5. Evaluation

> You cannot control what you cannot measure.
>
> *(Tom DeMarco, US-amerikanischer Informatiker, ⋆1940)*

Inhalt

Nachdem im vorherigen Kapitel die tatsächliche Realisierung besprochen wurde, behandelt dieses Kapitel die Evaluation des Systems. Der erste Abschnitt behandelt die vorgenommene Anwendungsstudie. Der zweite Abschnitt befasst sich kurz mit den Tests, die während der Entwicklung ausgeführt wurden.

5.1. Anwendungsstudie

Um die Anwendung zu evaluieren wurde eine Anwendungsstudie mit zwei potenziellen Nutzern des Systems durchgeführt. Ziel dieser Evaluation war die Identifizierung von Stärken und Schwächen der Anwendung, sowie die Sammlung etwaiger Verbesserungsvorschläge. Außerdem sollte überprüft werden, ob das System tatsächlich eine hilfreiche Prozessunterstützung liefert. Die Studie wurde als *Constructive Interaction* (vgl. O'Malley u. a. [20]) durchgeführt. Ein erfahrener Benutzer leitete, einen unerfahrenen Benutzer an, so dass diese beiden ihre Gedanken auf natürliche Art und Weise austauschten und dem Entwickler damit einen hilfreichen Einblick in ihre Denkmuster und ihre Anforderungen an das System erlangen konnte.

Die Studie des System offenbarte einige Schwachstellen des Systems und auch dazugehörige Verbesserungsmöglichkeiten. Die Schwachstellen und Verbesserungsmöglichkeiten sind dabei hauptsächlich auf die GUI bezogen.

Einer der Hauptkritikpunkte, war die fehlende Kennzeichnung von Änderungen im Metrikvorschlag. So kann der Kunde kaum sehen, was der Experte geändert hat. Als Idee wurde geäußert, dass man die Änderungen farblich markieren könnte. Dabei sollten die Farben eine Semantik besitzen. Rote Schrift würde beispielsweise bedeuten, dass ein Rechtschreibfehler behoben wurde, gelbe Schrift

würde auf eine Korrektur des Inhalts hinweisen, etc. Dies würde die Nutzer deutlich entlasten, wenn sie versuchen die Unterschiede zweier Versionen zu überprüfen. Die Darstellung zweier Versionen nebeneinander war ein weiterer Verbesserungsvorschlag, da bisher immer nur die aktuelle Version sichtbar ist. Sinnvoll wäre die Möglichkeit beliebige Versionen eines Vorschlags nebeneinander anzuzeigen und die Unterschiede zwischen beiden farblich zu markieren.

Außerdem wäre es sinnvoll neben den einzelnen Parameterfeldern weitere Textfelder anzubieten, in die eine Begründung für die Änderungen eingetragen werden kann. Alternativ könnten auch Fragen eingetragen werden, um Unklarheiten zu beseitigen. Dies wäre sinnvoll, da dann ein Großteil der Kommunikation über das System laufen könnte und beispielsweise keine Telefonate mehr geführt werden müssten. Man würde somit eine asynchrone Kommunikation unterstützen, da der derjenige, dem eine Frage gestellt wird, diese erst beantworten muss, wenn er dazu Zeit hat.

Außerdem sollte eine weitere Untergliederung der Spezifikationsparameter durch mehr Textfelder vorgenommen werden. So wären zwei getrennte Felder für Datensammlung und -validierung sinnvoll. Außerdem könnte man die Felder *Analyse und Interpretation* und *Ampeldarstellung* umstrukturieren. Zweckmäßig ist eine Teilung in drei Spalten mit Textfeldern, die als *Indikator*, *Ampeldarstellung* und *Interpretation* bezeichnet werden. In die Indikatorspalte würde man dann beispielsweise „Note zwischen sehr gut und gut“ eintragen. Dann würde in der Ampeldarstellungsspalte der Eintrag „grün“ folgen. Die Interpretation könnte dann „eine gut Note“ beinhalten. Dies entspricht auch einer Erfahrung die während der Entwicklung von MeDIC gemacht wurde. Durch diese Unterteilung wird das Wissen, im Gegensatz zu großen Textbereichen, sehr kompakt und übersichtlich dargestellt.

