devices and the connecting infrastructure.

In this paper we describe our automated testing environment for IoT in the application domain of building automation. This testing environment is one contribution of our institute in a joint CTI project[1] with FHNW's Institute of Automation and Siemens Switzerland, Building Technologies Division. Our part in the project is reviewing, prototyping, and evaluating the draft standard of the new internet-based instance of the BACnet protocol: BACnet/IT [BAC16]. In order to be able to evaluate our implementation of the communication stack and with it the new BACnet/IT draft standard, the need for an automatic tool for testing has arisen.

First of all we introduce building automation and BACnet. Then we look at two exemplary application scenarios, raising specific testing demands. The two scenarios shall illustrate requirements to testing tools for BACnet/IT specifically and distributed IoT systems in general. In practice, there are many more requirements and corresponding test cases that can be served with the same set of tools. Finally, we present the elements of our ATE.

Building Automation

Building automation is the (centralized) control of a building's heating, air conditioning, lighting

1 CTI Project: Convergence of Building Automation and IT World, KTI-Nr. 16841.1 PFES-ES

and other systems through a building management system.

Building automation of the future requires flexible communication solutions. Due to the development of costs and the spread of Internet technologies, standard IT solutions are increasingly being used. IT infrastructures and services are currently undergoing a process of adaptation to the new requirements of IoT by providing suitable protocols for the integration of devices (CoAP, RPL, 6LoWPAN). Building automation, which uses IT infrastructure, is also subject to this adaptation process.

BACnet/IT

BACnet is a well-established standard among building automation manufacturers. It has been originally defined for proprietarily wired infrastructure to connect sensors and actors in a building [BAC]. With the ubiquitous Internet of today, migrating BACnet onto standard Internet mechanisms is advantageous.

BACnet/IP allows communication via IP-based networks, but uses IP only as a data link and often communicates with IP broadcasts. Based on the data link, BACnet/IP uses its own BACnet-specific protocols on upper layers. This leads to massive acceptance problems in the IT world and prevents easy interoperability and integration into other domains. These problems have been recognized by the standardization committee and led to a new specification.

The new BACnet/IT draft standard describes how to implement the established application layer based on standard Internet protocols such as HTTPS, TLS, WebSocket, and standard Internet services, such as DNS and DHCP providing a basic communication layer for BACnet messages [BAC16].

With BACnet/IT, building automation no longer runs its own communication infrastructure, but becomes a guest among others on a network with standard Internet technologies. This cuts down operation cost and can even raise reliability if part of the saved budget is used to operate the common standard infrastructure with extra redundancy. As a trade-off, building automation must get along with configurations and restrictions set up by the network operators. Consider for example a bank or another company with high security standards as a building's tenant. Following today's best practices, the network will be configured in zones shielded against each other by firewalls, possibly using network access control, and so forth. Assignment of network addresses and device names may have to follow specific rules. Thus, the building automation devices need to acquire essential information from responsible IT services under more difficult conditions than in a proprietary network.

So, one of the specific challenges is that BACnet/IT devices must be able to cope with all kinds of restrictions a network operator may put on them. At the same time they must not rely on high service availability, because there may be no professional network operation at all. The latter is the case with smaller companies, when IT networking is neither business critical nor part of the core competence. The BACnet/IT draft standard tries to cover all relevant situations within this broad spectrum of possibilities. It defines how devices react to specific situations.

Example 1: Speed Test

In this and in the next section we describe two exemplary application scenarios, raising specific testing demands. Both scenarios shall illustrate requirements to testing tools for BACnet/IT specifically and distributed IoT systems in general.

Some IoT applications have significant real-time requirements. In building automation these are, for instance, those involving human observable reaction to human activity or alarm transmissions. Consider again the introductory light switch example: An actor triggers a light switch and the light in the room turns on. The light bulb (or more precisely the whole system) needs to react fast enough, so that humans don't experience a delay, not even in a whole corridor with lots of individual light bulbs. Hence, there are tight time constraints specified with such scenarios. Ensuring compliance with these requirements calls for measuring time under varying circumstances, e.g. different network loads.

Figure 1 illustrates this test case. There is a flow of messages between two BACnet/IT controllers. One controller is connected to a light switch and the other to a light source. The two deltas mark time spans we want to measure.

Measurements must be repeated under different conditions with varying network load and bandwidth or in different topologies. This allows comparing results to determine how network conditions affect the performance. Having to run the same tests many times raises the need to program

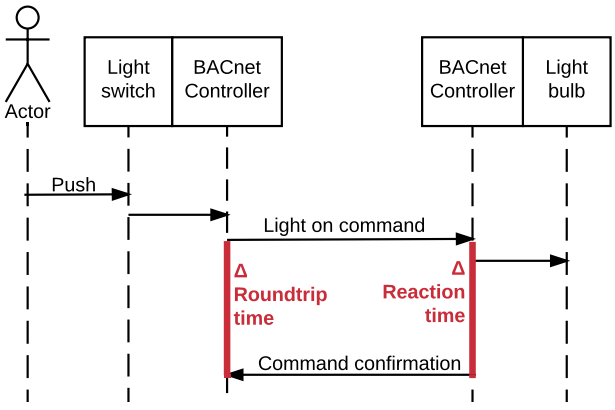

Figure 1: Measuring reaction and roundtrip time in a light switching scenario

the scenario to be run and measured so that it can be executed automatically over and over again. Further, the various environments in which the scenario shall be run must be set up and configured automatically. Without such support, broad band testing would simply be too expensive.

Example 2: Power Failure

Rarely occurring scenarios as a power outage, for instance, may have high impact, ranging from considerable cost to life threatening. Hence, it is especially important to simulate such situations during testing rather than waiting for them to occur in a live setting. As an example, you may consider a temporary power outage in a building. Some devices of the building automation system will stop to operate, while others will continue to work, based on resilient power supply, such as a local battery. Further, the network infrastructure may be inhibited so that communication channels between devices close down.

It is up to the application programs being run by the individual – temporarily disconnected – devices to cope with such a situation and to prevent major disaster. This, however, is not the topic we are concerned with in this example. We are interested in what happens, when power comes back again and both the network and some temporarily powerless devices start to recover. The problem then is, that many devices will restart their communication at the same moment, synchronized by electricity becoming available again simultaneously everywhere.

In such moments, specific network services become bottlenecks, because they receive requests from each device trying to integrate itself into the system. Consider for instance, name services, responsible for mapping device and service names to actual network addresses. The BACnet standard requires devices (re-)entering a network to announce their availability together with the application domain services they offer. This involves sending a registration message to a specific BACnet directory server (BDS), which cooperates with a standard domain name services (DNS).

Usually, devices are integrated into and removed from the system one by one, so that the BDS can easily handle these registrations. Having many devices synchronized through simultaneous power up will put the BDS and the network under unusual stress. Figure 2 illustrates this specific situation.

It is fairly easy to imagine similar stress situations with other IoT applications also, just because of the sheer number of devices interacting. Forced synchronization, can have various causes, e.g. a fire alarm that triggers a whole number of devices like sprinklers and safety lighting.

Of course, the BACnet/IT draft standard has foreseen such situations and prescribes counter measures, such as deferring the registration messages for a randomly selected time span. Our protocol stack implements this. Still, engineering standards require not only to verify such measures but also to quantify the load on bottleneck units as well as the duration until the system recovers to normal operation. On top of this, it is of interest, how many registration requests are dropped by the BDS due to overload.

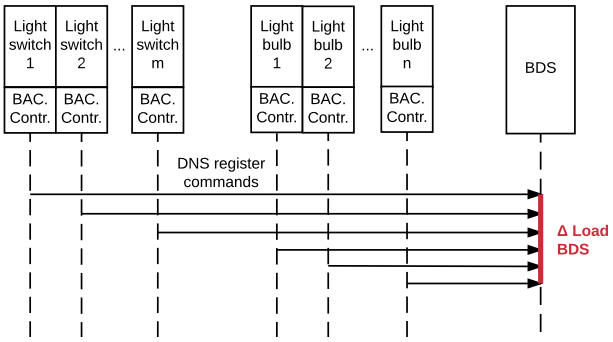

Figure 2: BDS load after a power outage

To run tests that can produce relevant data requires many (such as hundreds) devices to simultaneously send out a message. Connecting a hundred physical devices to a single power socket and power them up together might be possible but would be a costly and inflexible solution. Virtualization is simpler and cheaper. Simulating the synchronized power up situation requires automatically triggering many such virtual devices to send out specific messages at the same time. This process must be programmed (scripted) and invoked during the test. In addition, a stored test procedure will also lead to reproducible test results.

Requirements to a Test Environment

The two examples above illustrate a set of requirements to an automated test environment for IoT systems in general and our specific BACnet case especially:

- *Devices must be manipulated:* Many test scenarios require some degree of invasive action on the devices forming the system under test. Some scenarios involve devices entering or leaving a system, such as described in the power failure context above, so they must be deployed and run with specific configurations or halted. An actual application must be mocked to initiate message transmissions. To measure time intervals, respective instrumentation must be injected.
- *Tests must be controlled from a single workplace:* Testing an IoT system involves operating several devices in coordination. A small number of devices could be put together in a rack or on a table and be controlled physically. This does not work anymore when many devices are

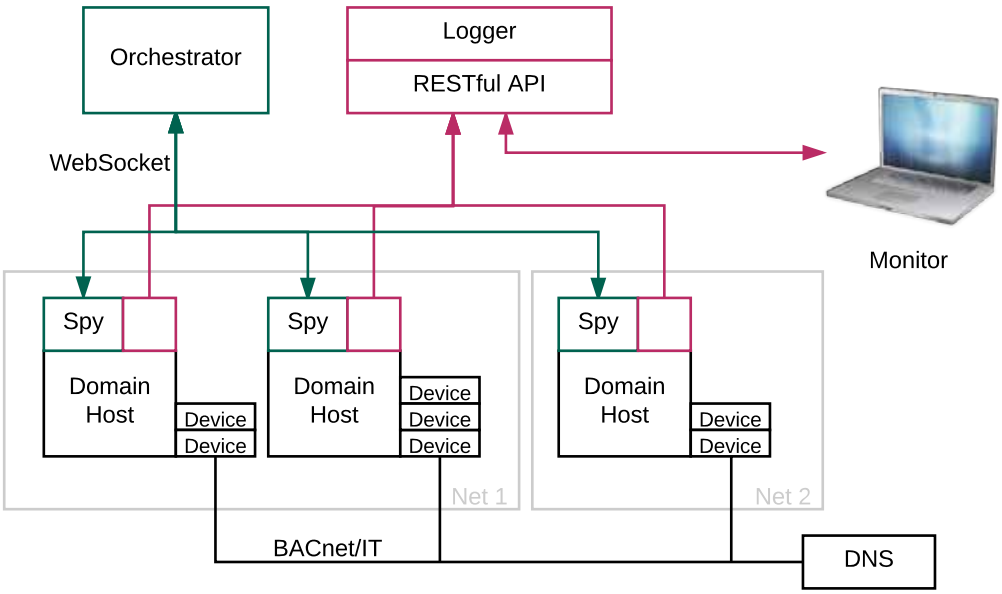

Figure 3: Overview of the components of our Automated Test Environment (ATE)

involved. Some tests and measurements, such as load testing, require coordinated access to a very large number of devices. This should be controlled from a single controlling workplace, managing all devices on the network including nodes in the infrastructure.

