Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterseminar

Benjamin Lindemann

Stress am IT-Arbeitsplatz - *Thesis Outline*

17. Februar 2013

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Kurzzusammenfassung

Negativer Stress kann den Menschen belasten und auf Dauer krank machen. Burn-out ist eine mögliche Folge. Am Arbeitsplatz wird Stress immer häufiger durch höheren Zeitdruck und komplexer werdende Problemstellungen ausgelöst. Um den Menschen vor den negativen Folgen von Stress zu schützen, soll ein computergestütztes System entwickelt werden, das Stress automatisch erkennt und den Menschen aktiv bei der Stressprävention unterstützt.

Das Ziel meiner Masterarbeit ist die Entwicklung eines Software-Frameworks, das bei der Messung der körperlichen Reaktionen auf Stress unterstützt und ein Interface zur Anbindung von weiteren Sensoren bietet. Verschiedene Bio-Sensoren können über angebotene Schnittstellen mit dem Framework verbunden werden. Die zentrale Steuerungseinheit akkumuliert in bestimmten Zeitabschnitten die übermittelten Sensordaten und stellt ein globales Stresslevel bereit. Um die Gründe des Stresszustandes zu ermitteln, können zusätzliche Sensoren den aktuellen Arbeitskontext des Menschen überwachen.

Die Implementierung der zentralen Steuerungseinheit des Frameworks erfolgt in Java. Die Anbindung der Sensoren an die zentrale Steuerungseinheit habe ich über eine ActiveMQ realisiert.

Mit Hilfe meines Frameworks kann der Fragestellung „Was ist Stress am Arbeitsplatz und wann tritt er auf?“ nachgegangen werden.

1 Einleitung

Vielfältige Formen von Stress dominieren zunehmend den Alltag des Menschen. Ständiger Leistungs- und Termindruck haben in den letzten Jahren besonders im Arbeitsumfeld enorm zugenommen. Hierdurch bedingt stieg die psychische Belastung kontinuierlich an. Viele Menschen erkrankten an dieser dauerhaften Belastung und wurden sogar arbeitsunfähig. [8]

Um diesem Problem vorzubeugen, muss der Mensch vor zu vielen Stressoren geschützt werden. Stress am Arbeitsplatz könnte durch ein computergestütztes System automatisiert erkannt werden. Das System würde „Stress messen“ und in einer Stresssituation gezielt Maßnahmen zur Stressprävention einleiten. Die ermittelten Messdaten könnten ebenfalls genutzt werden um der Fragestellung „Was ist Stress am Arbeitsplatz und wann tritt er auf?“ nachzugehen.

Das Ziel meiner Masterarbeit ist die Entwicklung eines Software-Frameworks, das bei der Messung der körperlichen Reaktionen auf Stress unterstützt und ein Interface zur Anbindung von weiteren Sensoren bietet. Mein Framework soll in erster Linie eine Verbindungskomponente zwischen den Sensoren darstellen. Die von den Sensoren gemeldeten Stresslevel werden im Framework akkumuliert, um ein globales Stresslevel pro Zeiteinheit wiedergeben zu können.

Die folgende Arbeit unterteilt sich in drei Abschnitte. Zunächst gebe ich einen Überblick über die Problemstellung und führe den Begriff *Stress* ein. Im Hauptteil beschreibe ich mein Konzept des Software-Frameworks und deren Schnittstellen. Abschließend gehe ich auf mögliche Risiken meiner Arbeit ein und deute auf Funktionen und Komponenten hin, die aufbauend auf meiner Arbeit entwickelt werden könnten.

2 Gegenstandsbereich

Um Stress am Arbeitsplatz erkennen zu können, muss zunächst klar sein, wie Stress auf den Menschen wirkt. Im Folgenden beschreibe ich kurz den Begriff *Stress* und gehe auf die Thematik der sensorbasierten Stresserkennung ein.

