Artikelserie: Kommunalsoftware auf dem Prüfstand[1]

(Auszug aus der Veröffentlichung im OKKSA-Newsletter Nr. 13 vom 08.03.2006)

Teil 3: Programminteraktion – Wer weiß noch was er tut?

(von Dr. Uwe Schwochert)

In den letzten Beiträgen dieser Serie wurde betrachtet, welche Anforderungen an die Fähigkeit einer Software zur Datenrecherche und –darstellung bestehen.

Nun wollen wir uns der täglichen Arbeit mit dem Programm zuwenden.

Grundsätzlich gibt es viele Aspekte, unter denen Bedienfreundlichkeit von Software betrachtet werden kann. Seitens der Normung sei hier auf die DIN EN ISO 9241, Teil 10 (Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten / Grundsätze der Dialoggestaltung) sowie auf die im Rahmen des Media@Komm-Projektes entstandene PAS 1020 (Designrichtlinien und Forumulierungsstandards für E-Government – Applikationen) verwiesen.

Unter Prüfaspekten galt es bei der Formulierung der Kriterien im OKKSA-Anforderungskatalog die Programmleistungen zu beschreiben, die ein Mindestmaß an sicheren Bedienabläufen beim Anwender gewährleisten. Anders ausgedrückt: bei der üblichen Nutzung sollte das Programm nicht selbst Ursache für mögliche Bedienfehler sein können.

"Übliche Nutzung" bezieht sich dabei auf den durchschnittlichen kommunalen Fachanwender. In der Konsequenz können all die Usability-Aspekte, die sich aus der Spezifik einzelner Anwendungssituationen ergeben, nicht in einem allgemeinen Kriterienkatalog berücksichtigt werden.

[FÜ2.1] Während der Bearbeitung kann der Benutzer den jeweiligen Bearbeitungskontext eindeutig erkennen. (Kann-Kriterium)

Das Programm soll durch die Gestaltung der Masken und die Auswahl der dargestellten Informationen eine Kontrolle darüber ermöglichen, mit welchen Daten man gerade arbeitet. Ein einfaches Beispiel wäre ein Textverarbeitungsprogramm, welches im Fenstertitel darstellt, welche Datei gerade geöffnet ist. Bei einem Finanzprogramm soll man bei der Haushaltsplanung erkennen, mit welchem Konto man gerade arbeitet.

Diese Anforderungen sind nahe liegend und dürften keine Probleme bereiten. Nicht ganz so selbstverständlich ist die Forderung, dass auch bei der Arbeit mit Datenbereichen erkennbar sein soll, dass man gerade mit einer Informationsauswahl arbeitet. So soll bei Übersichten der zur Anwendung kommende Filter erkennbar sein.

[FÜ2.2] Unterstützte Vorgänge und fachliche Abläufe sind im Programmablauf wieder erkennbar. Bei folgenschweren Aktionen warnt das Programm den Bearbeiter und ermöglicht ihm den Abbruch. (Muss-Kriterium)

[FÜ2.3] Es gibt eine für den Benutzer nachvollziehbare für die verschiedenen Programmteile einheitliche Eingabesystematik für standardisierte Bedienoperationen wie „Speichern", „Verlassen ohne Speichern" oder „Hilfe aufrufen". Diese Eingabesystematik soll

[1] Hinweis: Die in der Artikelserie genannten Kriterien stammen aus der aktuellen Fassung des "OKKSA Anforderungskatalogs für Fachprogramme in der Öffentlichen Verwaltung - Teilbereich Fachübergreifende Programmanforderungen".

Im Kriterienkatalog sind neben den Kriterien auch Rechts- und Normungsgrundlagen genannt, aus denen die genannten Kriterien abgeleitet werden.

Momentan (März 2006) wird der Anforderungskatalog entsprechend den Vorgaben des OKKSA-Vereins durch das Fachgremium aktualisiert.

sich an den in der jeweiligen Systemumgebung gängigen Standards orientieren. (Muss-Kriterium)

Neben der Kennzeichnung der jeweils zur Bearbeitung anstehenden Informationen benötigt der Anwender auch die Kenntnis über den Status der Bearbeitung.

Fragen wie "Habe ich jetzt schon gebucht?" "Wurde der Bescheid schon gedruckt?" "Ist der Vorgang abgeschlossen?" sollten sich durch die Bildschirmdarstellung beantworten lassen.

Darüber hinaus muss der Anwender auch erkennen können, was passiert, wenn er ein Bedienelement betätigt. In der Gestaltung der Dialoge zeigt sich dann auch das Know-how des Programmentwicklers in Bezug auf (1) die Standards des Betriebssystems und (2) die Arbeitsabläufe des Anwenders. Aber selbst Neulinge in der Programmierung sollten erkennen können, welche Erwartungshaltung ein Anwender mit dem Schließen eines Fensters oder dem Betätigen eines [OK]-, [Buchen!]- oder [Buchen ...]-Buttons verbindet.

Auch hier wieder der Hinweis, dass selbst umfangreiche Schulungen und Handbücher die Erwartungskonformität der Dialoggestaltung nicht ersetzen können. Hilfreicher sind hier Hilfestellungen direkt auf der Maske, wie sie unter Windows ja üblich sind: gelbe Sprechblasen, wenn man mit dem Mauszeiger länger über einem Bedienelement verweilt oder ein Kontexthilfe per rechte Maustaste.

[FÜ2.4] Das Programm bietet die Möglichkeit, die Eingabe innerhalb eines bildschirmorientierten Eingabebereiches zu einem Vorgang (zum Beispiel Maske) vor der erkennbaren Bestätigung der Werte (zum Beispiel Button anklicken oder Befehl zur Verarbeitung eintippen) über den gesamten Eingabebereich zu korrigieren bzw. zu verwerfen.

Im Grunde geht es hier um die Trennung zwischen Datenerfassung und Datenverarbeitung. Die Anforderung basiert auf der Erkenntnis, dass während einer Datenerfassung Situationen auftreten können, die die nachfolgende Verarbeitung der Daten nicht sinnvoll machen. Im einfachsten Fall hat der Benutzer einfach die falsche Maske aufgerufen. Aber auch, wenn er richtig liegt, könnte es sein, dass er nicht weiterarbeiten kann. Zum Beispiel könnte die Änderung einer Adresse über die entsprechende Maske daran scheitern, dass die neue PLZ von der entsprechenden Plausibilitätsprüfung abgelehnt wird. Auch wenn der Bearbeiter vielleicht vorher schon den neuen Straßennamen eingegeben hat, soll das Programm noch ein Abbrechen und eine Wiederherstellung der alten Adresse erlauben. In der Datenbank sollen vorerst keine Informationen geändert werden!

Das Abbrechen kann z. B. über einen [Abbrechen]-, [Schließen]- oder [Verwerfen] passieren. Aber auch eine Rückgängig-Funktion wie bei Office-Anwendungen könnte zum selben Ziel führen.

Bei den genannten Kriterien geht es nicht um Bequemlichkeit oder Nutzerfreundlichkeit. Es geht vielmehr darum, dass der Programmanwender der mit seiner Arbeitsaufgabe verbundenen Verantwortung gerecht werden kann. Und dabei sollte ihn die Software nicht behindern.

Nächster Teil der Serie: Dateneingabe – muss das Programm alles akzeptieren?