- *IoT systems under test must be scalable:* Depending on the application area, IoT systems vary in size. Even building automation systems may range from a few controllers to large amounts of devices. Depending on what to test or to measure, infrastructure from a private home LAN to an enterprise network with dedicated DNS and an aggressive firewall setup must be set up. The testing environment needs to scale from a few – maybe even physical devices – to large amounts of virtual entities.
- *Reproducible tests must be programmed:* There are two reasons, why test procedures must be repeated. First, the same tests may have to be run with varying parameters, such as a different number of devices, different network capacity, etc. Second, tests are not just a one-time shot to prove an implementation to be correct, but they are used as regression tests during further development and change. If for each test a lot of steps have to be repeated manually, the process becomes cumbersome and error-prone. Hence, test procedures should be written as automatically executable programs.
- *System data of devices must be accessible:* As the examples show, there is a need to collect various data on individual devices during testing. For instance, testing with the power outage scenario, we would like to measure CPU load, memory usage, number of open connections, and so on. This requires accessibility of the respective system data. For the first example from above, we need to measure time between specific network events. These events can be protocolled in a log file, which can later be pulled from the device to the central workplace for evaluation.

Related Work

Surprisingly, there are only a few frameworks available for testing distributed networks:

- *Java Device Test Suite* [JDTS]: It is designed for testing embedded mobile devices. It is rather a monitor for an embedded device on the network than a framework for distributed testing of these devices.
- *TETWorks* [TET]: It is designed for distributed testing and has support for different languages and platforms. Unfortunately, it does not allow reading of system parameters, which is imperative for our needs. However, the software is well documented and its approach is a good base for an extended tool.

None of these tools suits our needs, so we decided to implement our own *Automated Test Environment* (ATE). It is designed specifically for BACnet/IT, but our approach can be used in any building automation domain. We decided to base our environment on the approach of TETWorks. So, if you are familiar with TETWorks, then you will recognize some of its components in our new test environment.

Automated Test Environment

Our ATE depicted in Figure 3 allows us to deploy, start and control virtual devices in the cloud as well as in a laboratory and it simplifies logging and visualization of logging data. It consists of five major components: An *Orchestrator* to manage operation of BACnet/IT devices on the network, a *Logger* to collect status and event updates from these devices visualized by a *Monitor*, and a

network node including the two other major components: *Domain Host* and *Spy*.

A network node might be a virtual or physical machine containing one or more Domain Hosts (e.g BACnet/IT hosts). Each of these Domain Hosts can contain one or more BACnet/IT devices.

Virtual BACnet/IT devices usually do not produce traffic on the network because they are not programmed to play the role of a real device. Thus, communication must be invoked manually by the Orchestrator through the Spy, which serves as an intermediate interface. In the following sections we describe each of these components in more detail.

The Orchestrator plays an important role in simplifying automated testing in an IoT environment. It has been designed to fulfill some of the requirements listed above: running and controlling reproducible tests from a single workplace. With dozens of network nodes it is cumbersome to log in individually and to deploy devices manually. This is a common problem in cloud administration and there are many solutions that solve this problem for general use. However, our Orchestrator is more powerful than common cloud administration tools, because it is able to access Domain Hosts even after their deployment.

Some tests require coordinated teamwork from up to hundred BACnet/IT devices on the network. This calls for automated test procedures, which force devices to exchange messages in a fixed sequence. A test written as a procedure can be uniquely identified for documentation purposes and used in other network environments with different hosts. Doing so will produce comparable results, which can be used to compare setups and implementations.

Orchestrator

The *Orchestrator* is a centralized tool, which manages physical or virtual devices on the network. It controls these devices by sending commands of two different types over Websocket connections:

- *Maintenance Commands* trigger operations such as starting a *tcpdump* process or deploying a new Domain Host with given configuration. These commands are run by Spies and do not concern any BACnet/IT communication.
- *BACnet Commands* are directed towards the Domain Host. These commands trigger BACnet/IT devices to send messages to other physical or virtual BACnet/IT devices in the same domain.

The Orchestrator has two modes of operation:

- *Interactive mode:* A command line interface provides commands to control the devices and other nodes on the network. This mode of operation is not suited for large tests.
- *Test Procedures* are Groovy scripts of more complex command sequences. Such scripts automate coordination between BACnet/IT devices on the network. These test procedures have to be independent from each other to avoid complications, e.g. identifier collisions. So it is good practice to terminate all used devices after a test is completed.

The test procedures are divided into three main stages:

- *Setup* deploys all required devices for the test with a given configuration.
- *Test* contains the commands that must be executed to run the test including the evaluation of its results.
- *Teardown* phase contains the cleanup routine for the test.

Spy

The Spy is a Java application running on a virtual or physical machine. It is an interface for the Orchestrator to communicate with the Domain Host. Additionally, it manages operations on the node. These include deploying and halting of Domain Hosts and executing shell commands for reading CPU and memory usage and starting *tcpdump*, which is used to write network communication to a file. Afterwards, this file is sent to the Orchestrator for evaluation.

In essence, the Spy is a program that boots with the computer and controls some of the operations in the host system. It is similar to malware on a compromised computer in a botnet. To mitigate potential damage, we have limited a Spy's scope of shell commands to a few that can not do any harm to the host. This however, does not mean that no damage can be done to the network: A Spy can easily flood the network with traffic produced by a BACnet/IT device. Indeed, a DDoS attack on one BACnet/IT device conducted by other BACnet/IT devices is a relevant test case.

Domain Host

To run a test environment with distributed interacting devices a well-defined setup is needed. For example, a BACnet/IT system includes a BDS, a directory service, and a number of devices. These devices maintain objects and properties, which have to be configured. Further behavior has to be configured as well. For example, the configuration tells a device it has to announce itself to the BDS.

During development and testing of the BACnet/IT, the involved components have to be in an initial state. Thus, we have to simplify and automate the process of making the system ready for further tests.

A *Domain Host* has to fulfill two main tasks:

- It configures and starts the devices at a specific node according to a given configuration, which defines for example the number of devices and the value of their objects and properties, the IP address of the DNS, the usage of

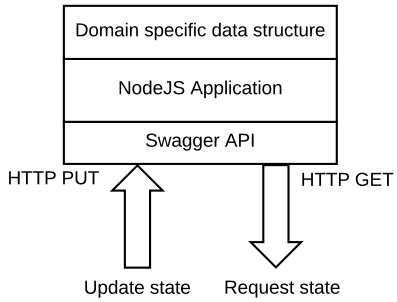

Figure 4: Simple architecture of the Logger

TLS, BasicAuth or CORS during communication and so on.

- A Domin Host acts as an interface between a Spy and the BACnet/IT devices on a node. As described above, it receives commands sent by a Spy, passes the commands to BACnet/IT devices, and forces them to communicate according to the received command.

Logger

The *Logger* receives and maintains logging data from all Spies, Domain Hosts, BACnet/IT devices and other BACnet/IT system components. It is a nodeJS application keeping track of the events and state changes (see Figure 4). For example, devices announce themselves with the BDS, send messages to each other, and may change the value of their properties. Besides the devices, the communication stack of a device has also a changing state and sends logging information to the Logger, and even the zone file entries of the DNS changes are reported to the Logger.

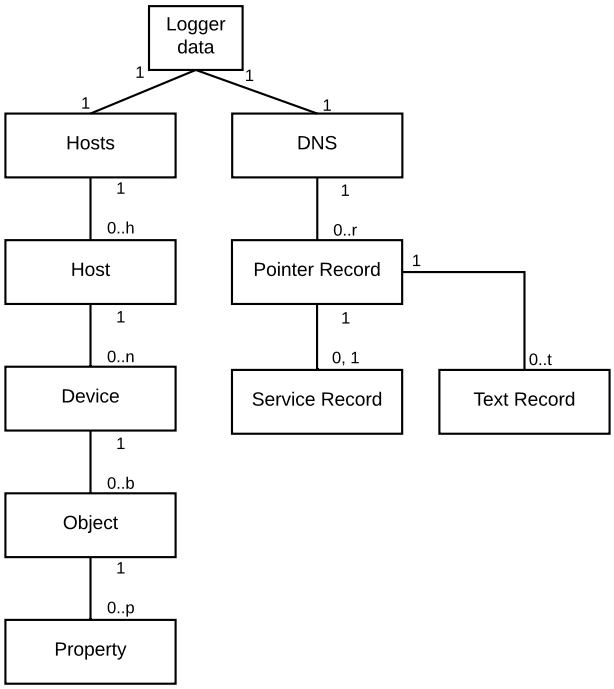

Figure 5: Data structure of the Logger applied to the domain of BACnet/IT

Every component of the system (e.g. a Spy, a Domain Host or even a BACnet/IT device) decides when and what information it will send to the Logger. After a message from another device is received, a device may inform the Logger about that incoming message.

Getting an overview of all the notable events and state changes is a challenge. Observing several standard output logs at the same time is arduous and confusing. Thus, one could adjust the logging level to standard output to get a proper system overview, but this is inconvenient, because the logging level can differ per component and test case. Heading this problem, our solution is oriented towards a notification mechanism. Whereby one designated Logger collects all the information it receives. Communication with the Logger is through a RESTful API.

For the definition and implementation of our RESTful API we use *Swagger* [SWG]. Swagger provides a number of tools to design a RESTful API and offers simple client and server implementations in different programming languages.

A typically RESTful API based on Swagger is defined as follows:

```
/route/{entity}:
  [HTTP METHOD]:
    parameters:
      - name: [name]
      in: path
      type: string
    responses:
      200:
        description: [text]
      404:
        description: [text]
```

Our Logger stores the received data in a domain specific data structure. For BACnet/IT, for example, we use the data structure described in Figure 5. With this data structure, we are able to keep track of all the important and notable events and state changes during the runtime of our BACnet/IT implementation.

The same RESTful API is also used by a *Monitor*. In our understanding a Monitor is just an application that analyzes and visualizes data from the Logger.

Monitor

Any Monitor implementation can request the current state from the Logger in JavaScript Object Notation (JSON). In our implementation of the Monitor we use D3 to create sunburst charts [D3JS]. D3 is a popular JavaScript library for data visualization. Sunburst charts fit very well to our 1-to-m entity relations of the data representation and let one zoom into areas of interest.

In Figure 6 we display information we care about during development and testing of the BACnet/IT system. On the innermost ring we see *Hosts*

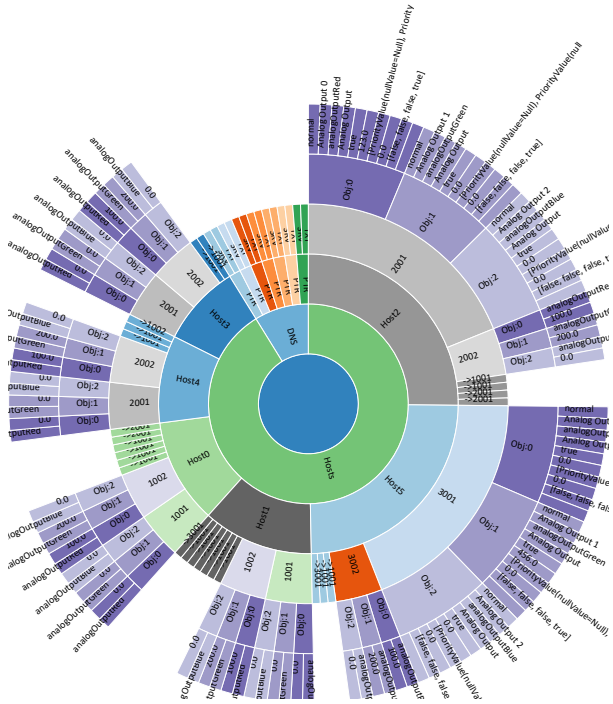

Figure 6: Sunburst visualization of Logger data in Monitor

and *DNS*. A host contains several devices, e.g. *Host2* contains devices 2001 and 2002, a device may have several objects, e.g. device 2001 has objects Obj:0-2, and finally an object consists of several properties (visualized in the outermost ring). In the upper part of Figure 6, the current state of the DNS of the system is fanned out.

Example 1 using the ATE

The diagram in Figure 7 describes how the ATE can be applied to the *Speed Test* example presented previously. For the test we need two instances running Ubuntu with a Spy installed on each one. One instance will host a BACnet/IT device simulating a light switch (A1) and the other will pose as an Internet connected light bulb (B2). The goal is to measure the roundtrip time of a request (BACnet/IT message) from A1 to B2 including a response from B2 to A1 confirming a successful execution of that request. The test is stored in a Groovy script (called procedure). It can be run not only in local environment but over different networks or even over Internet. Also, additional traffic may be generated on the network to simulate real-world conditions.