2.1 Stress

Seit Urzeiten ist der menschliche Körper darauf ausgelegt in Gefahrensituationen extrem schnell zu reagieren. Ein Beispiel aus der heutigen Zeit wäre ein Autofahrer auf der Autobahn, der sehr schnell reagieren muss und bremst, wenn ein Motorradfahrer seine Spur schneidet. Ein anderes Beispiel sind Eltern, die blitzschnell nach dem Kind greifen, das eine Sekunde später vom Wickeltisch gefallen wäre. Diese Beispiele zeigen, dass Stress im Alltag vorkommt und Teil des „normalen“ Tagesablaufes ist. Um in solchen Situationen schnell reagieren zu können,

muss der menschliche Körper eine hohe Leistung erbringen. Dazu schüttet er Noradrenalin aus. Zusätzlich schlägt das Herz schneller, Magen und Darm hören kurz auf zu arbeiten, die Muskeln sind angespannt und werden überdurchschnittlich gut mit Sauerstoff versorgt und das Gehirn arbeitet auf Hochtouren. [13, S.61f]

Nach der Stresssituation ist unser Körper entsprechend erschöpft und verlangt, dass wir uns ausruhen und erholen. Der Körper muss sich wieder in seinen sogenannten *Ruhewert* begeben. In dieser Regenerationsphase muss das Noradrenalin abgebaut werden und alle Organe und Muskeln sollten wieder in den Normalbetrieb übergehen können. Werden wir jedoch direkt mit dem nächsten Stressor konfrontiert, fährt sich das menschliche Stresssystem nicht wieder herunter, sondern bleibt auf dem hohen Niveau stehen. In diesem Fall haben wir über einen längeren Zeitraum eine hohe Konzentration an Stresshormonen im Blut. Dies führt zu einer längeren Hemmung der Immunabwehr und der Verdauung. Unser Körper kann sich an diesen Dauerstress gewöhnen und seinen *Ruhewert* nach oben setzen. Hierdurch wird der menschliche Körper unfähig, auf seinen eigentlichen, gesunden *Ruhewert* zurück zu kommen. Durch die dauerhafte Schwächung der Immunabwehr und der Verdauung können wir erschöpfen und körperlich sowie geistig erkranken. [13, S.65]

Stress am Arbeitsplatz kann durch viele verschiedene Einflüsse, wie zum Beispiel Informationsüberfluss, Zeit- und Termindruck oder ein zu hohes Arbeitstempo, ausgelöst werden [12, S.30]. Das persönliche Empfinden in einer Stresssituation spielt dabei eine wichtige Rolle. Denn Stress kann in zwei Formen, als positiver und als negativer Stress, auftreten. Negatives Stressempfinden kann unter anderem entstehen, wenn sich der Mensch bei der Bewältigung einer Aufgabe überfordert fühlt, wie es zum Beispiel durch einen *Cognitive Overload*[1] entstehen kann (siehe auch [3]). Diese negative Empfindung kann beim Menschen Demotivation auslösen. Erfährt ein Mensch diese Demotivation an seinem Arbeitsplatz, so wird er gegebenenfalls schlechtere Arbeit leisten. Zusätzlich wurde in den letzten Jahren beobachtet, dass der Mensch am Arbeitsplatz immer höheren psychischen Belastung ausgesetzt ist und die Arbeit immer intensiver wird [9, 10]. „Kann es also sein, dass Arbeit krank macht? Und wenn ja, was sind die konkreten Auslöser?“ [13] Zu dieser Fragestellung möchte ich eine Software entwickeln, die Forscher dabei unterstützt, neue Erkenntnisse auf dem Gebiet der computergestützten Stresserkennung und Stressprävention zu erlangen.

2.2 Stresserkennung

Empfindet eine Person Stress, so zeigt sie unter anderem bestimmte körperliche Reaktionen. Diese können beispielsweise schnellere Atmung, Schwitzen, erhöhte Muskelspannung und

[1] Frei übersetzt bedeutet *Cognitive Overload* soviel wie: *Überlastung der geistigen Fähigkeiten*

erhöhter Blutdruck sein [1, S.16]. Um Stress computergestützt erkennen zu können, müssen Sensoren eingesetzt werden, die diese körperlichen Reaktionen messen. Solche Sensoren werden Biosensoren genannt [11]. Eine Software muss dann die Messdaten auswerten und hieraus die aktuelle Höhe des Stresslevels ableiten.

Doch um den Grund für Stress am Arbeitsplatz ermitteln zu können, dürfen nicht nur die körperlichen Reaktionen betrachtet werden. Der aktuelle Arbeitskontext, also die aktiven Fenster, die E-Mails und Benachrichtigungen und die aktuell anstehenden Aufgaben des Arbeitenden, kann Auslöser für das Stressempfinden eines Menschen sein. Am Arbeitsplatz sollten also, neben den Biosensoren, auch Sensoren eingesetzt werden, die den Arbeitskontext überwachen.

Um die Daten aus diesen vielen verschiedenen Sensoren zusammenzutragen und global auswerten zu können, bedarf es einer zentralisierten Komponente, einem Framework.

