Gestaltungsrichtlinien für Editor-Simulatoren für graphartige Dokumente

Udo Kelter
Fachgruppe Praktische Informatik
Fachbereich Elektrotechnik und Informatik
Universität Siegen, 57068 Siegen
kelter@informatik.uni-siegen.de

Zusammenfassung: Unter Editor-Simulatoren verstehen wir Simulatoren, die Abläufe in einem System simulieren und die es zugleich erlauben, das System zu verändern. Wir untersuchen hier Editor-Simulatoren, die primär in Vorlesungen eingesetzt werden und die Vorgänge in graphartigen Dokumenten (z.B. Netzplänen oder Objektgraphen) illustrieren. Vorgeschlagen werden Richtlinien, wie die Bedienschnittstelle und die Funktionalität von solchen Editor-Simulatoren zu gestalten ist.

1 Einleitung

In der Informatik werden vielfach Graphen zur Modellierung von Informationen benutzt; dementsprechend häufig werden Algorithmen auf Graphen unterrichtet. Der Unterricht kann oft dadurch mit neuen Medien unterstützt werden, daß die Vorgänge in den Graphen durch animierte Simulationen visualisiert werden. Unter einer **Simulation** verstehen wir hier die schrittweise Darstellung einer Durchführung eines Algorithmus, unter einem **Simulator** ein Softwaresystem, das diese Darstellung erzeugt.

Vielfach will man in einer Unterrichtssituation die Struktur und/oder Beschriftung der Graphen ändern. Benötigt werden hierfür Softwaresysteme, die zugleich Grapheditoren und Simulatoren sind. Es stellt sich daher die Frage, wie solche **Editor-Simulatoren** äußerlich und bzgl. ihres Funktionsumfangs zu gestalten sind.

Dieses Papier berichtet über Erfahrungen, die bei der Entwicklung mehrerer Versionen eines Editor-Simulators, der die Vorwärts-/Rückwärts-Rechnung in Netzplänen simuliert, gewonnen wurden. Das System wurde im Rahmen des BMBF-Projekts MuSofT [Mu02] entwickelt. Formuliert werden die Erfahrungen in Form von Richtlinien, wie Editor-Simulatoren gestaltet werden können bzw. sollten.

Existierende Gestaltungsrichtlinien. Editoren bzw. Simulatoren fallen in die Klasse interaktiver, graphischer Softwaresysteme. Somit sind zunächst die generellen Regeln der Software-Ergonomie (Aufgabenangemessenheit, Steuerbarkeit, usw.; vgl. [DIN66234]) anzuwenden. Diese Regeln sind allerdings sehr abstrakt und eher als Bewertungskriterien bei einer (nachträglichen) Bewertung eines Systems verwendbar.

Wesentlich konstruktiver und detaillierter sind Gestaltungsrichtlinien, die zu konkreten GUI-Frameworks oder PC-Betriebssystemen gehören. Diese Regelwerke orientieren sich i.d.R. an Bürosoftware und sind auf eine relativ intensive, längerfristige Nutzung der Systeme ausgerichtet. Auf Editor-Simulatoren sind die Regelwerke nur teilweise und bedingt anwendbar: Animationen setzen oft komplett neue GUI-Elemente und graphische Darstellungsformen ein. Ferner werden sie meist nur sehr kurzfristig und einmalig benutzt.

Editor-Simulatoren als spezielle Applikationsklasse. Die in diesem Papier betrachteten Editor-Simulatoren stellen eine spezielle Klasse von Applikationen dar, die eigene Gestaltungsrichtlinien benötigen und die wie folgt charakterisiert werden können:

- Animiert werden Vorgänge in graphartigen Strukturen, z.B. Suchalgorithmen in Suchbäumen, Vorwärts-/Rückwärts-Rechnung in Netzplänen, Operationen auf Konfigurations- bzw. Versionsgraphen, Aufrufsequenzen in Objektgraphen u.ä.
- Die Beispiel-Graphen, an denen die Algorithmen simuliert werden, sollen sowohl hinsichtlich der Graphstruktur als auch der anderen Inhalte in der Unterrichtssituation ad hoc veränderbar sein, d.h. bei den hier vorliegenden Systemen handelt es sich nicht nur um Simulatoren, sondern zugleich um (einfache) Editoren für die entsprechende Graphklasse.
- Die Systeme sollen sowohl bei einem Vortrag im Hörsaal (unter Einsatz eines Beamers) verwendet werden als auch an PCs beim Selbststudium als auch innerhalb von Übungsgruppen zum Durchspielen von Übungsaufgaben. Hieraus folgt, daß der zu veranschaulichende Algorithmus komplett nachvollzogen werden, er im Simulator also implementiert sein muß[1].