The test consists of the following steps:

A. The Orchestrator deploys the Spy on a remote machine via SSH in case the Spy is not already deployed. The SSH tunnel is closed afterwards and all further communication is carried out over Websocket.

B. Maintenance Commands signal both Spies to deploy a Domain Host with a configuration delivered as a serialized object.

C. Maintenance Commands to Spies invoke the tcpdump process.

D. A BACnet Command to the Spy on instance 1 enters the Domain Host and triggers a BACnet message dispatch from device A1 to B2 on instance 2. The message destination is only provided as a BACnet/IT device identifier. A1 doesn't know the destination IP address yet and resolves the identifier via DNS. Finally, this message is sent to the returned IP address.

E. When a response has been received, the test is completed, the tcpdump processes are terminated, and the tcpdump file is delivered by the Spy to the Orchestrator, which analyzes both files and computes the test results.

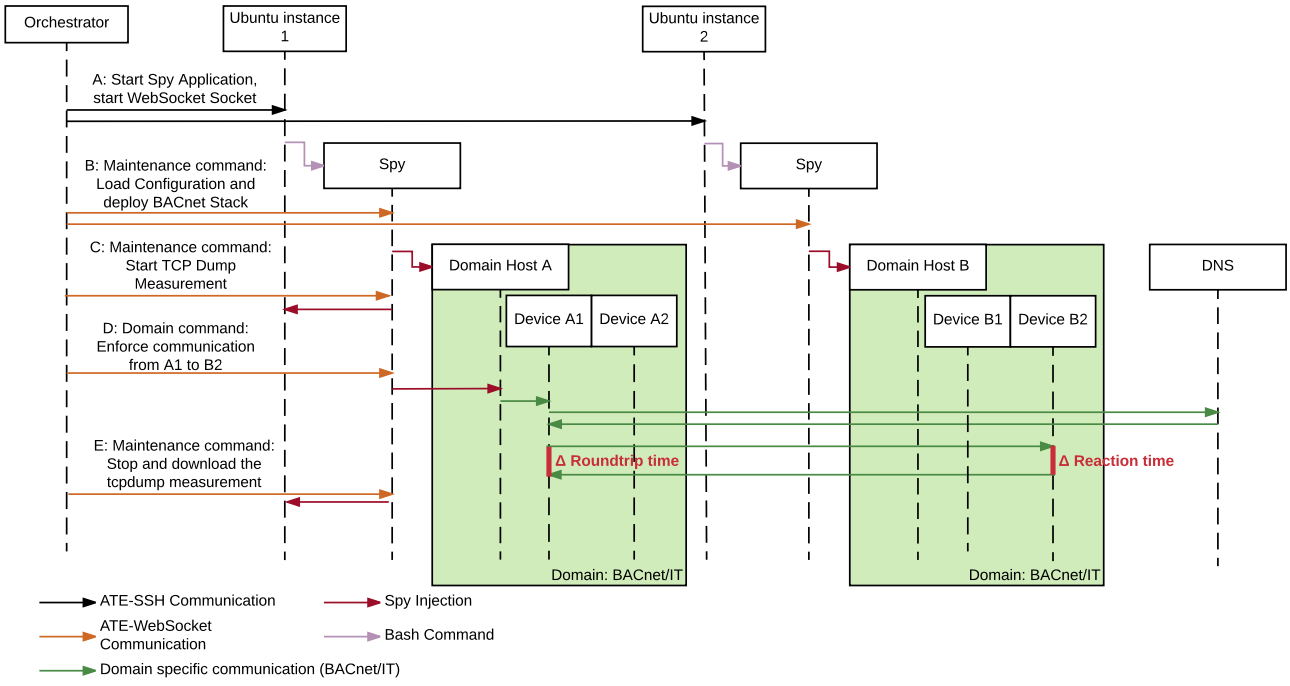

Figure 7: Communication flow initialized by the Orchestrator in example 1 (time measurements)

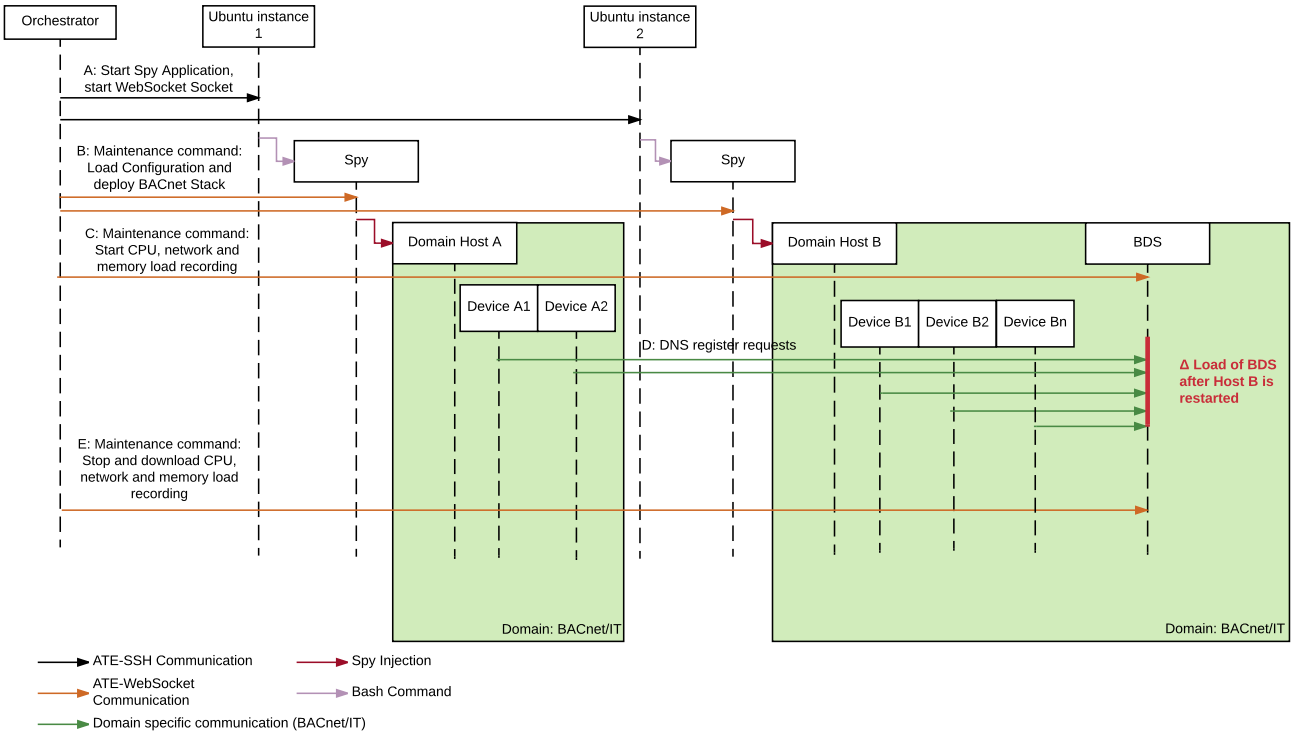

Figure 8: Communication flow initialized by the Orchestrator in example 2 (power outage)

Example 2 using the ATE

Let us pick up the second test example explained previously that occurs when all devices in a building power up at the same time and announce their availability to the BDS (Figure 8). During this process the BDS is flooded with registration requests from hundreds of devices residing on dozens of hosts. One can describe this massive load on the BDS as a DDoS attack.

During the test, we deploy a BDS device and ten Domain Hosts with ten BACnet/IT devices each. Depending on the configuration these devices may send a registration message immediately or wait a random time span before sending a registration message to BDS. This deferred registration helps to reduce the peak load on the BDS and avoids any dropped messages. The load is indicated by CPU and memory usage, which is recorded during the test. After recording, the CPU and memory usage is evaluated, and the evaluation results lead us to an improved device configuration for this network.

The test consists of the following steps:

A. The Orchestrator starts a Spy on each instance over SSH.
B. CPU and memory usage recording is started on the instance hosting the BDS.
C. For the main test, Domain Hosts are deployed on every instance. During startup, they deploy their BACnet/IT devices and automatically register them using the BDS.
D. The test is completed when all hundred devices have been registered.
E. The Orchestrator collects recordings from the BDS and evaluates the results.

Conclusions and Outlook

The Automated Test Environment presented in this article allows us to simulate real life scenarios like a power outage, BACnet/IT component failures, malfunction of involved network components or a complete rearrangement of the existing IT network. Without this ATE we wouldn't be able to automate tests in IoT systems in such a straightforward and efficient way.

In the future we want to improve the stability of our ATE and develop a first adaption to another use case beyond building automation. Further, continuous integration plugins and automated system sanity checks would extend the functionality and a graphical user interface could improve the ATE experience.

References

[BAC] BACnet, official website: http://www.bacnet.org/

[BAC16] Proposed Addendum bj to Standard 135-2016, BACnet® - A Data Communication Protocol for Building Automation and Control Networks, December 2016. http://www.bacnet.org/Addenda/Add-135-2016bj-apr1-draft-2_chair_approved.pdf

[D3JS] Data-Driven Documents: https://d3js.org/

[JDTS] Java Device Test Suite: http://www.oracle.com/technetwork/java/embedded/javame/javadevice-140362.html

[SWG] SWAGGER open source framework: http://swagger.io/

[TET] The Test Environment Toolkit, TETWorks on the web: http://tetworks.opengroup.org/

Optimierte Energiebewirtschaftung

Die Bereitstellung elektrischer Energie an die Haushalte ist eine komplexe Aufgabe der Energieversorgungsunternehmen. Die Energie wird über einen mehrjährigen Zeitraum in mehreren Tranchen von unterschiedlichen Anbietern und Marktplätzen und unter der ständigen Unsicherheit des Marktes beschafft. Eine gelungene Energiebeschaffung, also der Handel zum richtigen Zeitpunkt, bedeutet für ein Energieversorgungsunternehmen einen Wettbewerbsvorteil. In diesem Bericht wird die Energiebeschaffung erklärt und aufgezeigt, wie der Prozess mittels Software unterstützt werden kann.

Peter Gysel, Daniel Kröni | peter.gysel@fhnw.ch

Ein Energieversorgungsunternehmen (EVU) erzeugt üblicherweise den Strom nicht selbst, sondern beschafft sich die Energie entweder direkt von Kraftwerksbetreibern „over the counter" (OTC) oder aber an der Strombörse. Strom wird gehandelt wie andere Rohstoffe. Am Terminmarkt, wo der Strom für die nächsten Jahre gehandelt wird, kann sich das EVU für die zukünftige Lieferung bereits heute einen Preis absichern. Es wird mit standardisierten Produkten gehandelt. Dabei wird zwischen Base- und Peak-Produkten unterschieden. Das Base-Band ist durchgehend 24h, 7 Tage die Woche. Das Peak-Band deckt die Zeiten hohen Verbrauchs ab, von Mo. bis Fr. jeweils von 8:00 – 20:00 Uhr. Angeboten werden die Produkte mit unterschiedlichen Lieferperioden wie Jahr, Quartal oder Monat. Zum Beispiel kann man ein 1-Megawatt-Peak-Band für den Monat August kaufen.

Der Graph in Abbildung 1 zeigt den prognostizierten Energiebedarf in Megawatt (MW) einer Kleinstadt in der Kalenderwoche 25. Es ist klar ersichtlich, dass der Energiebedarf im Laufe des Morgens zunimmt, gegen Mittag einen Höhepunkt findet und abends, wenn die Lichter gelöscht werden, wieder abnimmt, mit einem Tiefpunkt in der Nacht. Zudem unterscheiden sich die 5 Werktage klar vom Wochenende, wenn die Industrie ruht. Die markierten Bereiche „Peak-Band" und „Base-Band" zeigen die bereits beschaffte Energie: ein 21-MW-Base-Band mit einem 2-MW-Peak Band darüber.