3 Masterarbeit

Das Ziel meiner Masterarbeit ist die Entwicklung eines Software-Frameworks, das bei der Messung der körperlichen Reaktionen auf Stress unterstützt und ein Interface zur Anbindung von weiteren Sensoren bietet. In diesem Abschnitt beschreibe ich mein Vorhaben, die von mir geleisteten Vorarbeiten und den aktuellen Stand der Implementierung.

3.1 Ansatz

Die Stresserkennung mittels Bio- und Kontextsensoren ist ein enorm umfangreiches Aufgabengebiet. Die *Anzahl an unterschiedlichen Sensoren*, die *Theorie der körperlichen Stressreaktion* und die *Implementierung benötigter Algorithmen zur Auswertung der Messdaten* sind für sich eigene Forschungsbereiche. Zusätzlich fehlt es an einer Komponente, die die Messdaten aus allen möglichen Sensoren zusammenträgt und so eine globale Auswertung der Daten vornimmt. Die Lösung für dieses Problem ist ein Software-Framework, das so flexible Schnittstellen hat, dass aus jeder Programmiersprache Daten an die zentrale Steuerungseinheit des Frameworks übertragen werden können.

Ich habe mich mit diesem Problem beschäftigt und ein Software-Framework entwickelt, das Schnittstellen mit Hilfe von Topics und Nachrichten über einen ActiveMQ[2] anbietet. Hierdurch können vorhandene Sensor-Implementationen zur Stresserkennung durch wenige Erweiterungen in ihrer Implementation an das Framework angebunden werden. Die Algorithmen bleiben dabei in der jeweils bevorzugten Programmiersprache der Sensorimplementation und müssen

[2] http://activemq.apache.org/ (besucht am 30.01.2013)

nicht aufwendig portiert oder in einer weniger effektiven Programmiersprache implementiert werden.

3.2 Abgrenzung

Mein Software-Framework stellt die zentrale Steuerungseinheit inklusive der Schnittstellen dar. Die Entwicklung verschiedener Sensoren und deren Implementierung sind nicht Teil meiner Arbeit.

Der Aufbau meines Software-Frameworks wird gezielt modular gehalten. So können zentrale Komponenten des Systems leicht ausgetauscht werden. Am Ende meiner Masterarbeit wird kein ausgereifter Algorithmus zur Akkumulierung der Sensordaten zur Bestimmung eines globalen Stresslevels im Framework implementiert sein. Stattdessen bietet mein Framework leicht austauschbare Komponenten an, die um die entsprechenden Algorithmen erweitert werden können.

3.3 Vorarbeiten

In der Ringvorlesung *Anwendungen 1* habe ich begonnen einen Softwareagenten zu entwickeln, der einen am Computer Arbeitenden vor Stressfaktoren schützen soll. Der Softwareagent sollte automatisch Stress erkennen und darauf reagieren. Mein Ziel war es, dass der Softwareagent in einer Stresssituation gezielt Informationen zurückhält, um diese erst später, wenn sich der Arbeitende wieder in einer entspannteren Situation befindet, anzuzeigen. Während dieser Entwicklungsphase habe ich mich mit dem Thema Stress und der Betrachtung des aktuellen Aufgabenkontextes des Arbeitenden beschäftigt. [4]

Die Ringvorlesung *Anwendungen 2* gab mir die Gelegenheit thematisch nahe Forschungsarbeiten anderer Wissenschaftler zu betrachten. Dabei kam für mich das Résumé heraus: Stresserkennung ist ein sehr komplexes und umfangreiches Thema. [7]

Die ersten Implementierungsschritte folgten im *Projekt 1*. Hier habe ich die ersten Teile meines Frameworks genauer spezifiziert und diese implementiert. Es entstand unter anderem der Kern des Jira-Plug-ins. Um geeignete Sensoren zur Stresserkennung zu finden, wurden verschiedene Sensoren von mir begutachtet und getestet. Um neben den biologischen Stressmerkmalen auch den aktuellen Arbeits- und Aufgabenkontext zu erfassen, habe ich mir zusätzlich Gedanken über aufgabenbasiertes Arbeiten (siehe auch [2]) gemacht. [5] Am Ende von *Projekt 1* stellte sich erneut heraus, dass zur automatisierten Stresserkennung noch sehr viel Forschungsarbeit geleistet werden muss.

Die bis hierhin geleisteten Vorarbeiten haben mit gezeigt, dass das Ziel meiner Masterarbeit sehr hoch angesetzt war. Ich habe mein Ziel an dieser Stelle neu spezifiziert. Statt einen Softwareagenten zur automatischen Stresserkennung am Menschen zu entwickeln, konzentriere ich mich nun auf die Basis für solch einen Softwareagenten: Das Framework.

