Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterprojekt 1

Benjamin Lindemann

Stress am IT-Arbeitsplatz - *Projektbericht*

18. August 2012

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Kurzzusammenfassung

Das Ziel meiner Masterarbeit ist es, ein System zur Stresserkennung am IT-Arbeitsplatz zu entwickeln. Beliebige Sensoren ermitteln einen Stresslevel pro Zeiteinheit. Dieser wird eingesetzt, um in Stresssituationen gezielt weiteren Stress zu vermeiden.

Im Masterprojekt 1 konnte ich mit der Evaluation von geeigneten Sensoren zur Stressmessung am IT-Arbeitsplatz beginnen. Hier konnte ich bereits erste Sensoren ausschließen. Außerdem habe ich eine erste Spezifikation angestellt, die das Framework, das ich in meiner Masterarbeit entwickeln möchte, näher beschreibt. Teile dieser Spezifikation konnten bereits erfolgreich implementiert werden.

Dieser Bericht fasst die Ergebnisse des Masterprojekts 1 zusammen und gibt einen Ausblick auf die weiterführende Arbeit.

1 Einleitung

1.1 Motivation

Die immer größer werdende Informationsflut am IT-Arbeitsplatz führt unweigerlich zu einem *Cognitive Overload* [4]. Der Arbeitende läuft Gefahr durch Popup-Benachrichtigungen immer wieder aus seinem aktuellen Arbeitskontext gerissen zu werden. Diese ständigen Unterbrechungen können zu einem erhöhten Zeit- und Konzentrationsaufwand führen und den Arbeitenden stressen [3]. Ist der Arbeitende über längere Zeit diesem Stress ausgesetzt, besteht die Gefahr, dass er sich zunehmend schlechter fühlt, schlechtere Arbeit leistet und ggf. sogar als Arbeitskraft ausfällt.

Um diesem Problem entgegenzuwirken, möchte ich ein System zur Stressprävention entwickeln, das den Arbeitenden bei der Konzentration auf seine Arbeit unterstützt. Mein Ziel ist es, eine Software zu entwickeln, die den Stress beim Arbeitenden frühzeitig automatisiert erkennt und aktiv darauf reagiert.

1.2 Vision

Mein Ziel ist es, eine Software zu entwickeln, die präventiv gegen Stress eingesetzt wird und so den Arbeitenden aktiv in seiner Arbeit unterstützt. Hierzu könnte die Software mit Hilfe verschiedener Sensoren Stress beim Arbeitenden erkennen, diesen einstufen und je nach Stufe die Informationsflut am IT-Arbeitsplatz zurückhalten. Die Informationen werden dem Arbeitenden dabei nicht vorenthalten, sondern lediglich zeitlich verzögert mitgeteilt. So erhält der Arbeitende eine kurze und effektive Konzentrationsphase für die Bearbeitung einer konkreten Aufgabe. Die Erkennung des Stresses und die Kontrolle der Informationsflut soll dabei für den Anwender möglichst unbemerkt ablaufen.

In meiner Masterarbeit möchte ich für diese Problemstellung ein Framework schaffen, das unabhängig von der Implementierung der einzelnen Sensoren ist. Hierdurch können später auf einfache Weise neue Sensoren in das bestehende System integriert werden. Um dies zu ermöglichen, spezifiziere und implementiere ich eine gemeinsame Kommunikationsschnittstelle zwischen den Sensorimplementationen und dem Framework. Die Algorithmen zur Bestimmung des Stresslevels sollen in die jeweilige Implementation der Sensoren verlagert werden. Nur so kann ich garantieren, dass die spezifischen Messdaten der unterschiedlichsten Sensoren korrekt ausgewertet und verarbeitet werden. Das Framework selbst soll die Steuerung der Informationsflut übernehmen.

2 Auswahl von Sensoren

Das Framework, das ich in meiner Masterarbeit entwickeln möchte, benötigt Inputdaten von verschiedenen Sensoren. Diese Sensoren sollen Stress, bzw. Stressfaktoren beim Arbeitenden messen. Um geeignete Sensoren zu finden, habe ich mir für das Masterprojekt 1 die Aufgabe gestellt, verschiedene Sensoren auszuprobieren. Hierdurch sollen Sensoren gefunden werden, die besonders gut für die Stresserkennung am IT-Arbeitsplatz geeignet sind.

