Konsistenz

Ein wichtiges Ziel für Designer ist eine konsistente Benutzerschnittstelle. Die Definition von Konsistenz ist jedoch schwer zu fassen und hat mehrere Ebenen, die manchmal miteinander in Konflikt geraten, und manchmal ist Inkonsistenz auch vorteilhaft. Das Argument für Konsistenz ist, dass eine Befehlssprache oder eine Folge von Aktionen geordnet, vorhersagbar, durch wenige Regeln beschreibbar und von daher leicht zu erlernen und zu erinnern sein sollte. Diese überlappenden Konzepte werden durch ein Beispiel verdeutlicht, das zwei Arten von Inkonsistenz zeigt (A veranschaulicht das Fehlen eines jeglichen Versuches zur Konsistenz, und B zeigt Konsistenz mit nur einem Verstoß):

Konsistent	**Inkonsistent A**	**Inkonsistent B**
Zeichen löschen / einfügen	Zeichen löschen / einfügen	Zeichen löschen / einfügen
Wort löschen / einfügen	Wort entfernen / bringen	Wort entfernen / einsetzen
Zeile löschen / einfügen	Zeile zerstören / schaffen	Zeile löschen / einfügen
Absatz löschen / einfügen	Absatz töten / erschaffen	Absatz löschen / einfügen

Jede der Aktionen in der konsistenten Version ist die gleiche, hingegen variieren die Aktionen bei der inkonsistenten Version A. Die Verben der inkonsistenten Aktion sind alle akzeptabel, aber ihre Vielfalt suggeriert, dass man zum Erlernen länger braucht, mehr Fehler machen kann, Anwender langsamer werden und es schwerer haben, sich an sie zu erinnern. Die inkonsistente Version B ist in gewisser Weise noch boshafter, weil es eine einzelne unvorhersehbare Inkonsistenz gibt, die so dramatisch hervorsticht, dass diese Sprache wahrscheinlich gerade wegen ihrer Inkonsistenz behalten wird. Um diese Vorstellungen festzuhalten, schlug Reimer (ICHBI) eine Aktionsgrammatik vor, um zwei Versionen einer grafischen Systemschnittstelle zu beschreiben. Sie zeigte, dass die Version mit der einfacheren Grammatik leichter zu erlernen war. Payne und Green (1986) erweiterten ihre Arbeit, indem sie sich den vielfältigen Ebenen der Konsistenz (Lexik, Syntaktik und Semantik) mittels eines strukturierten Zeichensystems, das sie Aufgabenaktions-Grammatik (task-action grammars - TAGs) nannten, zuwandten. Sie beschäftigen sich ebenfalls mit einigen Aspekten der Vollständigkeit einer Sprache, indem sie versuchen, einen vollständigen Satz an Aufgaben zu charakterisieren. Beispielsweise stellt hoch, runter und links einen unvollständigen Satz von Bewegungsaufgaben für den Pfeil-Cursor dar, weil rechts fehlt. Wenn erst einmal der vollständige Satz der Zuordnungen von Aktionen und Aufgaben niedergeschrieben wurde, kann die Grammatik der Befehlssprache dagegen getestet werden, um Vollständigkeit zu