Im *Projekt 2* konnte ich dann die neue Ausrichtung meines Ziels weiter verfolgen. Die Implementation meines Software-Frameworks wurde von mir gezielt umstrukturiert und an die neuen Anforderungen angepasst. Die Schnittstellen habe ich erweitert und das Jira-Plug-in ausgelagert, so dass es als eigenständiges Programm laufen und über die AcitveMQ mit der zentralen Steuerungseinheit kommunizieren kann. Außerdem stellte ich weitere Überlegungen zur Spezifikation an. So entstand eine weitere Kommunikationsebene für die Plug-ins. Hierüber können die Plug-ins Menüpunkte in einem Kontextmenü eines System Tray Icons auf dem zentralen Rechner anlegen. Der Benutzer kann über diese Einträge dann zum Beispiel das Plug-in konfigurieren. Auf der ActiveMQ habe ich die relevanten Topics spezifiziert und die hierzu gehörenden Nachrichtenformate entwickelt. [6]

3.4 Konzept

Die Systemarchitektur meines Frameworks ist in Abbildung 1 dargestellt und kann in drei Hauptkomponenten unterteilt werden:

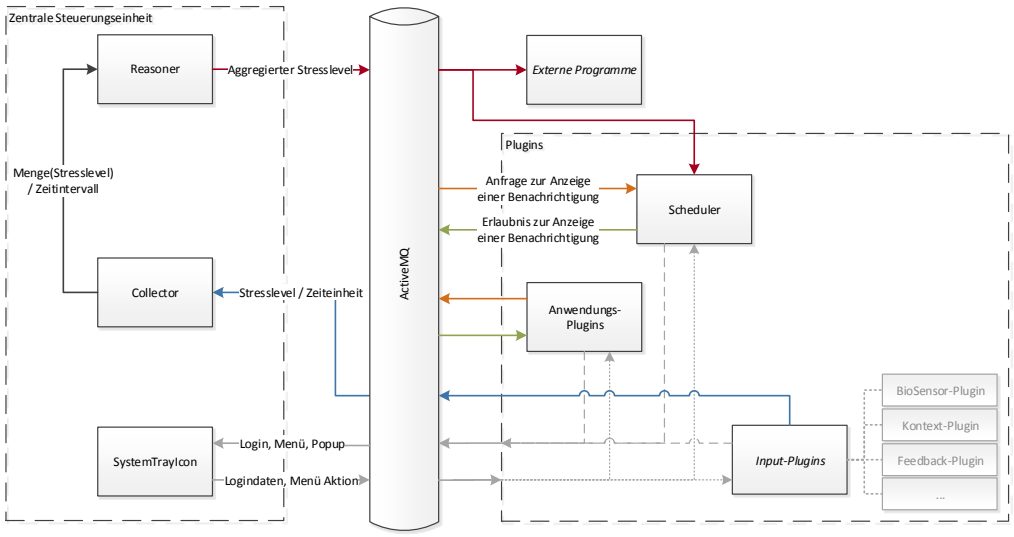

Abbildung 1: Übersicht über die Systemarchitektur meines Frameworks

Die zentrale Steuerungseinheit besteht aus den Komponenten *Collector* und *Reasoner*, die die Messdaten verarbeiten, sowie der Komponente für das System Tray Icon, das als Benutzerschnittstelle dient. Der *Collector* vereinigt die Messdaten der Plug-ins pro Zeiteinheit in einer Menge. Diese Menge gibt er weiter an den *Reasoner*. Der *Reasoner* ist als Listener des *Collectors* implementiert. Er verarbeitet die Menge an Messdaten und berechnet ein globales Stresslevel für die Zeiteinheit der vorhandenen Menge.

Das System Tray Icon bietet dem Benutzer unter anderem ein Kontextmenü an. Hierüber werden dem Benutzer Informationen angezeigt und Konfigurationsmöglichkeiten geboten. Jedes Plug-in kann eigene Menüpunkte in das Kontextmenü einfügen und dem Benutzer so eine Interaktionsschnittstelle zum Plug-in bieten. Außerdem können über das System Tray Icon Pop-up-Benachrichtigungen angezeigt und Logindaten vom Benutzer angefordert werden.

Die zentrale Steuerungseinheit ist der Teil des Frameworks, der auf dem Rechner des Anwenders ausgeführt werden muss.