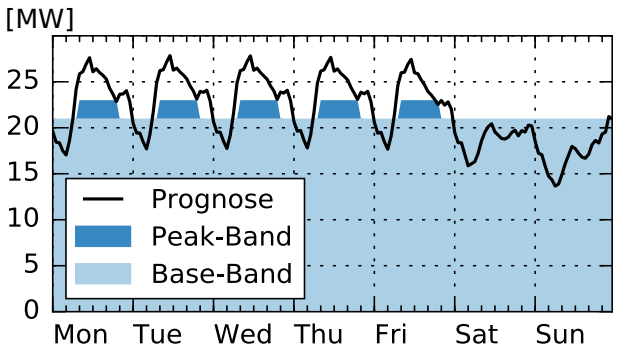

Abbildung 1: Prognostizierter Energiebedarf und Eindeckung mit Standardprodukten einer Kleinstadt in Kalenderwoche 25

Tatsächlich beschafft sich ein EVU die benötigte Energie in mehreren Tranchen und bei unterschiedlichen Anbietern. Die Beschaffung ist ein Prozess mit mehreren Phasen, der bereits Jahre vor der Lieferung beginnt. Dieser Prozess ist in Abbildung 2 dargestellt. In einem Fenster von etwa drei Jahren bis ein Quartal vor der Bereitstellung der Energie, wird die grosse Menge der benötigten Energie in mehreren Tranchen beschafft (Phase a). Das gibt dem EVU die Sicherheit bereits einen grossen Teil eingedeckt zu haben, für den Fall, dass der Energiepreis plötzlich unerwartet stark steigt.

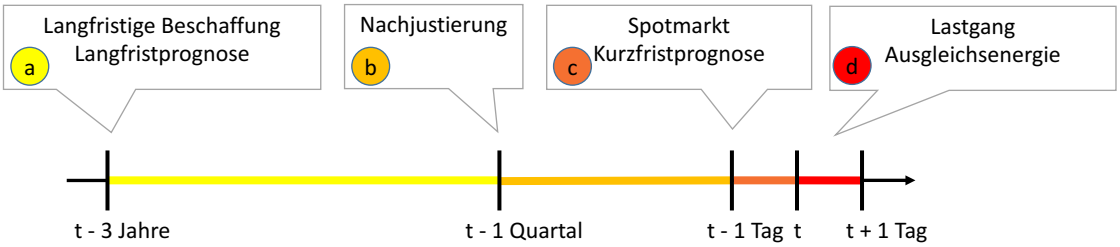

Abbildung 2: Mehrstufige Beschaffung der Energie, die zum Zeitpunkt *t* geliefert wird

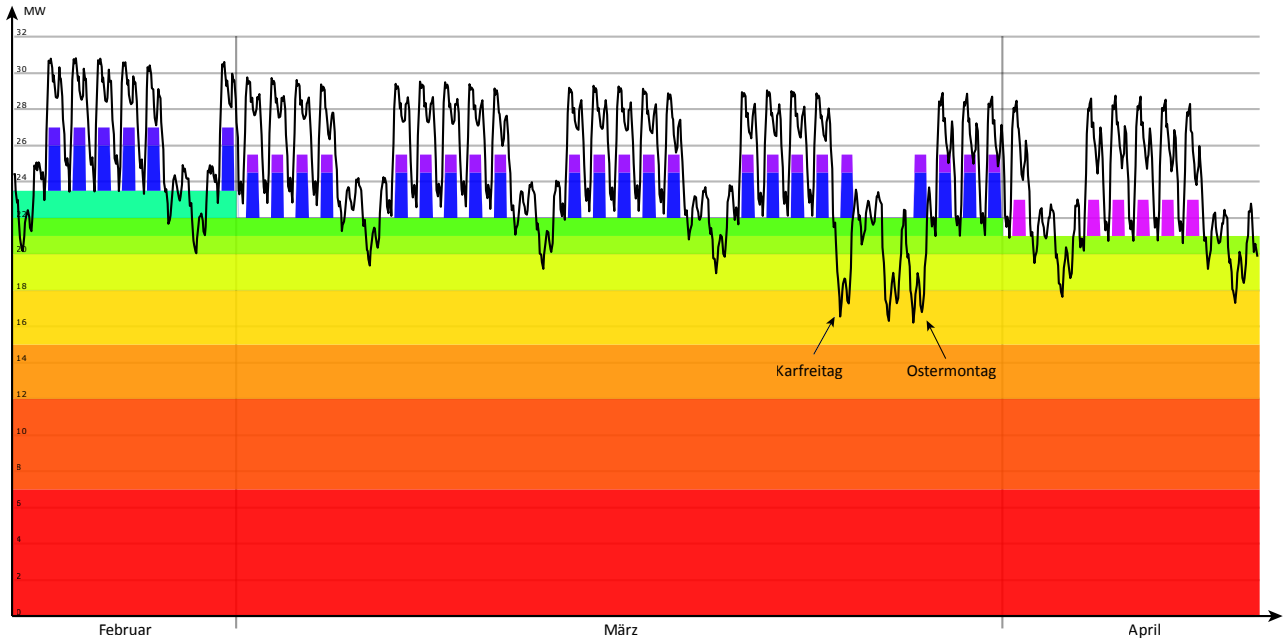

Abbildung 3: Detailliertes Beschaffungsportfolio für die Kalenderwochen 8 bis 14

Das EVU verwendet Langfristprognosen, um abzuschätzen wie viel Strom benötigt wird, aber drei Jahre vor der tatsächlichen Lieferung, kann die Prognose nicht sehr präzise sein. Es ist gut möglich, dass im Laufe der Zeit noch neue Kunden hinzukommen. Je näher der Zeitpunkt der Lieferung kommt, umso präziser wird die Prognose für die benötigte Energie. Im Quartal vor der Lieferung wird dann die Eindeckung durch Kauf und Verkauf von Standardprodukten nochmals nachjustiert (Phase b). Am Tag vor der Lieferung wird eine präzise Kurzfristprognose erstellt. Am Spotmarkt, wo die kurzfristigen Geschäfte gemacht werden, wird nun die Differenz zwischen der bereits beschafften Energie und der mit der Kurzfristprognose vorhergesagten Energie ausgeglichen (Phase c). Nun gibt es üblicherweise eine Abweichung zwischen der prognostizierten und der tatsächlich konsumierten Energie – die Ausgleichsenergie (Phase d).

Im Rahmen eines KTI-Projekts[1] haben wir eine Software entwickelt, welches den eben beschriebenen Beschaffungsprozess in den Phasen (a) und (b) mit den folgenden Funktionalitäten unterstützt:

- *Visualisierung der Beschaffungen:* Die in Abbildung 1 dargestellte Eindeckung mit einem Base- und einem Peak-Band ist stark vereinfacht. Tatsächlich wird die Energie in mehreren Tranchen beschafft. In Abbildung 3 sind die separaten Beschaffungen der Kalenderwochen 8 bis 14 in unterschiedlichen Farbtönen dargestellt. Mit den steigenden Temperaturen im Frühling sinkt der Energiebedarf. Entsprechend ist der eingedeckte Base-Anteil im März um 1.5 Megawatt tiefer als noch im Februar und im April um ein weiteres Megawatt reduziert. Die starken Abweichungen zwischen der Prognose und der Beschaffung Ende März sind auf die Feiertage der Ostern zurückzuführen. Zudem werden diverse Kennzahlen, wie die total beschaffte Energiemenge und verschiedene Indikatoren der Abweichungen zwischen Prognose und Beschaffung gerechnet, um eine detaillierte Einschätzung der Beschaffungssituation zu ermöglichen.
- *Periodische Überprüfung der Marktsituation und Notifikation:* Die Standardprodukte werden an der *European Energy Exchange*[2] gehandelt. Um keine Marktchance zu verpassen oder eine böse Überraschung zu erleben, lädt das Programm stündlich die neuesten Marktdaten für die relevanten Produkte und analysiert diese. Werden konfigurierbare Grenzwerte unter- bzw. überschritten, wird mit einem E-Mail die für den Energiehandel verantwortliche Person kontaktiert.
- *What-If Analyse für die Nachjustierung:* Im Quartal vor der Bereitstellung der Energie wird die bereits beschaffte Energie durch Kauf- und Verkauf von Standardprodukten nochmals anhand einer aktualisierten Langfristprognose des Verbrauchs angepasst. Dabei muss entschieden werden, wie viele Megawatt Base oder Peak gekauft bzw. verkauft werden sollen. Die aktuelle Eindeckung, die aktuellste Prognose und die Preise für die Standardprodukte sind bekannt. Unbekannt aber sind die Preise am Spotmarkt. Diese werden erst am Tag vor der Lieferung gemacht. Energie, die in

1 Regelung von virtuellen elektrischen Speichern, KTI-Nr. 17307.1 PFEN-ES

2 http://www.eex.com

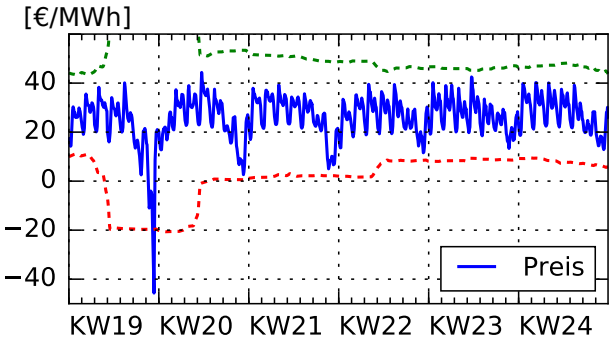

Abbildung 4: Energiepreise am Spotmarkt KW 19 bis 24

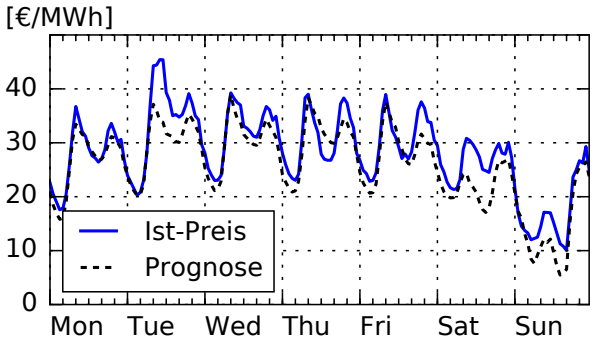

Abbildung 6: Vergleich Ist-Preis / Prognose der KW 25

dieser Phase nicht beschafft wird (Short Position), muss später am Spotmarkt zu einem unbekannten Preis eingekauft und überschüssige Energie (Long Position) verkauft werden. Um zu entscheiden welche Geschäfte jetzt getätigt werden sollen, ist eine Prognose für die Preise am Spotmarkt notwendig.

- *Kostenminimale Nachjustierung:* Basierend auf der Langfristprognose des Energiebedarfs, der aktuellen Eindeckung, der Preise am Terminmarkt und einer Preisprognose für den Spotmarkt wird ein mathematisches Kostenmodell für die Nachjustierung entwickelt. Mithilfe einer Optimierungssoftware kann nun anhand dieses Modells diejenige Nachjustierung berechnet werden, welche in der Gesamtbetrachtung die geringsten Kosten verursacht.