In diesem Abschnitt beschreibe ich, welche Sensoren ich bereits kennen gelernt habe und ausprobieren konnte. Zunächst stelle ich verschiedene kamerabasierte Sensoren vor. Darauf folgend beschreibe ich Biosensoren, die ich kennen gelernt habe. Eine spezielle Auswahl einzelner Sensoren für meine Masterarbeit konnte ich im Masterprojekt 1 noch nicht treffen. Dies muss im nächsten Projektschritt erfolgen.

2.1 Kamerabasierte Sensoren

Kamerabasierte Sensoren erheben ihre Messdaten durch Aufzeichnung eines Videos und Verarbeitung des Videobildes per Software. Häufig werden hierfür einfache Webcams genutzt. Eine spezialisierte Software verarbeitet anschließend das Videobild. Es gibt aber auch spezialisierte Hardware. Dazu gehören zum Beispiel Eyetracker, zur Erkennung der Augenbewegung und des Pupillendurchmessers, und hochauflösende Hochgeschwindigkeitskameras, die mehr Bildinformationen liefern als beispielsweise Webcams.

Folgend möchte ich die von mir evaluierten, kamerabasierten Sensoren vorstellen und ihre Einsatzmöglichkeit in meiner Masterarbeit aufzeigen.

Emotions-Erkenner Die Mimik einer Person kann, mit Hilfe einer Kamera und einer entsprechenden Software, aufgezeichnet und erkannt werden [6]. Aus der Mimik kann die Software dann Emotionen ableiten. Verschiedene Emotionen wie Ärger, Wut oder Verzweiflung, können ein Anzeichen für Stress sein [2].

Es hat sich gezeigt, dass kamerabasierte Emotions-Erkenner häufig sehr viel Rechenkraft benötigen, um das Videobild in Echtzeit zu analysieren [6]. Zudem sind sie in der Stresserkennung fehleranfällig, da die Mimik von der aufgezeichneten Person aktiv beeinflusst werden kann. Aus diesen Gründen werde ich keine Emotions-Erkenner in meiner Masterarbeit einsetzen.

Eyetracker Ein anderes Indiz für Stress kann die schnelle Augenbewegung oder der Pupillendurchmesser [1, 8] sein. Spezialisierte kamerabasierte Sensoren für die Erkennung der Augenbewegung und die Messung des Pupillendurchmessers sind Eyetracker.

Im Masterprojekt 1 stand mir ein Tobii X120 Eyetracker[1] zur Verfügung. Der X120 kann die Augenbewegung aufzeichnen. Er ist ein stationärer Eyetracker, was bedeutet, dass er fest an einem bestimmten Ort stehen muss. Ich habe im Masterprojekt 1 erste Versuche mit dem X120 durchgeführt. Der Eyetracker lieferte nur dann gute Ergebnisse, wenn er sehr präzise kalibriert wurde und der Anwender sich mit seinem Kopf in einem geringen Bereich vor dem Eyetracker aufhielt. Andernfalls wichen die Messergebnisse stark ab. Die Kalibrierung musste nach einer Veränderung der Sitzposition erneut durchgeführt werden. Ein komfortabler Einsatz im IT-Arbeitsalltag ist mit diesem stationären Eyetracker meiner Meinung nach nicht möglich. Daher werde ich den Tobii X120 Eyetracker nicht in meiner Masterarbeit einsetzen.

Die Nachteile eines stationären Eyetrackers könnten jedoch durch den Einsatz eines mobilen Eyetrackers aufgehoben werden. Ein mobiler Eyetracker wird häufig wie eine Brille getragen, wodurch er unabhängig von der Sitzposition und der Kopfbewegung ist. Ein mobiler Eyetracker könnte also durchaus als Sensor für meine Masterarbeit in Frage kommen, wenn seine Messdaten in Echtzeit abgefragt werden können. Bisher hatte ich jedoch nicht die Möglichkeit einen mobilen Eyetracker zu testen.