Die ActiveMQ ist die Schnittstelle zwischen der zentralen Steuerungseinheit und den Plug-ins. Es werden verschiedene Topics auf der ActiveMQ angeboten, um die verschiedenen Kommunikationskanäle zu realisieren.

Die ActiveMQ kann auf einem beliebigen Rechner im Netzwerk gestartet werden. Wichtig ist nur, dass sowohl die zentrale Steuerungseinheit wie auch sämtliche Plug-ins eine aktive Netzwerkverbindung zur ActiveMQ haben.

Die Plug-ins repräsentieren verschiedene Komponenten, die mit der zentralen Steuerungseinheit über die ActiveMQ kommunizieren. Die Spezifikation meines Frameworks schreibt dabei nicht vor, was ein Plug-in messen muss, sondern nur, welche Daten ein Plug-in an die zentrale Steuerungseinheit übermitteln soll. Aktuell sind in der Systemarchitektur Input-Plug-ins, Anwendungs-Plug-ins und ein Scheduler eingeplant.

Input-Plug-ins messen den Stress am Menschen und überwachen seinen Arbeitskontext. Sie können in Kategorien, beispielsweise Biosensor-Plug-ins, Kontext-Plug-ins und Feedback-Plug-ins, eingeteilt werden.

Anwendungs-Plug-ins dienen der aktiven Stressprävention, in dem sie das Verhalten von Programmen, die als Stressor aggieren, gezielt manipulieren. Hierzu wird ein Add-on in ein vorhandenes Programm, beispielsweise Microsoft Outlook, eingebunden. Das Add-on soll mögliche Aktionen des Programms, die Stress beim Menschen auslösen könnten, erkennen und unterbinden oder zeitlich verzögern.

Der Scheduler regelt die Anzeige von Benachrichtigungen, da diese ebenfalls als Stressor wirken können. Er kann eine Hilfskomponente für Anwendungs-Plug-ins sein, die Benachrichtigungen von Programmen, wie zum Beispiel „Sie haben eine neue Nachricht“, verzögert anzeigen wollen.

Das Framework kann zusätzlich um Plug-ins mit anderen Einsatzmöglichkeiten erweitert werden. So sind zum Beispiel Plug-ins denkbar, die keine Messdaten ermitteln, sondern nur bestimmte Informationen im Kontextmenü des System Tray Icons anzeigen.

Jedes Plug-in kann zudem auf einem beliebigen Rechner im Netzwerk gestartet werden, solange eine aktive Netzwerkverbindung zur ActiveMQ besteht.

3.4.1 Schnittstellenbeschreibung

Auf der ActiveMQ werden verschiedene Topics angeboten, die als Schnittstellen zwischen der zentralen Steuerungseinheit und den Plug-ins dienen. Im Folgenden benenne ich die wichtigsten Topics meines Frameworks und beschreibe kurz deren Zweck.

StressCompanionConfigTopic Über die Topic zur Konfiguration des Frameworks synchronisieren sich die Instanzen der Konfigurationsklasse. Wird zur Laufzeit eine Änderung an einer Konfiguration, zum Beispiel direkt in der zentralen Steuerungseinheit, vorgenommen, so wird diese über die *StressCompanionConfigTopic* publiziert.

StressCompanionInputTopic Jedes Plugin, das Messdaten erstellt, die zur Berechnung des Stresslevels relevant sind, veröffentlicht seine Messdaten auf der *StressCompanionInputTopic*. Die zentrale Steuerungseinheit greift diese Daten ab und bildet pro Zeiteinheit eine Menge an verfügbaren Messdaten, um diese weiter zu verarbeiten.

StressCompanionResultTopic Aus der erstellten Menge an Messdaten wird ein globales Stresslevel berechnet. Dieses wird auf der *StressCompanionResultTopic* bereit gestellt. Externe Anwendungen haben so die Möglichkeit das vom Framework berechnete Stresslevel zu verwenden. Sie können so, unabhängig von den eigentlichen Sensordaten, auf jeden Stressor angemessen reagieren.

StressCompanionSystemTrayMenuTopic Zur Interaktion mit dem Benutzer wird auf dem Rechner des Benutzers ein System Tray Icon angezeigt. In das Kontextmenü des System Tray Icons können Plug-ins eigene Menüpunkte einbinden. Hierzu stellen die Plug-ins die Menüstruktur auf der *StressCompanionSystemTrayMenuTopic* bereit.