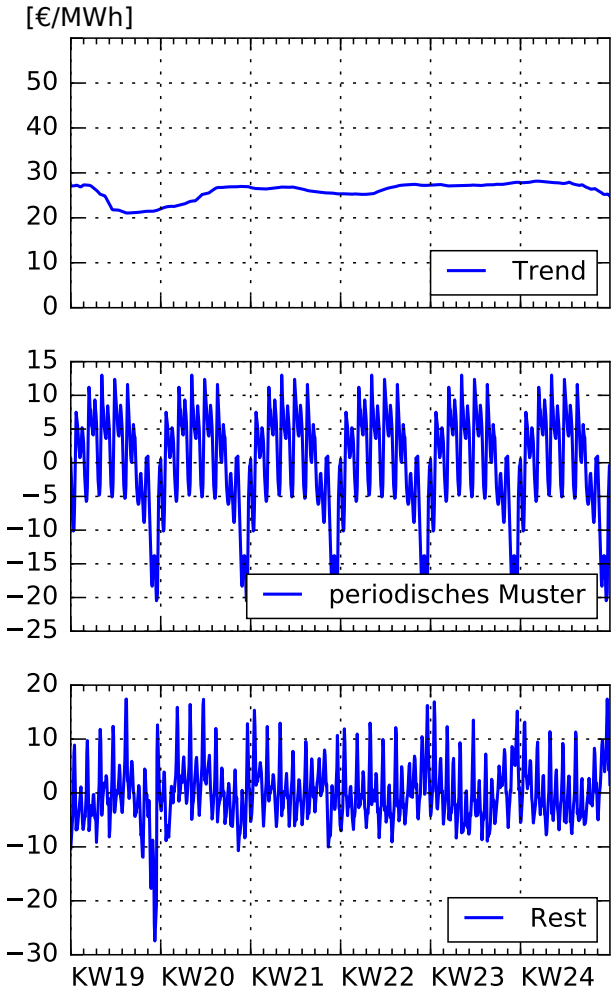

Abbildung 5: Spotpreis Dekomposition

Spotmarkt Prognose

Abbildung 4 zeigt die Preise am Spotmarkt in der Einheit €/MWh über die sechs Kalenderwochen 19 bis 24. Die Daten sind stundengenau aufgelöst. Erkennbar sind ein sich wöchentlich wiederholendes Muster sowie ein Ausreisser am Sonntag der KW 19. Tatsächlich erhielt man an diesem Sonntag zu jeder abgenommenen Megawattstunde noch über 40 Euro dazu. Ursache war eine massive Überproduktion bedingt durch Wind und Sonne.

Börsenkurse lassen sich im Allgemeinen nicht genau vorhersagen. Aber sich wiederholende Muster sollen natürlich bei einer Prognose nicht ausser Acht gelassen werden. Dazu wird die Zeitreihe der Spotpreise in drei Zeitreihen zerlegt, die punktweise addiert wieder der ursprünglichen Zeitreihe entspricht:

$$Spotpreis_t = Trend_t + Muster_t + Rest_t$$

Der *Trend* entspricht der langfristigen Entwicklung der Reihe. Das periodische *Muster* ist das gesuchte charakteristische Wochenprofil und der *Rest* ist das, was übrigbleibt. Vor der Zerlegung werden die Ausreisser ausserhalb eines gleitenden Durchschnitts +/− 3 Standardabweichungen abgeschnitten. Dann wird mittels einem gleitendem Durchschnitt (Fenstergrösse: 1 Woche) die Trendkomponente extrahiert und vom Ist-Preis abgezogen (Abbildung 5 Trend). Aus den trendfreien Daten wird nun das periodische *Muster* extrahiert, indem für jeden Wochentag und jede Stunde die Daten gemittelt werden. D.h. für jeden der sechs Montage werden die Werte um 0:00 Uhr gemittelt. Dann die sechs Werte der nächsten Stunde. Abbildung 5 zeigt die Dekomposition der in Abbildung 4 dargestellten Spotpreise. Zur Prognose der Spotpreise für die KW 25 wird nun dieses sich wiederholende *Muster* im einfachsten Fall zu einem gemittelten *Trend* hinzuaddiert. Da die einzelnen

Komponenten nun isoliert vorliegen, lassen sich damit leicht auch komplexere Spotpreisszenarien modellieren. Abbildung 6 zeigt die Prognose und den tatsächlichen Preis. Mit der Spotpreisprognose sind jetzt alle Parameter für eine Bewertung einer Nachjustierung der Beschaffung gegeben.

Nachjustierung

Im obersten Graphen der Abbildung 7 ist zu sehen, dass in der KW 25 laut Prognose jeweils nachts und das ganze Wochenende zu viel Energie beschafft wurde. Dafür fehlt Energie zu den Spitzenlastzeiten. Wie wäre die Situation, wenn das durchgehende Base-Band um −3 MW abgesenkt und das Peak-Band um +7 MW erhöht würde? Wie wäre die Abdeckung und was würde da kosten?

Das What-If-Analyse-Tool rechnet für solche Szenarien die wichtigsten Kennzahlen und visualisiert das Resultat. Abbildung 7 zeigt die Situation vor der Nachjustierung, die geplante Anpassung und die Situation nach der Anpassung. Nach der Justierung wäre die prognostizierte Energie deutlich besser abgedeckt. Aber welche finanziellen Dimensionen hat dieser Handel? Einerseits wird Base verkauft und Peak gekauft, andererseits muss die Differenz nach der Justierung noch am Spotmarkt ausgeglichen werden.

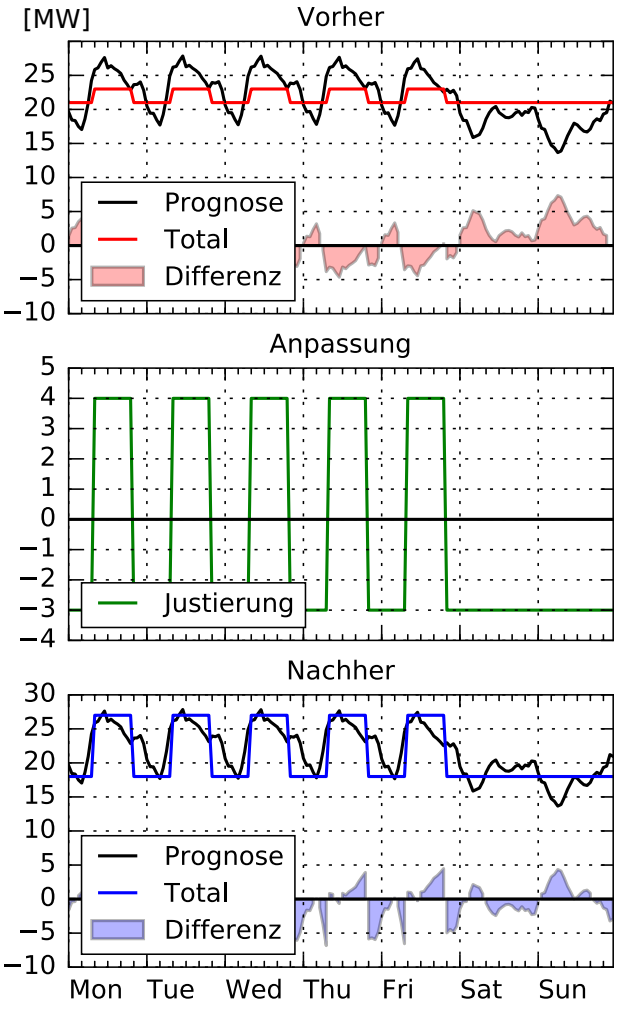

Abbildung 7: Nachjustierung [Vorher, Anpassung, Nachher]

	Kosten [€]	Kaufen [€]	Verkaufen [€]
Nachjustierung	588	14'700 (7 MW Peak über 5×12h zu 35 €/MWh)	14'112 (3 MW Base über 7×24h zu 28 €/MWh)
Spot	6010	8653 (232 MWh Short)	2643 (130 MWh Long)
Total	6598		

Tabelle 1: Auswirkungen einer Nachjustierung von -3MW Base und +7MW Peak

Die What-If Analyse berechnet nun die Auswirkungen der angedachten Nachjustierung. Am Terminmarkt werden 7 MW Peak-Band zu einem Preis von 35 €/MWh gekauft. Wie bereits erwähnt, deckt das Peak-Band nur die fünf Arbeitstage jeweils von 8:00 – 20:00 Uhr, also 12 Stunden ab. Benötigt werden also 5×7×12 MWh = 420 MWh. Die verkauften 3 MW Base-Band hingegen decken alle 7 Wochentage über die vollen 24 h ab. Die verkaufte Energie beläuft sich also auf 3×7×24 MWh = 504 MWh. Base-Band Energie ist preiswerter – wir rechnen hier mit 28 €/MWh. Die Differenz zwischen der Bedarfsprognose und der nachjustierten Beschaffung (Abbildung 7: Nachher / Differenz) wird nun am Spotmarkt zu den prognostizierten Preisen ausgeglichen. Für obiges Szenario müssen 232 MWh dazugekauft und 130 MWh verkauft werden. Tabelle 1 gibt einen Überblick über die finanziellen Konsequenzen der vorgesehenen Nachjustierung.

Kostenminimale Nachjustierung

Die What-If Analyse berechnet die Auswirkung einer angedachten Nachjustierung. Aber welche Nachjustierung, wie viel Base- und Peak-Band müsste man für minimale Kosten wählen?

Anstatt manuell nach einer guten Lösung zu suchen, wird ein mathematisches Modell definiert, welches optimal gelöst werden kann. Das Ziel dabei ist die Minimierung der total anfallen-

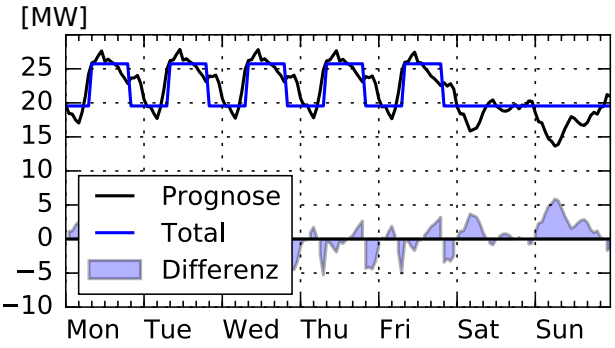

Abbildung 8: Optimale Nachjustierung

	Kosten [€]	Kaufen [€]	Verkaufen [€]
Nach-justierung	1963	8805 (4.2 MW Peak über 5×12h zu 35 €/MWh)	6843 (1.5 MW Base über 7×24h zu 28 €/MWh)
Spot	3934	6055 (150 MWh Short)	2121 (139 MWh Long)
Total	5897		

Tabelle 2: Auswirkungen der optimalen Nachjustierung

den Kosten *costTotal*. Diese setzen sich aus den Kosten für die Nachjustierung *costFutures* und den Kosten zusammen, die danach am Spotmarkt anfallen *costSpot:*

costTotal = *costFutures* + *costSpot*

Die Energiemenge für die Nachjustierung entspricht der gewünschten Leistung mit dem entsprechenden Lieferprofil (*profileBase* bzw. *profilePeak*) multipliziert. Dies sind Zeitreihen mit den entsprechenden Lieferzeiten:

amountBase = *powerBase* × *profileBase*
amountPeak = *powerPeak* × *profilePeak*

Die Kosten für die Nachjustierung berechnen sich aus der Menge multipliziert mit dem Preis:

costFutures =
Σ (*amountBase* × *priceBase*) +
Σ (*amountPeak* × *pricePeak*)

Die Kosten am Spotmarkt setzen sich zusammen aus der Menge der auszugleichenden Energie *amountSpot* und der Spot-Preis-Prognose *spotPricePred*:

costSpot = Σ (*amountSpot* × *spotPricePred*)

Die Menge der Energie, die am Spotmarkt ausgeglichen werden muss, ist die Differenz zwischen der Langfristprognose *prediction* und dem *total* nach der Nachjustierung:

amountSpot = *prediction* −
(*total* + *amountBase* + *amountPeak*)

Ein Solver kann nun automatisch nach einer Variablenbelegung für *powerBase* und *powerPeak* suchen, so dass *totalCost* minimal wird. Für die KW 25 findet der Solver folgende optimale Lösung: Kauf von 4.2 MW Peak-Band und Verkauf von 1.5 MW Base-Band. Tabelle 2 gibt eine Übersicht über die entstehenden Kosten und Abbildung 8 zeigt die Beschaffungssituation nach dieser optimalen Nachjustierung. Mit dieser optimalen Nachjustierung spart man rund 700 € in der KW 25 gegenüber der von Hand geschätzten Nachjustierung. Auf das Jahr betrachtet sind das rund 40’000 Franken. Die Resultate basieren auf der Prognose der Spotmarktpreise und sind entsprechend mit Vorsicht zu geniessen.