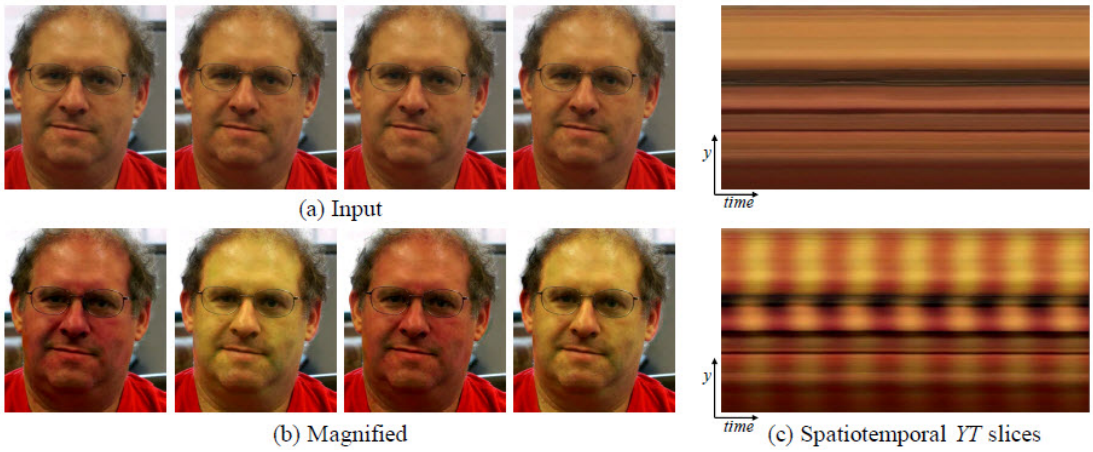

Abbildung 1: Eulersche Videoverstärkung aus [7]

[1] http://www.tobii.com/en/eye-tracking-research/global/products/hardware/tobii-x60x120-eye-tracker/ - zuletzt besucht am 08.08.2012

Puls-Messer Der Puls des Menschen ist ein wichtiger Indikator für Stress. Forscher am *MIT Computer Science and Artificial Intelligence Laboratory* haben eine Möglichkeit gefunden, mit Hilfe von Bildverarbeitung den Puls einer Person im Kamerabild sichtbar zu machen (siehe Abbildung 1) [7].

Diese Art der Pulsmessung ist für meine Masterarbeit sehr interessant. Die Evaluierung dieser Methode erfolgt im Masterprojekt 2.

2.2 Biosensoren

Der menschliche Körper stellt verschiedene Zustandsgrößen über die aktuell im Körper ablaufenden biologischen Vorgänge zur Verfügung. Diese Zustandsgrößen können mit Biosensoren gemessen werden. Bei den nichtinvasiven Biosensoren bringt man Sensoren zum Beispiel auf der Haut des Anwenders an. Durch diese Sensoren können unter anderem der Hautleitwert, der Puls oder die Atmung gemessen werden. Folgend möchte ich kurz drei Forschungseinrichtungen und deren Arbeit mit Biosensoren vorstellen.

Medizin Labor der FH Brandenburg Das Biosignalverarbeitungslabor der FH Brandenburg[2] beschäftigt sich mit dem Einsatz von Biosensoren wie beispielsweise EKG, EEG, EMG, GSR, Sauerstoffsättigung und Blutvolumenpuls. Die Erfahrungen der Mitarbeiter des Labors könnten hilfreich bei der Auswahl der Biosensoren für meine Masterarbeit sein.

Im Masterprojekt 1 konnte ich bereits eine erste Verbindung zum Labor aufbauen. Weitere Gespräche sind bereits geplant.

BIOPAC Die HAW Hamburg besitzt in ihrem Labor am Bergedorfer Campus ein *BIOPAC*[3]. An diesem ist es möglich, verschiedene Sensoren anzuschließen und die Messdaten der Personen am PC auszulesen.

Als Proband in einer Versuchsreihe konnte ich das Gerät bereits kennen lernen. Hierbei bekam ich einen ersten Überblick über den enormen Funktionsumfang des Pakets. Es bedarf jedoch weiterer Tests mit dem Gerät selbst, um festzustellen, ob das Gerät für die Stressmessung am IT-Arbeitsplatz geeignet ist.