StressCompanionSystemTrayPopupTopic Das System Tray Icon wird ebenfalls zur Anzeige von Pop-up-Benachrichtigungen verwendet. Um eine Pop-up-Benachrichtigung anzuzeigen, muss ein Plug-in den Titel und den Nachrichtentext auf der *StressCompanionSystemTrayPopupTopic* bereitstellen. Die zentrale Steuerungseinheit greift diese Nachrichten dann ab, wandelt sie in das Pop-up um und zeigt es an.

StressCompanionLoginRequestTopic Manche Plug-ins, insbesondere die Kontext-Plug-ins, müssen sich an einem externen System anmelden, um benutzerbezogene Daten abgreifen zu können. Hierzu muss der Benutzer seine Logindaten eingeben können. Das Plugin kann die Logindaten über die *StressCompanionLoginRequestTopic* bei der zentralen Steuerungseinheit anfordern. Die zentrale Steuerungseinheit erfragt die Logindaten über ein Eingabefenster beim Benutzer. Die Logindaten werden dem Plugin dann von der zentralen Steuerungseinheit über die gleiche Topic zur Verfügung gestellt.

3.4.2 Implementation in Java

In den Projekten 1 und 2 habe ich eine Implementation meines Software-Frameworks in Java vorgenommen [5, 6]. Die Implementation ist in mehrere kleine Projekte strukturiert. Die einzelnen Projekte sind als Apache Maven[3] Projekte aufgesetzt. Hierdurch können Abhängigkeiten zwischen den Projekten sehr leicht aufgelöst werden. Die zentrale Steuerungseinheit ist in einem Projekt, jedes Plug-in in seinem eigenen Projekt organisiert. Zusätzlich gibt es ein Util-Projekt, in dem global verwendete Klassen implementiert werden.

Zentrale Steuerungseinheit Die Implementation der zentralen Steuerungseinheit umfasst unter anderem den *Collector* und den *Reasoner*. Diese erfüllen bisher lediglich ihren groben Zweck und sollen als austauschbare Platzhalter dienen. Wissenschaftler, die auf meiner Arbeit aufbauen, können so relativ einfach bessere Algorithmen für diese Basisklassen implementieren.

Der *Collector* fasst aktuell die Messdaten der Sensoren in einer Menge zusammen, sofern die Messdaten innerhalb eines bestimmten zeitlichen Intervalls aufgezeichnet wurden. Somit ergibt sich eine Menge an Messdaten pro zeitlichem Intervall. Gibt es für ein Intervall keine Messdaten mehr auf der ActiveMQ, wirft der *Collector* ein Event, sodass der *Reasoner* aktiv wird und die aktuelle Menge an Messdaten verarbeitet. Der *Reasoner* bildet nun einen gewichteten Mittelwert über die Messdaten. Die Gewichtung wird von den Sensoren pro Messdatum angegeben. Der berechnete Mittelwert ist dann das globale Stresslevel für dieses Intervall.

[3] http://maven.apache.org/ (besucht am 10.02.2013)

Die Verbindung zwischen dem *Collector* und dem *Reasoner* wurde von mir bewusst Event-basiert implementiert. Hierdurch können mehrere verschiedene Klassen an den *Collector* angebunden werden. Jede angebundene Klasse wird dann automatisch über neue Mengen von Messdaten benachrichtigt. Auf diese Weise kann das Framework parallel verschiedene Auswertungen der aktuellen Messdaten vornehmen.

Plug-ins Die Messdaten werden von den Plug-ins in einer Nachricht auf der *StressCompanionInputTopic* veröffentlicht. Hierbei handelt es sich nicht um die Rohdaten der Messung, sondern um bereits aufbereitete Daten. Jedes Plug-in muss aus seinen gemessenen Rohdaten ein Stresslevel berechnen und dieses mit einer Gewichtung versehen. Die Gewichtung soll eine Aussage über die Verlässlichkeit des Messwertes geben. Je geringer die Gewichtung, als desto unzuverlässiger muss der Messwert angesehen werden. Auf diese Weise können zum Beispiel Sensor-Plug-ins mögliche Messfehler kennzeichnen, damit diese Werte die Berechnung des globalen Stresslevels nicht negativ beeinflussen. Das ermittelte Stresslevel wird zusätzlich mit dem aktuellen Zeitstempel versehen, um den Zeitpunkt der Messung anzugeben.

Konfigurationsklasse Die Konfigurationsklasse stellt eine systemweite Repräsentation der aktuellen Konfiguration dar. Pro Komponente des Frameworks existiert eine Instanz der Konfigurationsklasse. Diese Instanzen synchronisieren sich automatisch über die ActiveMQ. Wird also in einer Komponente des Frameworks die Konfiguration angepasst, erfahren alle Komponenten von dieser Änderung und können sich entsprechend der neuen Konfiguration verhalten.