2 Anforderungen und Randbedingungen

Zeitlicher Rahmen. Wir gehen davon aus, daß die Simulationen primär in einen Vortrag eingebettet sind. D.h. die Begriffe, statische Datenstrukturen und/oder Struktur der Algorithmen werden vorab durch Vortrag unterstützt mit Folien präsentiert. Eine oder mehrere Simulationen visualisieren gezielt die dynamischen Abläufe. Hieraus folgt, daß der Zeitrahmen zum Durchgehen von einem oder mehreren Beispielen typischerweise auf den Bereich von 1 - 5 Minuten eingeschränkt werden muß. Hieraus ergeben sich folgende Anforderungen:

[1] Es handelt sich natürlich nicht um eine normale Implementierung, denn die würde keine Ausgaben erzeugen, die einzelne Zwischenschritte anzeigen.

- Das System soll schnell gestartet werden können. Zeitraubende Lade- und Einstellungsvorgänge vor Beginn der eigentlichen Simulation sollten nicht notwendig sein.
- Es ist davon auszugehen, daß der Simulator oder ein ähnliches Programm den Zuhörern *nicht* vertraut ist. Daher muß i.d.R. die Bedeutung der wichtigsten Elemente der Anzeige des Systems kurz erklärt werden. Für diese Erklärungen stehen nur ca. 30 - 60 Sekunden Zeit zur Verfügung. Da man pro Bedien- oder Anzeigeelement 1 - 2 Sätze brauchen wird, folgt im Umkehrschluß, daß nur ca. 3 - 6 erklärungsbedürftige Bedien- oder Anzeigeelemente vorhanden sein dürfen.

Projektionsfläche und Bildauflösung. Mit modernen, sehr guten Beamern lassen sich ähnliche Anzeigeergebnisse erzielen wie mit hochauflösenden Bildschirmen. In der Praxis an Universitäten ist man aber meist deutlich entfernt von solchen Idealverhältnissen: Vielfach werden nur tragbare Kleinbeamer eingesetzt, deren Lichtstärke für eine Projektionsfläche von maximal ca. 2.5 m * 1.8 m reicht, speziell bei ungünstigen Untergründen. Vielfach wird auf abgetönte, lichtschluckende Wände projiziert und nicht auf gut reflektierende Leinwände. Vielfach wird in Räumen mit nur 3 Meter Höhe und waagerechtem Boden unterrichtet; in einem großen Teil des Raums ist dann je nach Bestuhlung nur der obere Teil der Wand sichtbar.

Bei Editor-Simulatoren, die an vielen Standorten einsetzbar sein sollen, muß von eher ungünstigen Voraussetzungen ausgegangen werden. Sicherheitshalber muß man daher von einer Informationsdichte ausgehen, die einem Bildschirm mit 800*600 Punkten Auflösung entspricht.

Da die Simulationen i.d.R. auch für das Selbststudium an privaten Rechnern (mit relativ guten Bildschirmen) gedacht sind, sollte deren leistungsfähigere Anzeige optional ausnutzbar sein.

3 Allgemeine Richtlinien

Aus den in Abschnitt 2 erwähnten Randbedingungen lassen sich einige allgemeine Richtlinien ableiten, die nicht nur für Editor-Simulatoren gelten, sondern auch für diverse ähnliche Arten von Multimedia-Systemen, z.B. nicht modifizierbare Simulationen bzw. Animationen in graphischen Strukturen.

3.1 Richtlinien zum Fensteraufbau

Das Ausgabefenster eines Editor-Simulators sollte sollte folgendermaßen aufgebaut sein:

1. Es sollte nur ein ungeteiltes Hauptfenster geben, also keine Mehrfenstertechnik oder geteilte Fenster.
2. Es sollte nur eine Buttonleiste mit Buttons für die 6 - 10 wichtigsten Funktionen vorhanden sein.

3. Die Buttons sollten hinreichend groß (ca. zwei- bis dreimal größer als Buttons in gängigen Office-Paketen) sein.
4. Die Buttons sollten sowohl ein graphisches Symbol als auch eine Beschriftung enthalten.

   Tooltips und ähnliche Hilfen reichen nicht aus, da sie nur für kurze Zeit, sofern der Vortragende überhaupt über die entsprechende Stelle kommt, lesbar sind.
5. Die Fonts von Beschriftungen und sonstigen Texten sollten bei 1024*768 Auflösung ca. 16 - 20 Punkte groß sein. Wenn die Größe einstellbar ist, ist dies vorteilhaft, sofern auch alle graphischen Objekte angepaßt werden; letzteres verursacht einen sehr hohen Realisierungsaufwand.

Ein Problem bei der Beschriftung von Buttons ist oft die Länge der deutschen Kommandonamen. Englische Bezeichnungen sind meist wesentlich kürzer, führen aber zu einem unschönen Sprachengemisch, da in den Tooltips und den Erklärungen zur Simulation (Protokollausgaben) i.d.R. deutsche Erklärungen ausgegeben werden sollten.

3.2 Richtlinien zum Einsatz von Farben

Hier sind natürlich zunächst allgemeine Grundregeln zu beachten (wenig Farben benutzen, einzelnen Farben eine klare Bedeutung geben, Farben nur für zusätzliche Information verwenden, sinnvolle Farbkontraste wählen usw.). Wir konzentrieren uns hier auf ein spezielles Problem, das typisch in den o.g. Einsatzszenarien ist: Farbverfälschungen. Diese treten aus zwei Gründen auf:

- durch Beamer
- infolge Projektion auf abgetönt gestrichene oder vergraute Wände.