2.3 Sonstige Sensoren

Torsten Rauschan hat in seiner Bachelorarbeit zwei Sensoren vorgestellt, die Alternativen bzw. Erweiterungen zu den genannten Sensoren bieten [5]. Zunächst hat er sich mit dem

[2] http://www.fh-brandenburg.de/informatik/biolab.html - zuletzt besucht am 06.08.2012
[3] http://www.biopac.com/ - zuletzt besucht am 06.08.2012

Neural Impulse Actuator der Firma OCZ Technology beschäftigt. Das Gerät misst Impulse der Gesichtsmuskeln und der Gehirnströme, um diese in Maus- oder Tastaturbefehle umzusetzen. Das Gerät konnte ich in Selbsttests im Masterprojekt 1 ausprobieren. Dabei stellte sich heraus, dass der Anwender zunächst sehr lange trainieren muss, um das Gerät überhaupt verwenden zu können. Zusätzlich existierte zum Zeitpunkt dieser Ausarbeitung kein programmierbares Interface, um die Daten des Geräts auszulesen. Aus diesen beiden Gründen werde ich den *Neural Impulse Actuator* nicht in meiner Masterarbeit einsetzen.

Das zweite Gerät, das Torsten Rauschan getestet hat, ist das Wii Balance Board. Mit diesem Board können die Bewegungen der Füße, die auf dem Board stehen, gemessen werden. Ist der Anwender unruhig, wippt er mit seinem Fuß auf und ab. Dieser Impuls könnte dann als Stressreaktion betrachtet werden. Im Masterprojekt 1 konnte ich noch kein Wii Balance Board testen.

2.4 Zusammenfassung

Die kamerabasierten Sensoren sind, bis auf den Puls-Messer und ggf. den mobilen Eyetracker, nicht für den Einsatz in meiner Masterarbeit geeignet. Eine Evaluierung des Puls-Messers und des mobilen Eyetrackers stehen aus.

Der Bereich der Biosensoren wurde von mir im Masterprojekt 1 noch nicht hinreichend evaluiert. Hier Bedarf es weiterhin der Kommunikation mit den wissenschaftlichen Einrichtungen und weiteren Tests von zur Verfügung stehenden Sensoren.

Weitere, alternative Sensoren zur Stressmessung, wie sie Torsten Rauschan eingesetzt hat, könnten ebenfalls noch evaluiert werden. Aus zeitlichen Aspekten sehe ich jedoch in der aktuellen Situation von der zusätzlichen Evaluierung dieser Alternativen ab.

3 Framework

In meiner Masterarbeit möchte ich ein Software-Framework entwickeln, das Sensorinformationen verarbeiten und das Benachrichtigungsverhalten von Anwendungen steuern kann. Hierzu müssen die grundsätzliche Struktur sowie die einzelnen Komponenten geplant werden. Im Projekt 1 konnte ich bereits erste Teile des Frameworks genauer spezifizieren und diese Teile implementieren.

3.1 Planung

Die Struktur des Frameworks soll so aufgebaut werden, dass jede Komponente, die von extern Daten an das Framework anbindet, als Plugin implementiert wird. Bei den Plugins unterscheide

ich zwischen *Input-Plugins*, die Sensoren anbinden, und *Anwendungs-Plugins*, die sich in eine bestehende Anwendung, zum Beispiel einen E-Mail Client, integrieren. Die Kernkomponenten des Frameworks werden unter der Bezeichnung *StressCompanion Core* zusammengefasst.

Der generelle Aufbau des Frameworks ist in der Abbildung 2 dargestellt.