Über die Konfigurationsklasse wird ebenfalls der aktuelle Ausführungszustand des gesamten Frameworks abgebildet. Dieser beschreibt, ob aktuell Messungen durchgeführt oder ob die Komponenten des Frameworks heruntergefahren werden sollen.

Jira-Plug-in Neben der zentralen Steuerungseinheit, der Anbindung an die ActiveMQ und der Konfigurationsklasse habe ich ein Kontext-Plug-in in Java implementiert. Hierbei handelt es sich um eine Jira[4] Anbindung. Das Plug-in liest die Tickets des angemeldeten Benutzers aus und ermittelt hierüber ein Stresslevel. Dieses Plug-in dient als Beispiel für die Implementation eines Plug-ins und für einige mögliche Anbindungen an die zentrale Steuerungseinheit meines Frameworks.

[4] http://www.atlassian.com/software/jira/overview (besucht am 30.01.2013)

3.5 Chancen und Risiken

Neben den Vorteilen meines Frameworks birgt die Realisierung meiner Idee auch einige Risiken. Beides möchte ich folgend kurz darstellen.

3.5.1 Chancen

Mein Software-Framework stellt eine zentrale Komponente zur Verbindung verschiedener Sensoren zur Stressmessung am Menschen dar. Die dadurch gewonnene Möglichkeit, die Messdaten an einer zentralen Stelle aufzugreifen und auszuwerten, kann Wissenschaftler bei der Suche nach der Antwort auf die Frage „Was ist Stress am Arbeitsplatz und wie kann computergestützt darauf reagiert werden?“ unterstützen.

Der Aufbau meines Software-Frameworks ist bewusst modular gehalten. Hierdurch kann jede Komponente, teilweise sogar zur Laufzeit, ausgetauscht werden. Diese Flexibilität bietet Freiraum zur dynamischen Anpassung der eingesetzten Sensoren, um gegebenenfalls schnell auf Veränderungen der Anforderungen der Stressmessung reagieren zu können.

Außerdem bleibt die Implementierung einzelner Plug-ins durch die Anbindung über eine ActiveMQ programmiersprachenunabhängig. Der große Vorteil daran ist, dass zum Beispiel Sensorimplementationen mit aufwendigen Algorithmen in der jeweils optimalen Programmiersprache implementiert werden können.

3.5.2 Risiken

Die Verwendung einer ActiveMQ birgt jedoch auch ein großes Risiko. Es muss gewährleistet sein, dass eine sichere, stabile und ausreichend schnelle Netzwerkverbindung zwischen der zentralen Steuerungseinheit, der ActiveMQ und sämtlichen Plug-ins besteht. Sollte diese Netzwerkverbindung einmal ausfallen, so kann keine Komponente mehr auf die Nachrichten, die auf der ActiveMQ liegen, zugreifen.

Die durch die Netzwerktechnik bedingte Verzögerung bei der Nachrichtenübertragung führt zusätzlich zu einer eingeschränkten Echtzeitfähigkeit des Systems. Hierzu muss im Einzelfall abgewogen werden, ob die Netzwerkverzögerung ein Problem für den Anwendungsbereich darstellt.

Weitere Risiken ergeben sich aus der von mir gewählten Schnittstellenarchitektur. Die von mir definierten Topics und Nachrichten können nur einen Teil der tatsächlich benötigten Menge an Topics und Nachrichten abdecken. Hier muss im Einzelfall das System entsprechend erweitert werden.

4 Ausblick

Ich habe ein Software-Framework entwickelt, das bei der Messung der körperlichen Reaktionen auf Stress unterstützt und ein Interface zur Anbindung von weiteren Sensoren bietet. Die aktuelle Implementation kann Sensoren und andere Plug-ins anbinden und die gemeldeten Daten verarbeiten. Außerdem ist es möglich, die Plug-ins über ein Kontextmenü auf einem System Tray Icon zu konfigurieren.

Im Rahmen einer Masterarbeit können jedoch nicht alle Anforderungen, die an ein solches Framework gestellt werden, umgesetzt werden. Um Stress am Arbeitsplatz automatisiert zu erkennen und aktiv darauf reagieren zu können, müsste mein Framework um weitere Funktionen und Komponenten erweitert werden. Wissenschaftler könnten auf meiner Arbeit aufbauen, um dieses Ziel zu erreichen. Folgend gebe ich einen Überblick über einige solcher Funktionen und Komponenten.