Grundsätzlich muss den Kunden wieder mehr Freiraum eingeräumt werden, da sich der Prozess, wie in Abschnitt 3.1 beschrieben, geändert hat. So sollten die Kunden auch Änderungen vornehmen können, bevor sie Bewertungen abgeben. Außerdem muss der FINISHED-Zustand wieder verlassen werden können, da möglicherweise auch später noch Modifikationen vorgenommen werden sollen. Dennoch soll der Zustand bestehen bleiben, um zumindest vorläufig abgeschlossene Vorschläge zu markieren.

Insgesamt wurde das neue System sehr positiv aufgenommen, da der Prozess nun deutlich systematischer vorangehen kann. Auch die klare Struktur bei der Nutzung des neuen Systems, die Datenbankunterstützung und die Technologieunabhängigkeit durch den Webservice sind als positive Merkmale zu nennen. Um den Fortschritt zu dokumentieren, werden in der Datenbank, wie bereits angesprochen, verschiedene Version des Vorschlags abgelegt, um später die Entwicklung nachverfolgen zu können.Das Wegfallen der Excel-Dokumente wurde ebenfalls als sehr gute Verbesserung bezeichnet, da diese erstens nicht genormt sind, zweitens nicht direkt an eine Datenbank angebunden sind und drittens keine systematische Unterstützung bieten. In dem Zusammenhang außerdem

ist die künftige Anzeige von Änderungen besonders hervorzuheben, die als sehr hilfreich eingestuft wurde. Die automatischen Benachrichtigungen per E-Mail wurden ebenfalls als sehr sinnvoll erachtet, um immer den aktuellen Stand zu kennen. Zusätzlich sparen sie auch Zeit, die Telefonate oder selbst geschriebene E-Mails kosten würden. Somit kann festgestellt werden, dass das System im Großen und Ganzen sehr hilfreich ist und seinen Nutzen durch die hier und in Abschnitt 6.1 beschriebenen Erweiterungen noch deutlich steigern kann.

5.2. Test des Programms

wBei der Evaluation des Programms wurden strukturierte Tests eingesetzt. Während der gesamten Entwicklung wurde in regelmäßigen Intervallen die Funktionalität getestet durch den Entwickler getestet. Als Testansatz wurden Black-Box-Tests gewählt. Die Persistent Entities wurden als reine nonmodale Datenhaltungsklassen als nicht unbedingt testenswert erachtet. Die Session Beans der Persistenz wurden informal während der Entwicklung getestet. Da die Controller die Hauptfunktionalität darstellen und die Persistenzoperationen beim Test der Controller ebenfalls mitgetestet werden, wurde entschieden nur die Controller formalen Unit-Tests zu unterziehen. Den Webservice ebenfalls per Unit-Test zu testen, wurde nicht als sinnvoll erachtet, da dieser keine zusätzliche Funktionalität gegenüber den Controllern bereitstellt. Die Oberfläche wurde aufgrund der nötigen Benutzerinteraktion während der Entwicklung Benutzertests durch den Entwickler und durch einen potenziellen Nutzer unterzogen. Außerdem wurde die oben beschriebene Anwendungsstudie vorgenommen. Bei den Tests der Controller wurde eine Verwendung von zustandsbasierten Tests als nicht passend angesehen, da die zu testenden Session Beans nicht über relevante Zustände verfügen, sondern eigentlich nur eine Fassade für den Zugriff auf die Persistent Entities darstellen. Dementsprechend wurde das Testen per N+ und FREE (vgl. Binder [5]) verworfen. Da der Aufwand für das relativ kleine Projekt auch für SCENT [21] zu groß war, wurde auch diese Testfallherleitung nicht verwendet. Außerdem wurden das UC-basierte Testen als zu GUI-orientiert betrachtet. Letztendlich war die natürlichste Wahl für Tests, Äquivalenzklassen- bzw. Grenzwerttests. Die verwendeteten Testfälle sind in Anhang in (vgl. Abschnitt A.3.1) zu finden.

Eigentlich sollten die Unit-Tests automatisiert ausgeführt werden. Da jedoch bei normaler Benutzung von JUnit die Injektionsmechanismen des EJB-Containers nicht mehr funktionieren, muss man ein Framework wie OpenEJB [28] einsetzen. In der Kürze der Zeit konnte dieses aber nicht zufriedenstellend konfiguriert werden, weswegen der *Universal Testclient* des *Rational Application Developers 7.5* eingesetzt wurde. Damit konnte zwar nicht automatisiert getestet werden, aber die Testfälle konnten dennoch zufriedenstellend abgearbeitet werden.