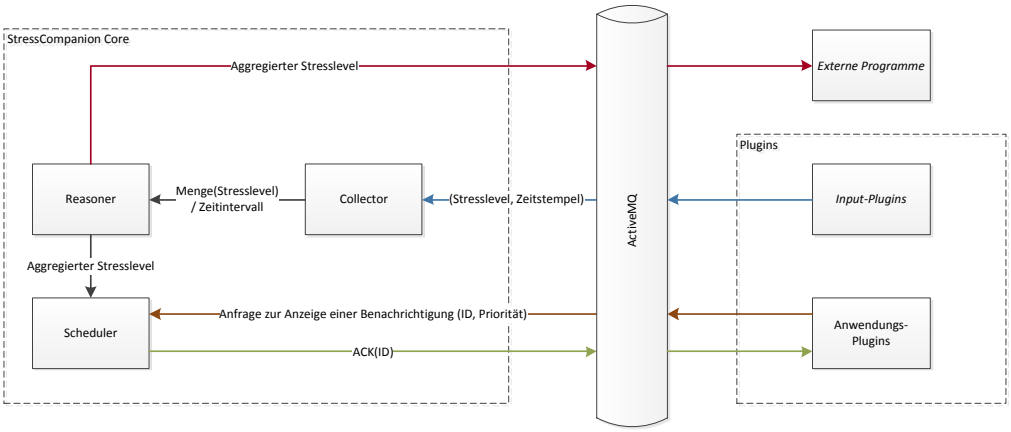

Abbildung 2: Komponenten des Frameworks

Ein *Input-Plugin* verarbeitet die rohen Messdaten des implementierten Sensors und wertet diese Messdaten aus. Es berechnet ein Stresslevel pro Zeiteinheit und übermittelt diesen an den *StressCompanion Core*. Die Sensorinformationen sollen so vereinfacht werden, dass beliebige Sensoren an den *StressCompanion Core* angebunden werden können.

Ein *Anwendungs-Plugin* soll die Funktionalität einer bestehenden Anwendung, zum Beispiel eines E-Mail Clients, erweitern. Dabei bietet das *Anwendungs-Plugin* Funktionen zur Stressprävention. Bei einem E-Mail Client könnte sich ein *Anwendungs-Plugin* beispielsweise vor die Komponente zur Anzeige von Benachrichtigungen setzen. Hier würde es dann die Anzeige der Benachrichtigungen steuern, in dem es für jede Benachrichtigung eine Priorität vergibt und beim *StressCompanion Core* um die Erlaubnis zur Anzeige bittet. So könnten die Benachrichtigungen des E-Mail Clients in Abhängigkeit des aktuellen Stresslevels gesteuert werden.

Um eine reibungslose Kommunikation zwischen den Plugins und dem *StressCompanion Core* zu ermöglichen, müssen Schnittstellendefinitionen ausgearbeitet werden. Um eine einheitliche und Implementierungssprachen unabhängige Kommunikation zu ermöglichen, setze ich als Kommunikationsmedium zwischen den Plugins und dem *StressCompanion Core* auf eine

ActiveMQ[4]. Die Definition der Schnittstelle beschränkt sich somit auf einen Nachrichtenkanal und ein Nachrichtenformat.

3.2 Aufbau des Frameworks

Im Folgenden beschreibe ich meine aktuellen Definitionen der Komponenten und der Schnittstellen zwischen den Komponenten meines geplanten Frameworks. Die Bezeichnung *Software* oder auch *bestehende Software* beschreibt immer ein Computerprogramm, das nur durch ein Anwendungs-Plugin Bezug zu meiner Masterarbeit bekommt.

3.2.1 Beschreibung der Komponenten

StressCompanion Core Im Kern der Anwendung sollen die von den *Input-Plugins* gemeldeten Stresslevel verarbeitet werden. Der *Collector* ruft die gemeldeten Stresslevel von der ActiveMQ ab und ordnet sie anhand der von den *Input-Plugins* übermittelten Zeitstempeln in Zeitintervalle ein. Auf diese Weise können Stresslevel von verschiedenen Sensoren, die in einem kurzen Zeitraum gemessen wurden, in einer Menge zusammengefasst werden.

Die Menge von Stressleveln pro Zeiteinheit erhält der *Reasoner*, der die eigentliche Logik des Frameworks beinhaltet. Er verarbeitet die Menge der Stresslevel und ggf. weiteren Informationen zu einem aggregierten Stresslevel. Das aggregierte Stresslevel wird dann an den *Scheduler* weitergegeben und auf der ActiveMQ veröffentlicht. Die Veröffentlichung ermöglicht es externen Programmen, die keine weitere Kommunikation mit dem Framework benötigen, das aktuelle Stresslevel weiter verarbeiten zu können.

Der *Scheduler* benötigt das aggregierte Stresslevel, um Anfragen von *Anwendungs-Plugins* zu verarbeiten. Er ruft die von den *Anwendungs-Plugins* gemeldeten Prioritäten der Nachrichten von der ActiveMQ ab. Dann wird er die Nachrichten freigeben, die von der Priorität unter dem aktuellen Stresslevel angezeigt werden dürfen.[5] Die Freigabe wird ebenfalls über die ActiveMQ an die *Anwendungs-Plugins* zurückgemeldet.