Reasoner Im Reasoner kommen Messdaten von verschiedenen Sensoren zusammen, die gegebenenfalls unterschiedliche Stressmerkmale gemessen haben. Zur verlässlichen Stresserkennung sollte der Reasoner um robuste Algorithmen erweitert werden, die die Menge an Stressleveln pro Zeiteinheit auswerten. Diese Algorithmen müssen sicherstellen, dass nicht nur die Menge an Messdaten sinnvoll ausgewertet wird, sondern auch kein einzelner Sensor durch einen Messfehler das gesamte Ergebnis verfälscht.

Scheduler Um auf Basis der ermittelten Stresslevel automatisiert reagieren und den Menschen bei seiner Stressprävention unterstützen zu können, wird ein Plug-in benötigt, das unter anderem die Anzeige sämtlicher Pop-up-Nachrichten auf dem Desktop des Anwenders regelt. Hierzu sollte der Scheduler als Plug-in realisiert werden. Der Scheduler muss dann die aktuellen Stressleveldaten von der ActiveMQ empfangen und das Benachrichtigungsverhalten der laufenden Anwendungen mit Hilfe von Add-ons regeln. Die Add-ons könnten ebenfalls über die ActiveMQ mit dem Scheduler kommunizieren.

Feedback-Plug-in Um Messungen durchzuführen, die Daten darüber liefern, wann ein Mensch gestresst ist und welche körperlichen Reaktionen der Mensch in dieser Situation zeigt, könnte ein weiteres konkretes Plug-in, ein Buzzer, implementiert werden. Der Anwender soll den Buzzer drücken, sobald er sich gestresst fühlt. Das Plug-in erstellt dann periodisch Abbilder der aktuellen Messdaten der Sensoren und speichert diese ab. Drückt der Anwender den Buzzer erneut, wird der aktivierte Stresszustand wieder aufgehoben. Das Plug-in sollte hier ein letztes

Abbild erstellen. Durch die Abbilder der Messdaten können später Aussagen zum Zustand des Menschen in der Stresssituation gemacht werden.

Literatur

[1] Prof. Dr. Gert Kaluza. *Stressbewältigung: Trainingsmanual zur psychologischen Gesundheitsförderung*. Springer, 2011. ISBN: 97836421371198.

[2] Mik Kersten. »Focusing knowledge work with task context«. AAINR26735. Diss. Vancouver, BC, Canada, Canada: University of British Columbia, 2007. ISBN: 978-0-494-26735-6.

[3] David Kirsh. »A Few Thoughts on Cognitive Overload«. In: *Intellectica* 30 (2000), S. 19–51.

[4] Benjamin Kuska. *CoSEC - A Stress Companion - Ein Software-Agent zur Unterstützung von hochkonzentrierter Arbeit*. Techn. Ber. Hochschule für angewandte Wissenschaften Hamburg, 2012.

[5] Benjamin Lindemann. *Stress am IT-Arbeitsplatz - Projektbericht*. Techn. Ber. Hochschule für angewandte Wissenschaften Hamburg, 2012.

[6] Benjamin Lindemann. *Stress am IT-Arbeitsplatz - Projektbericht 2*. Techn. Ber. Hochschule für angewandte Wissenschaften Hamburg, 2013.

[7] Benjamin Lindemann. *Stress am IT-Arbeitsplatz - Related Work*. Techn. Ber. Hochschule für angewandte Wissenschaften Hamburg, 2012.

[8] Andrea Lohmann-Haislah. *Stressreport Deutschland 2012*. Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, 2012.

[9] Prof. Dr. Christina Maslach und Prof. Dr. Michael P. Leiter. *Die Wahrheit über Burnout*. Springer-Verlag Wien, 2001.

[10] Dipl.-Päd. Bärbel Meschkutat und Dipl.-Psych. Dr. Beate Beermann, Hrsg. *Psychosoziale Faktoren am Arbeitsplatz unter Berücksichtigung von Streß und Belästigung*. Bundesanstalt für Arbeitsschutz, 1995.

[11] Saraju Mohanty. *Biosensors: A Survey Report*. 2001.

[12] Ludger Rensing u. a. *Mensch im Stress - Psyche, Körper, Moleküle*. 1. Aufl. München: Spektrum-Akademischer Vlg, 2005.

[13] Hans-Peter Unger und Carola Kleinschmidt. *Bevor der Job krank macht*. Kösel-Verlag, 2011.