Das Modell kann nun weiter verfeinert werden. Beispielsweise könnte man einschränken, dass am Spotmarkt maximal 100 MWh Energie dazugekauft werden müssen, um das Risiko steigender Preise zu minimieren. Tatsächlich ist das realisierte Modell deutlich komplexer als das oben skizzierte. Es unterstützt zusätzlich separate Kauf- und Verkaufspreise für die Standardprodukte und den Spotmarkt sowie Parameter zur Kontrolle des Risiko-Ertrags-Verhältnisses.

Fazit und Ausblick

In diesem Bericht haben wir aufgezeigt, wie die Bewirtschaftung des Energieportfolios eines EVUs mittels What-If Analysen und mathematischer Optimierung unterstützt werden kann.

Die Spotpreisprognose könnte durch den Einbezug weiterer Einflussfaktoren wie zum Beispiel Ölpreise oder Kennzahlen zur Wirtschaftsleistung und dem Einsatz von Machine-Learning Algorithmen weiter präzisiert werden. Schliesslich liesse sich das System so weiterentwickeln, dass es selbständig die unterschiedlichsten Szenarien simuliert und basierend auf einer Risikobewertung zu den günstigsten Zeitpunkten automatisch Handelsempfehlungen vorschlägt.

aWall: Agile Collaboration using Large Digital Multi-Touch Cardwalls

Despite the availability of many digital agile board tools, most co-located agile software teams still use physical cardboards for their daily standup meetings. This is due to the fact that existing digital agile boards lacks supporting a collaborative workspace, direct interaction for the whole team in meetings, or making project information directly visible. In this paper we present aWall, a digital agile cardwall designed for the highly collaborative agile work style using large multi-touch wall displays. The effectiveness of aWall was evaluated in a user study with eleven software practitioners. Our findings indicate that aWall enables and encourages team work due to the large size of the wall, accessibility and visibility of large amounts of information, and possibility of customization of the interface. Based on this work, we suggest that augmenting digital cardwalls with large interactive touch technology and new interaction concepts is a useful way to support effective collaborative agile software development processes.

C. Anslow, R. Burkhard, M. Kropp, M. Mateescu, D. Vischi, C. Zahn | martin.kropp@fhnw.ch

In agile software development, physical cardwalls continue to be an essential part of the agile processes despite the relative large number of available digital tools. Although digital cardwall tools like JIRA [3] and VersionOne [12] are commercially available and have been adopted by a large number of agile companies, some studies show [4, 8, 9]. Azizyan et al. conducted interviews with software practitioners and found that 31% of companies used both project management tools and physical cardwalls, where the usage of the cardwalls was not restricted to co-located teams [4]. Mateescu et al. found that 10 out of 11 teams still use physical cardwalls typically in combination with digital tools [9]. Despite their prevalence, physical cardwalls still have issues as content is not digitalized and not integrated with issue tracking systems. To address the issue with physical cardwalls, we aim to bridge the gap by creating a large digital cardwall that supports elements of the physical nature, integration with existing tracking systems, while also preserving the agile collaborative work style.

In this paper we present aWall, a digital agile cardwall designed for use by co-located and distributed teams (see Figure 1). aWall has the size of classical physical cardwalls by using large multitouch high resolution displays and so provides enough space for the whole team to interactively collaborate. We first give an overview of related work, followed by the design and user interface of aWall. We then present a user study conducted with aWall and software practitioners to evaluate the usability and effectiveness of the design. The paper concludes with a final summary and future work.

Related Work

Physical artifacts like pin boards, sticky cards, flip charts or whiteboards are used as a means of communication and collaboration by agile teams [15]. The physical nature of artifacts is important to the collaborative process. For example being able to manipulate the cards easily (writing and posting) and their permanent availability on the cardwall helps support effective communication at least in co-located teams. Some studies show that physical cardwalls are valued for their flexibility, light-weight and easy usage, providing a big picture, and permanent and instant availability of information [8, 9]. Information can be concurrently edited during meetings and team members can see who is changing what. Cardwalls help to foster awareness and transparency in teams by acting as information radiators [6]. Paredes et al. conducted a survey of existing literature on information visualization techniques used by agile software development teams and found that information radiators and cardwalls are most frequently used for agile teams in communication and progress tracking [13].

There are some disadvantages of physical artifacts such as cards may get lost and they cannot be searched or shared easily [15]. Physical cardwalls are not well suited for distributed environments and displaying large amounts of information is difficult [8]. A common practice is to put extra information around the core cardwall content [9]. Any attempt to overcome these disadvantages by digitalizing cards and cardwalls should retain the advantages of the physical form while also benefiting from translation to the digital medium [15].

Digital agile tools lack the support for social interaction and team cognitive activities compared with physical tools. Several commercial digital tools exist to support the collaborative process in agile teams, such as *JIRA Agile* and *VersionOne*. These tools have been reported to account for less than 10% of tools used to support agile processes, meanwhile physical walls, paper, and spreadsheets account for almost 50% [4].

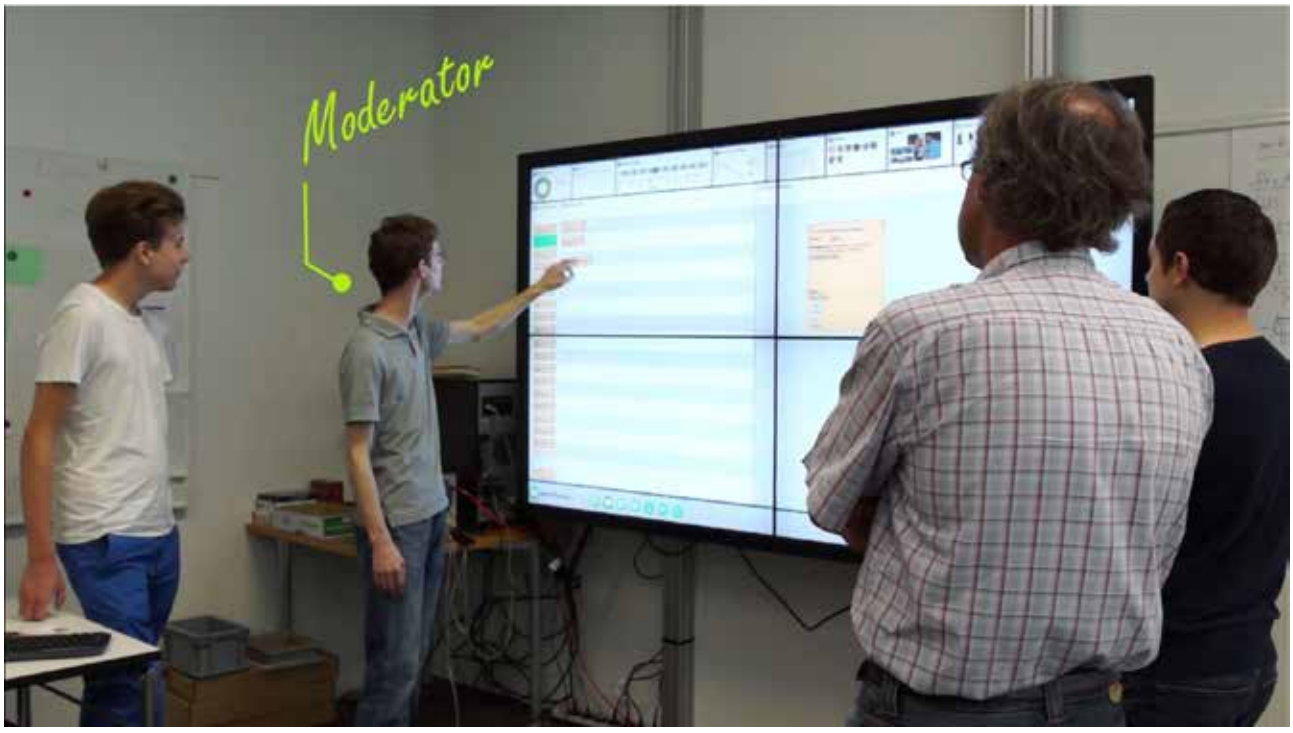

Figure 1: aWall – digital agile cardwall displayed on a large high resolution multi-touch wall (2x2 46" HDready displays) for planning and agile team meetings

A number of digital research tools have been developed for use on large interactive surfaces (e.g. horizontal and vertical). *DAP* [10] and subsequently *AgilePlanner* [16] were early prototypes developed to support agile planning on horizontal tabletops for co-located teams. *SmellTagger* supports collaborative code reviews for co-located teams using multi-touch tabletops [11]. *CodeSpace* uses shared touch screens, mobile touch devices, and *Kinect* sensors to share information during developer meetings but does not focus on any particular agile process [5]. Anslow et al. [2] evaluated large display walls for collaborative software visualization. *SourceVis* used large multi-touch tabletops to support code reviews using collaborative visualization techniques [1]. Rubart developed a basic prototype for multi-touch tabletops to support Scrum meetings [14]. *dBoard* is a Scrum board on a vertical touch screen with video capabilities for distributed development [7]. Based on our review we conclude that most digital agile tools only partially support collaborative agile processes and meetings.

Essentially, commercial or research digital tools do not sufficiently support the collaborative agile process effectively [8, 9]. Users value the traceability of information in digital tools, linking possibilities of artifacts, and the flexibility to adapt the tools to the users' needs. The main disadvantages of digital cardwall tools are that they are often too complicated to use, need to navigate to information and extra steps for operation, and no direct and concurrent interaction by all team members [8, 9]. The focus of most digital cardwalls has been centered on the daily standup meeting, and lack support for the whole agile process including other agile meetings and activities like sprint planning, retrospectives, and user story groomings [9]. Due to these shortcomings we have developed a digital cardwall tool to support collaborative agile meetings more effectively.

aWall – Digital Agile Cardwall

To understand how agile teams use cardwalls in practice, we conducted a field study and interviewed 44 participants from eleven companies [9]. When asked about the requirements for a digital agile cardwall, the interviewees stressed the importance of non-functional requirements. These included the need for a large size display, configurable views, instant availability of information, overview of information, at all time visible information, within easy reach context dependent information, increased readability of information, multi-user simultaneous touch interaction, direct interaction with data, and limited navigation. Our hypothesis is that existing digital tools do not adequately support the communication and collaborative aspects for agile team meetings effectively.

Based on our study and hypothesis we developed aWall to support agile teams (co-located or distributed) more effectively than existing physical and digital tools. aWall helps support agile team meetings (e.g. daily stand up, sprint plan-

ning, and retrospectives) by providing information dashboards, maintaining user stories and tasks, showing dependencies among user stories, customization of agile processes, and integration with issue tracking systems. aWall was developed by an interdisciplinary project team of computer scientists and psychologists (from the School of Engineering, and the School of Applied Psychology). We now outline the design and user interface of aWall, followed by a user study.

Design

Based on the requirements elicited during the interviews, we identified a number of design considerations.