Plugins Ein *Input-Plugin* implementiert einen Sensor zur Stresserkennung. Die Programmiersprache, in der das Plugin den Sensor anspricht, ist hierbei irrelevant und in der Regel abhängig von dem jeweiligen SDK[6] des Sensors. Jedes *Input-Plugin* muss selbstständig die

[4] ActiveMQ ist ein Kommunikationsmedium mit einem einheitlichen Nachrichtenformat. Für weitere Informationen siehe `http://activemq.apache.org/` - zuletzt besucht am 08.08.2012

[5] Eine genaue Definition, ab welchem Stresslevel welche Prioritäten aktiviert sind, erfolgt im nächsten Projektschritt.

[6] Software Development Kit

gemessenen Werte verarbeiten und diese in einem Stresslevel pro Zeiteinheit zusammenfassen. Das Stresslevel pro Zeiteinheit muss dann unverzüglich auf der ActiveMQ unter einem definierten Nachrichtenkanal in einem definierten Nachrichtenformat veröffentlicht werden.

Die Auslagerung der Verarbeitung der rohen Sensorinformationen in das *Input-Plugin* ermöglicht mir, das Framework unabhängig von einem spezifischen Sensor aufzubauen. Der Kern des Frameworks, also alles, was im *StressCompanion Core* zusammengefasst ist, soll somit nur die Verarbeitung von Stressleveln und die Freigabe von Benachrichtigungen übernehmen.

Ein *Anwendungs-Plugin* stellt eine Erweiterung einer bestehenden Software dar. Es soll die Funktionalität der Anzeige von Benachrichtigungen in der bestehenden Software steuern, um so zusätzlichen Stress zu vermeiden. Das *Anwendungs-Plugin* schaltet sich dabei vor die Stelle in der bestehenden Software, die die Benachrichtigung anzeigt, und hält die Benachrichtigung so lange zurück, bis das *Anwendungs-Plugin* die Erlaubnis zur Anzeige der Benachrichtigung vom *Scheduler* bekommen hat. Dann gibt das *Anwendungs-Plugin* die Benachrichtigung frei und überlässt die Anzeige der Benachrichtigung der bestehenden Software.

3.2.2 Beschreibung der Schnittstellen

Die Nachrichten, die über den ActiveMQ ausgetauscht werden, sollen im JSON-Format formuliert sein. Jede Nachricht beinhaltet grundlegende Informationen, eine eindeutige ID und die Versionsnummer der Anwendung, die intern im LivingPlace Hamburg[7] abgestimmt wurden. Die erweiterten Einträge in der jeweiligen Nachricht hängen von der jeweiligen Situation ab.

Schnittstelle *Input-Plugin - StressCompanion Core* In der Nachricht vom *Input-Plugin* an den *StressCompanion Core* sollen das vom *Input-Plugin* ermittelte Stresslevel und der Zeitstempel, an dem das Stresslevel erkannt wurde, enthalten sein. Das Stresslevel macht eine Aussage darüber, wie stark der vom *Input-Plugin* erkannte Stress ist. Als Einheit soll eine einfache Skala von 1 bis 10 dienen, wobei 1 für das geringste und 10 für das höchste Stresslevel steht.

Schnittstelle *Anwendungs-Plugin - StressCompanion Core* Die Anfragen zur Anzeige von Benachrichtigungen enthalten eine Benachrichtigungs-ID, die pro bestehender Software eindeutig die Benachrichtigung identifiziert, und eine Priorität, die die Dringlichkeit der Benachrichtigung ausdrückt. Als Antwort erhält das *Anwendungs-Plugin*, ebenfalls über die

[7] http://www.livingplace.org - zuletzt besucht am 08.08.2012

ActiveMQ, eine ACK[8]-Nachricht inkl. der zugehörigen ID. Mit Hilfe der ID können somit pro bestehender Software mehrere Benachrichtigungen in der Warteschlange stehen.

3.2.3 User Interface

Zur Interaktion zwischen Anwender und Framework wird, nach dem Start des Frameworks, ein SystemTray Icon angezeigt. Hierüber kann ein Menü aufgerufen werden, das verschiedene Optionen anzeigt. Neben möglichen Optionen des *StressCompanion Core* zur Konfiguration der grundlegenden Einstellungen des Frameworks kann jedes Plugin einen eigenen Eintrag im Menü erhalten, der wiederum eine unbegrenzte Anzahl an Unterpunkten enthalten darf. Dieses Plugin-Menü kann verwendet werden, um Konfigurations-Optionen pro Plugin für den Anwender anzubieten. Um das Plugin-Menü zu erstellen, sendet das Plugin eine *updateMenu*-Nachricht über die ActiveMQ. Der *StressCompanion Core* verarbeitet die Informationen der Nachricht und setzt diese in einen Unterpunkt des Menüs um. Die Aktionen der einzelnen Einträge im Plugin-Menü sind als JSON-Nachricht zu formulieren. Die JSON-Nachricht wird bei Aktivierung des Menüpunktes an das entsprechende Plugin gesendet. Der prinzipielle Ablauf ist in Abbildung 3 dargestellt.

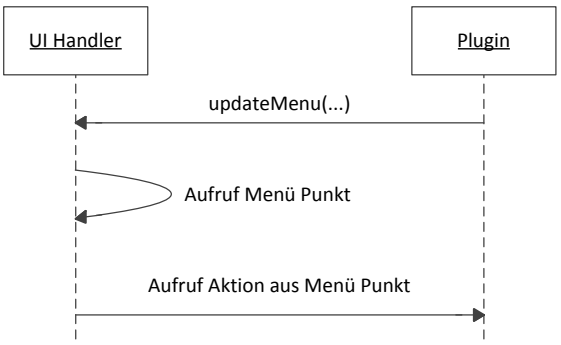

Abbildung 3: Ablauf der Einbindung eines Plugin Menüs

[8] Abkürzung für acknowledgement, engl. für Bestätigung

3.3 Aktueller Stand der Implementierung

Im Masterprojekt 1 konnte ich bereits erste Teile meines geplanten Frameworks umsetzen. Für die Anbindung an die ActiveMQ habe ich eine Utility-Klasse implementiert.

Als erstes *Input-Plugin* wurde von mir eine Anbindung an das Ticketsystem eines Jira-Repositories geschaffen. Dieses *Input-Plugin* liest die Tickets des angemeldeten Anwenders aus und ermittelt das Stresslevel anhand der Anzahl der offenen Tickets. Die Implementation soll als Beispiel für ein *Input-Plugin* dienen.

Auch das erste *Anwendungs-Plugin* ist eine Anbindung an das Ticketsystem eines Jira-Repositories. Es liest ebenfalls die Tickets des angemeldeten Anwenders aus. Die Tickets werden dann in einer Liste in einem Unterpunkt des Plugin-Menüs angezeigt. Jedes neue Ticket wird als Popup-Benachrichtigung angezeigt. Die Implementation der Nachfrage zur Anzeige der Benachrichtigung am *Scheduler* steht noch aus. Dieses Plugin ist in keine bestehende Software integriert, sondern soll als Beispiel für die Integration in das Plugin-Menü und die Verzögerung von Popup-Benachrichtigungen dienen.

Neben den Kernkomponenten und den zwei Plugins wurden im Masterprojekt 1 erste Interaktionsmöglichkeiten mit dem Framework implementiert. Ein SystemTray-Icon dient als Interaktionspunkt. Ein Menü zeigt Optionen, mit denen das Framework konfiguriert werden kann. Jedes Plugin hat die Möglichkeit, einen Menüpunkt zu erstellen und diesen zur Laufzeit zu aktualisieren. Dies konnte am Beispiel des Jira-Anwendungs-Plugins bereits implementiert werden.

4 Ausblick

Im Masterprojekt 1 konnte ich erste Sensoren ausprobieren. Einige Sensoren, die ich in Erfahrung gebracht habe, blieben jedoch bisher ungetestet. Im Masterprojekt 2 werde ich also noch einmal Zeit in die Evaluierung von geeigneten Sensoren stecken. Hierzu sind Gespräche mit dem Medizin Labor der FH Brandenburg und der medizinischen Fakultät der Universität Ulm geplant. Außerdem möchte ich mich mit dem *BIOPAC* aus dem Labor der HAW Hamburg Campus Bergedorf beschäftigen.