- *Physical Size:* A digital cardwall needs to satisfy not only the needs for interacting with the digital content, but also provide enough physical space to display information to effectively support team collaboration. Therefore, the size of a digital cardwall needs to be at least comparable to that of physical cardwalls. aWall consists of four (2×2) 46" displays, for a wall size of 2.05 m width and 1.25 m height (see Figure 1).
- *High Resolution:* Each display in aWall is 3840×2160 pixels, for a total resolution of 15360×8640 pixels. The high resolution display wall provides enough real estate to display large amounts of information at once while still ensuring the readability of text elements, widgets, and views.
- *Multi-User and Multi-Touch:* The display wall consists of a 12 point multi-touch infrared optical overlay (PQ Labs frame[1]) which is attached to the display wall. The multi-touch capabilities allow multiple users to work simultaneously with artifacts and provides an accurate and effective touch experience.
- *Integration with Issue Tracking Systems:* aWall is designed to run on top of existing third party issue tracking systems such as JIRA. Therefore, infrastructure functionality can be reused and already defined agile processes utilized.
- *Availability of Information:* aWall can replace physical cardwalls and act as the team's external memory of the project. For that, aWall should be installed in a team's open office area, always being switched on, and have a permanent view of the task board.
- *Web Technologies:* In order to have a ubiquitous and easily deployable design, aWall was developed as a web application based on HTML5 and JavaScript technology. For multi-touch support we used the *interact.js* framework[2].

1 http://multitouch.com/
2 http://interactjs.io/

User Interface

The aWall user interface contains a number of different views, widgets, and interaction techniques designed to support different types of agile meetings.

- *Action and Information View:* The results of the interviews showed that most interaction with the cardwall takes place during agile meetings. Each meeting has specific goals, operates on different data, and requires different supporting tools and information. To support these different types of information handling, we divide the display into an action view and an information view. Figure 2 shows the view for a daily standup meeting highlighting the separation into information view and action view. The action view is the main work area, which is dedicated to the core artifacts of a specific meeting. The main interactions during a meeting are performed by users on the action view. The information view provides supporting information and tools needed for the meeting. The information view represents the dynamic memory of the team and as any dynamic system they need to allow for change. For example, the information view for the daily standup meeting contains additional information, like a timer widget showing the meeting moderator and a countdown, a team widget showing the team members, a definition-of-done widget, an impediment list widget, and a burndown chart for an iteration. When necessary, new widgets can be added and removed from the information view.
- *Dedicated Views:* aWall provides dedicated views that are tailored to the specific needs of agile meetings. For the sprint planning meeting shown in Figure 3, the action view is divided into three columns. The left column shows the top priority user stories of the product backlog. The center column shows the so far selected user stories for the next iteration. The right column shows a detailed view of the currently selected user story. This column can be used by the product owner to discuss and clarify open issues during the meeting with the development team. Relevant documents can be easily attached and opened in the application. Figure 4 shows the retrospective meeting view after team members have sent their iteration feedback where the notes have been ordered on the right side. Users can navigate between the different meeting views by means of a navigation bar displayed at the bottom of the view.
- *Information Widgets:* The information view consists of a set of widgets (e.g. team widget, timer widget, fun widget, avatar widget (see Figures 2, 3, 4) and can be independently configured for each agile meeting. Each widgets is designed to support distinct aspects of the

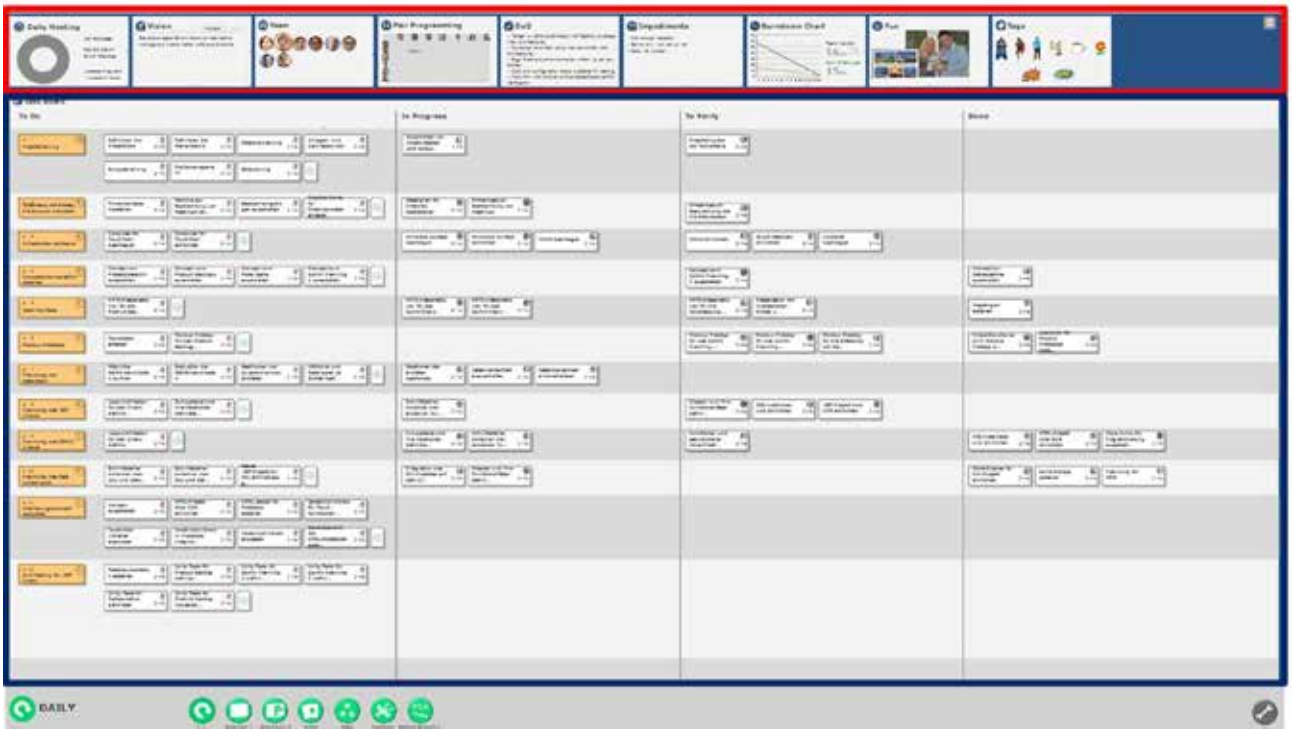

Figure 2: Daily standup with the following views: Information view (top section with red border) and action view (middle section with blue border)

collaborative agile process. The team widget shows the team members and can be used to assign people to tasks during a daily stand-up meeting. The timer widget supports time boxing during the meeting and furthermore, allows to choose a meeting moderator. The moderators' names are stored in the application and future moderators can be suggested based on previous selections. The fun widget allows users to post personal or fun images to the information view to help bring emotion to the cardwall and foster team thinking. The avatar widget can be used to drag avatars to any position on the wall or attach it to tasks or user stories. Both the fun and avatar widgets are designed to help with the interpersonal process in agile teams (emotion management, team spirit). All widgets can be detached from the information view and moved around the cardwall to facilitate user interaction.