Wurden die Sensoren ausgewählt, muss pro Sensor ein Plugin geschrieben werden, das Stress beim Anwender erkennt und einstuft. Dieses Plugin muss das Stresslevel über die spezifizierte Schnittstelle an das Framework senden. Um meine Annahmen evaluieren zu können, müssen mindestens zwei verschiedene Sensoren als Plugin implementiert werden. Nur so kann die *Reasoner*-Komponente die gemessenen Stresslevel dieser Plugins aggregieren.

Das Framework selbst bedarf weiterer Spezifikation. Es steht eine Definition der *Reasoner*-Komponente und der *Scheduler*-Komponente aus, die die interne Logik näher beschreiben. Diese zwei wichtigen Kern-Komponenten stellen ein Standbein des Frameworks dar. Um die Kommunikation zwischen den Plugins und dem Framework zu vereinheitlichen, müssen sämtliche Nachrichten, die zwischen den Plugins und dem Framework über die ActiveMQ gesendet werden, und die Nachrichtenkanäle auf der ActiveMQ genau spezifiziert werden. Hierzu möchte ich eine formale Beschreibung meines Systems aufstellen.

Nach der Vollendung der Spezifikation folgt die Implementierung der einzelnen Komponenten und Schnittstellen. Dieser Prozess soll zeitnah an die Spezifikation anschließen. Das gesamte Framework muss dann mit einigen Sensor-Implementationen verbunden werden. Mehrere Selbsttests sollen die prinzipielle Funktionsweise des Frameworks und der Plugins sicherstellen.

Neben der Spezifikation und Implementierung muss ich mir Gedanken über die Echtzeit-Anforderungen machen, die an das Framework gestellt werden. Das Problem hierbei könnte die ActiveMQ sein, die eine zeitliche Verzögerung der Bereitstellung der Sensorinformationen hervorruft.

Ist das ganze System lauffähig und einsatzbereit, möchte ich meine Annahmen mit einigen Probanden evaluieren. Hierzu muss ein genaues Testszenario ausgearbeitet werden. Ich muss die Aufgaben der Probanden genau festlegen und beschreiben sowie sämtliche Messergebnisse spezifizieren, die ich für die spätere Auswertung der Tests und Evaluierung meiner Annahmen benötige.

Literatur

[1] Barreto, Armando ; Gao, Ying ; Adjouadi, Malek: Pupil diameter measurements: untapped potential to enhance computer interaction for eye tracker users? In: *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*. New York, NY, USA : ACM, 2008 (Assets '08), S. 269–270. – ISBN 978-1-59593-976-0

[2] Becker-Asano, Christian: *WASABI: Affect Simulation for Agents with Believable Interactivity*, Universität Bielefeld, Doktorarbeit, März 2008

[3] Kersten, Mik: *Focusing knowledge work with task context.* Vancouver, BC, Canada, Canada, University of British Columbia, Dissertation, 2007. – AAINR26735

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

[5] Rauschan, Torsten: *Konzeption und Umsetzung innovativer Beobachtungsverfahren in Usability Untersuchungen*, Hochschule für angewandte Wissenschaften Hamburg, Bachelorarbeit, September 2009

[6] Valenti, R. ; Sebe, N. ; Gevers, T.: Facial Expression Recognition: A Fully Integrated Approach. In: *Workshop on Visual and Multimedia Digital Libraries*, URL http://www.science.uva.nl/research/publications/2007/ValentiVMDL2007, 2007

[7] Wu, Hao-Yu ; Rubinstein, Michael ; Shih, Eugene ; Guttag, John ; Durand, Frédo ; Freeman, William: Eulerian video magnification for revealing subtle changes in the world. In: *ACM Trans. Graph.* 31 (2012), Juli, Nr. 4, S. 65:1–65:8. – ISSN 0730-0301

[8] Zhai, Jing ; Barreto, A.B. ; Chin, C. ; Li, Chao: Realization of stress detection using psychophysiological signals for improvement of human-computer interactions. In: *SoutheastCon, 2005. Proceedings. IEEE*, april 2005, S. 415 – 420