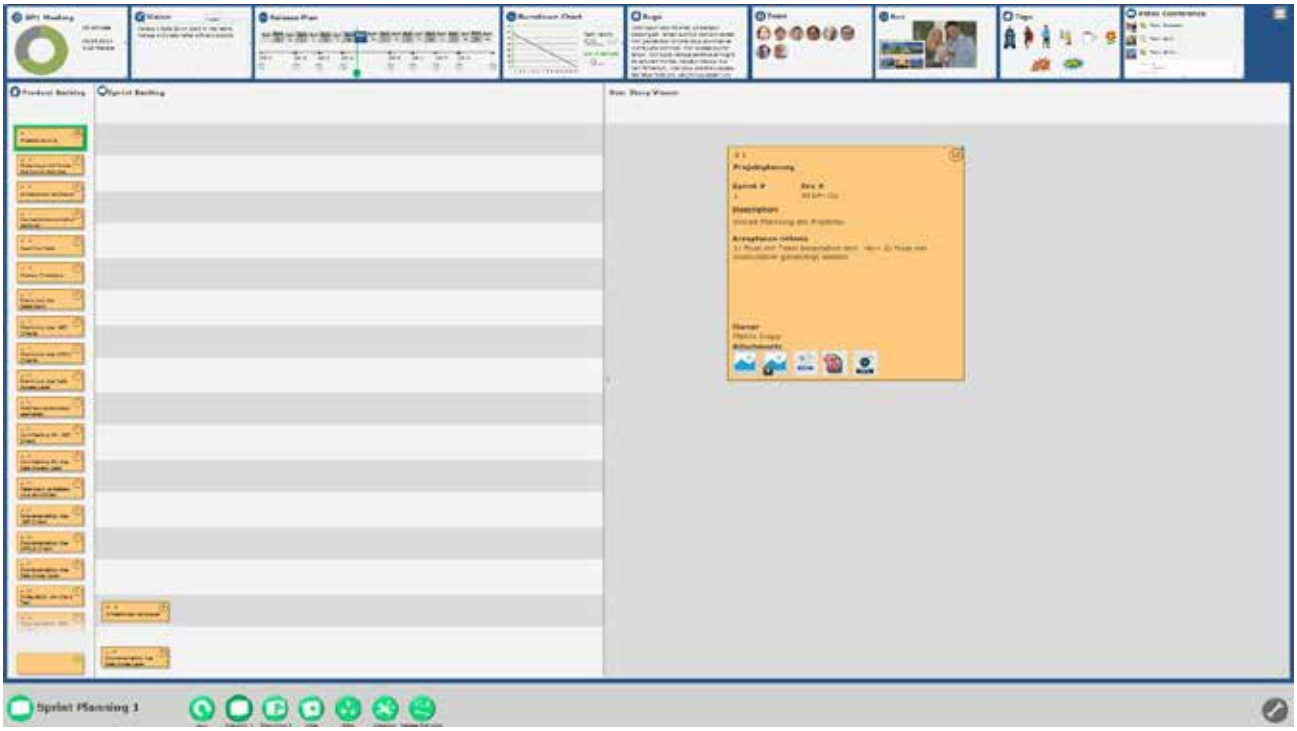

Figure 3: Sprint planning meeting with a user story detail view

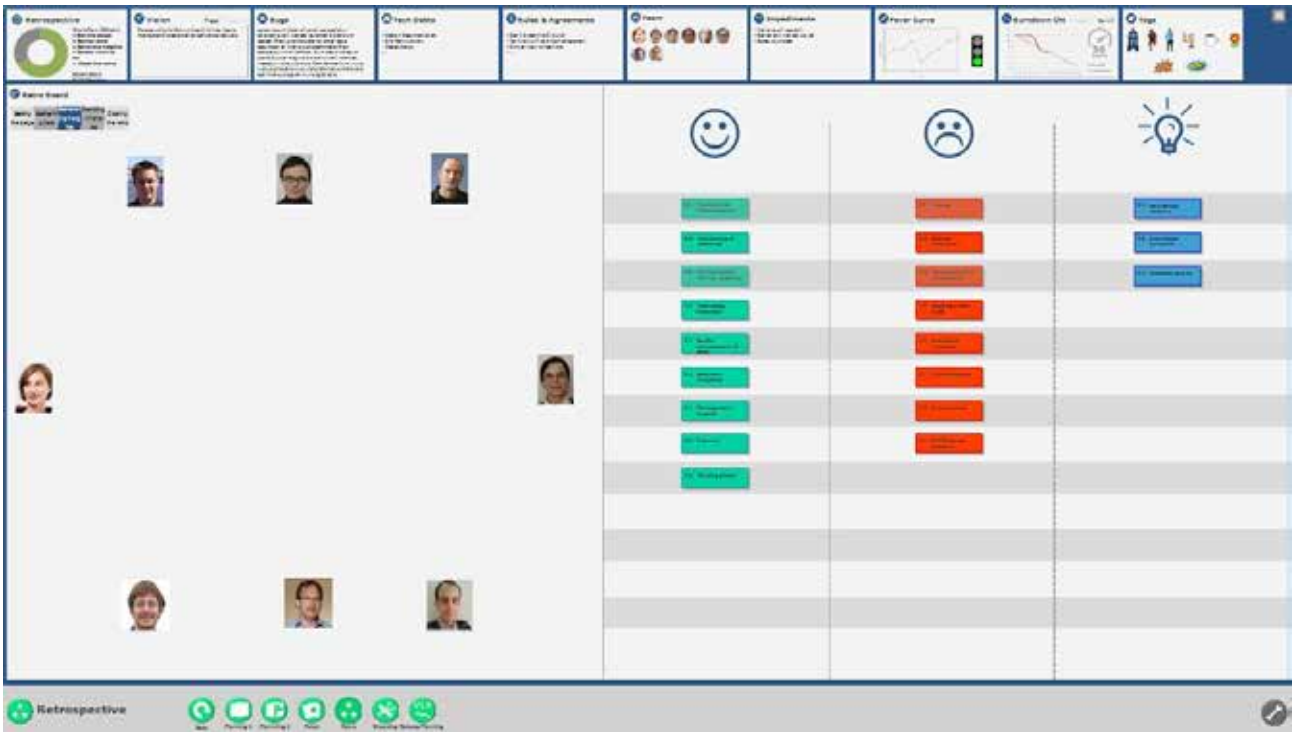

Figure 4: Retrospective meeting view

- *Availability of Information:* Any information needed for a meeting is visible and accessible; either on the action view or on the information view. If the team needs different supporting information, additional widgets can be switched on or off in the configuration button on the right side of the information view.
- *Interaction:* aWall supports multi-touch and multi-user interaction. Fluid interaction with widgets and cards is enabled by gestures like tap, double tap, drag-and-drop, and pinch-to-zoom supporting changing task and user story cards position, moving widgets around the cardwall, and changing the size of a widget. Data can be either entered on the cardwall with a virtual or physical keyboard or via the underlying issue tracker system and mobile devices such as tablets.
- *Scalability of Information:* By default, user stories and tasks cards show only a few details (e.g. title). By increasing card size with a pinch-and-zoom gesture more information is displayed. The text size increases concomitantly with the widening of the cards so that information can be more easily read depending on the distance from the cardwall. When all information is shown the widget automatically switches into edit mode, so that data can be added or modified.

User Study

To evaluate the design of aWall, we conducted a user study with professional agile practitioners. The main focus of the study was on the usability and discoverability of functionality, support of agile workstyle, and applicability to real life situations in agile teams performing the daily stand up and sprint planning meetings. The user study was conducted with an early aWall prototype where participants had to complete various tasks with the aWall working in groups.

We recruited eleven employees (nine men and two women – see Table 1) from the same companies that participated in our interview study [9]. Most participants had many years of experience in IT, and several of them in agile development. They came from different fields and covered a wide spectrum of agile team roles. Among the participants were four Scrum Masters, two agile coaches, two senior developers, one agile grandmaster, one UX consultant and one head of a software development department. Two of the companies were from the assurance domain, one manufacturing, two service providers, one engineering, and one enterprise software development company. Four companies sent two employees, and three companies sent one employee each. All companies had been applying agile processes for at least one year.

Procedure

We divided the eleven participants randomly into two groups. Both groups completed the same tasks with the aWall. Upon signing an informed consent statement, the participants were asked to act as a team during the workshop. Prior to the user study, the participants received a presentation on the interview study results, but did not receive any information about the aWall application. Each participant received three tasks to be

Gender	IT Exp	Agile Exp	Job Title	Company	Group
male	23	3	Head SW Dev	D	1
male	5	1.5	Senior Dev	E	1
male	13	2	Grandmaster	C	1
male	10	3	Agile Coach	F	1
male	19	4	Senior Dev	G	1
male	10	3	UX Consultant	B	1
female	8	3	Agile Coach	C	2
female	15	5	Scrum Master	A	2
male	15	3	Scrum Master	A	2
male	1	1	Scrum Master	E	2
male	6	2	Scrum Master	F	2

Table 1: Demographics of workshop participants: gender, IT experience, agile experience, job title, company (anonymized), and workshop group

solved together in groups using aWall. The tasks involved a daily standup meeting and a sprint planning meeting. After receiving the task, each participant read the task out aloud to the other participants and completed it with their help. The daily standup task was to start the daily standup meeting, choose a moderator for the meeting, and update the task board during the meeting. For example: *"In this team you play the role of team member M. Please find a way to carry out a daily standup. The application suggests a moderator. Please ask the team member suggested by the application to play the moderator. Please act as a team accordingly to the received instructions."*

The sprint planning task was to show and discuss a user story during the meeting and move the story to the sprint backlog. The third task was to decide upon how to conduct the retrospective meeting. After completing the tasks for each type of meeting the participants discussed the benefits and disadvantages of aWall for that type of meeting together with the two moderators. The discussions were recorded and the results written down. Both workshops were conducted by two moderators and lasted one hour each.

Findings

The overall feedback for the prototype was very positive, with the participants considering aWall to be usable, capable to support agile processes in general and especially the collaborative working style in teams.

- *Size Aspects:* The participants especially valued the large size and high resolution of aWall. The large size supports real team collaboration capabilities, similar to physical cardwalls. Displaying a large amount of information at once was deemed positive. One participant stated: *"With the large size you can display many user stories and tasks."*
- *Readability of Information:* Most participants considered the displayed information to be legible, especially since the card titles are relatively large. Some participants considered the actual cards to be too small. Therefore, it is very important to be able to display the whole content of a card and enlarge the font size so that the whole team can read it from a distance. One participant stated: *"That's really a nice feature that cards can be enlarged and font size increases to improve readability."*
- *Availability of Information:* The participants especially valued the availability of additional information and functionality for the different meetings. The separation of the display into action view and information view was easily understood. Some participants mentioned that elements placed on the upper side of the display wall might be out of reach for smaller people. Another participant liked the extra features: *"I like the extra features around the main view and the additional information."*
- *Discoverability of Functionality:* The participants discovered most functionality of aWall by themselves and could easily interact with the display wall. There were some issues with discoverability of those functions that were not a straight-forward transfer of the pin-boards into the digital world. For example, the timer widget has no corresponding artifact in the practice of agile teams. Whereas, direct implementations of the pin-boards functionality (e.g. the task-board shown in the daily standup meeting) were instantly understood and deemed as valuable by the participants. That was also the case for the widgets inspired from agile practices such as the team widget which is based on the observation that agile teams sometimes write the team members' names on the cards or even hang their pictures on the pin-boards.
- *Third-Party System Integration:* The integration with third-party tools was positively rated. Tasks modified during the daily standup meeting, are immediately synchronized in the agile project management tool (JIRA). There is no extra effort to update the tasks manually from the physical cardwall after the meeting. One participant stated: *"The link to JIRA with automatic update of data is important."*
- *Flexibility and Customization:* Increased flexibility with respect to both the manner of conducting the meetings and displaying information was considered important by the participants. For example, the timer widget solicited choosing a moderator at the beginning of a meeting. The flexibility provided by aWall

was also positively rated, especially with respect to conducting retrospective meetings that sometimes might prove strenuous. The participants considered that it is important to create a proper environment especially for this type of meeting as sometimes they tend to transmute into a drill. Most participants were in favour of a greater flexibility of the time boxing, with only optionally choosing a moderator and not showing the elapsed time, but the time of day during the daily meeting. The participants valued the team widget, but requested to have more information being displayed (e.g. absences, vacation days) and allow for more customization. Furthermore, the participants remarked that they should be able to add functionality to aWall on their own and not be dependent on standard functionality as often is the case with other agile tools.

- *Agile Collaborative Workspace:* Offering tags and avatars as well as the fun view was positively seen as bringing emotions onto the board. One participant mentioned the positive effect of avoiding of media disruption, by being able to do all interaction with only one medium: *"With such a board we could probably avoid media discontinuity."*
- *Filtering and Representation of Information:* The participants requested especially to have filter functions, to highlight and show the desired information. As an example, participants requested to highlight all tasks of a team member, when touching that person in the team view. The usage of colors for different types of user stories was suggested to increase readability (e.g. to distinguish between technical tasks, bug reports, or user requirements).
- *Task Time Recording:* Some participants suggested automatically capturing the time spent on a task combined with computing of the work hours on the task would help provide further metric details of performance.
- *Provenance of Information:* Some participants suggested having automatic recordings of meetings with voice recognition and transcriptions of the discussions form the interactions in front of the display wall for later recollection and analysis of the meetings.

Conclusions and Acknowledgments

Current agile cardwalls don't fulfil today's requirements for effective software development. We aim to bridge that gap with aWall, a digital cardwall tool to support co-located and distributed agile teams. aWall provides a collaborative workspace using large multi-touch displays, information transparency, direct information interaction without the need for navigation, support for the whole agile process, and dedicated views for different types meetings. We conducted a user study with eleven agile practitioners and found that they especially valued the large size of the wall due to the physical space affordances, the dedicated views with context specific information, and the always visible and direct information access. Our future work involves deploying aWall within companies.

Thanks to the University of Applied Sciences and Arts Northwestern Switzerland for funding this project as part of their strategic initiative to fostering interdisciplinary work. Thanks to the companies and people for participating in the interviews field study and user study. This research was supported by project VALCRI, European Commission Grant FP7-IP-608142. Thanks to Robert Biddle for feedback on early drafts of this paper.

References

[1] C Anslow, S. Marshall, J. Noble, and R. Biddle. Sourcevis: Collaborative software visualization for co-located environments. In VISSOFT, pages 1-10. IEEE, 2013.

[2] C. Anslow, S. Marshall, J. Noble, E. Tempero, and R. Biddle. User evaluation of polymetric views using a large visualization wall. In SoftVis, pages 25{34. ACM, 2010.

[3] Atlassian. Jira, 2015. https://www.atlassian.com/software/jira.

[4] G. Azizyan, M. K. Magarian, and M. Kajko-Matsson. Survey of agile tool usage and needs. In AGILE, pages 29-30. IEEE, 2001.

[5] A. Bragdon, R. DeLine, K. Hinckley, and M. Morris. Code space: Touch + air gesture hybrid interactions for supporting developer meetings. In ITS, pages 212- 221. ACM, 2011.

[6] A. Cockburn. Agile Software Development: The Cooperative Game. Addison-Wesley, 2006.

[7] M. Esbensen, P. Tell, M. Cholewa, J.and Pedersen, and J. Bardram. The dboard: A digital scrum board for distributed software development. In ITS, pages 161-170. ACM, 2015.

[8] S. Gossage, J. Brown, and R. Biddle. Understanding digital cardwall usage. In AGILE, pages 21-30. IEEE, 2015.

[9] M. Mateescu, M. Kropp, Greiwe St., R. Burkhard, D. Vischi, and C. Zahn. Erfolgreiche Zusammenarbeit in agilen Teams: Eine Schweizer Interview-Studie über Kommunikation in agilen Teams, 22 Dec 2015. http://www.swissagilestudy.ch/studies.

[10] R. Morgan, J. Walny, H. Kolenda, E. Ginez, and F. Maurer. Using horizontal displays for distributed and collocated agile planning. In XP, pages 38-45. Springer, 2007.

[11] M. Müller, M. Würsch, T. Fritz, and H. Gall. An approach for collaborative code reviews using multi-touch technology. In CHASE Workshop. ACM, 2012.

[12] Version One. Enterprise agile platform, 2015. http://www.versionone.com.

[13] J. Paredes, C Anslow, and F. Maurer. Information visualization for agile software development teams. In VISSOFT, pages 157-166. IEEE, 2014.

[14] J. Rubart. A cooperative multitouch scrum task board for synchronous face-to-face collaboration. In ITS, pages 387-392. ACM, 2014.

[15] H. Sharp, H. Robinson, and M. Petre. The role of physical artefacts in agile software development: Two complementary perspectives. Interacting with Computers, 21(1- 2):108-116, 2009.

[16] X. Wang and F. Maurer. Tabletop agileplanner: A tabletop-based project planning tool for agile software development teams. In TABLETOP, pages 121-128. IEEE, 2008.

Fachhochschule Nordwestschweiz
Institut für Mobile und Verteilte Systeme
Bahnhofstrasse 6
CH-5210 Windisch

www.fhnw.ch/technik/imvs
Tel. +41 56 202 99 33