Die Farbverfälschungen können in der Praxis so gravierend sein, daß Texte oder graphische Inhalte kaum noch erkennbar sind. Hieraus resultiert die generelle Anforderungen, daß die Farben in einem Editor-Simulator einstellbar sein sollten, daß das System also in diesem Sinne “beamertauglich” ist. Bzgl. der Einstellung der Farben können zwei Fälle unterschieden werden:

- Vor der eigentlichen Lehrveranstaltung kann eine Probevorführung gemacht werden, d.h. der Vortragsraum ist zugänglich, und es ist genug Zeit hierfür vorhanden. Dies ist z.B. bei Vorlesungen, die regelmäßig im gleichen Hörsaal stattfinden, der Fall. In diesem Fall sollte es möglich sein, die Farben einzeln zu verstellen. Von Farbverfälschungen sind i.a. nicht alle Farben betroffen, die nicht betroffenen Farben können natürlich unverändert bleiben. Im schon erwähnten Netzplansimulator kann für rund 20 Anzeigedetails die Farbe individuell eingestellt werden. Die bisherige Erfahrung mit dem System zeigt, daß nur sehr wenige Farben gegenüber ihrer Grundeinstellung verändert werden.

  Es ist hier akzeptabel, wenn die Einstellungsarbeiten ca. 5 - 10 Minuten dauern.

– Eine Probevorführung ist nicht möglich, das Problem wird erst während der Vortragssituation erkennbar. In diesem Fall kann für Korrekturmaßnahmen i.a. nur sehr wenig Zeit geopfert werden, da die Zeit ohnehin knapp ist und da ansonsten zu sehr vom Lehrgegenstand abgelenkt wird. Akzeptabel ist ein Zeitbedarf von ca. 10 - 15 Sekunden.

  Unter diesen Randbedingungen können nur sehr grobe "Einstellschrauben" bedient bzw. ausprobiert werden. Mögliche technische Lösungen sind z.B. eine Einstellmöglichkeit, bei der alle Farben gleichmäßig aufgehellt oder getönt werden, oder vorbereitete Farbpaletten, zwischen denen umgeschaltet wird.

4 Richtlinien für das Gesamtsystem

Bei einem Editor-Simulator müssen normalerweise zwei Hauptzustände unterschieden werden: der Editiermodus und der Simulationsmodus. Im Editiermodus sind Funktionen zum Verändern des Graphen und anderer Daten im dargestellten Modell verfügbar. Im Simulationsmodus kann die Simulation gestartet und gesteuert werden. Bzgl. der Modushandhabung sollten folgende Richtlinien beachtet werden:

1. Es sollte möglich sein, sehr rasch zwischen beiden Modi umzuschalten.
2. Es sollte ständig deutlich angezeigt werden, in welchem Modus sich das System zur Zeit befindet. Es sollte nicht nur der aktuelle Modus, sondern auch die Gesamtmenge aller möglichen Modi erkennbar sein.
3. Es sind zwei Hauptgruppen von Buttons bzw. Bedienelemente vorhanden: (a) Buttons zum Umschalten des Modus und zur Anzeige des aktuellen Modus, (b) im jeweiligen Modus vorhandene Funktionsbuttons. Diese beiden Gruppen sollten, speziell wenn sie zusammen in der Button-Leiste stehen, deutlich voneinander getrennt sein.
4. Funktionen des jeweiligen Modus sollten in einem anderen Modus nicht aufrufbar sein, selbst wenn dies technisch möglich wäre. M.a.W. muß der Modus immer explizit umgeschaltet werden.

   Der Aufwand für die explizite Umschaltung wirkt zunächst lästig. Das Weglassen dieses Bedienschritts schadet in der Unterrichtssituation aber mehr als es nützt, weil es zu einem impliziten Moduswechsel führt, der den Zuhörern erklärt werden müßte. Ein expliziter Wechsel ist hier in Wirklichkeit einfacher.
5. Beim Umschalten in den Editiermodus und anschließendem Editieren sollte, sofern nur für die Simulation irrelevante Daten geändert werden (Beispiel: Verschieben von Knoten auf der Zeichenfläche), der bisher erreichte Zustand der Simulation nicht verlorengehen. Wenn dennoch die Simulation anschließend neu gestartet werden soll, dann kann dies durch ein explizites Rücksetzen erreicht werden.

5 Richtlinien für die Editierfunktionen

Gängige CASE-Editoren für graphartige Modelle (z.B. für diverse UML-Diagrammtypen) weisen eine Vielzahl von Funktionen und Funktionsvarianten auf. Bei der dort unterstellten intensiven Nutzung der Systeme ist diese Gestaltung auch angemessen, nicht hingegen bei den Einsatzbedingungen, die wir für Editor-Simulatoren unterstellt haben. Hier sollte der Funktionsumfang auf den wirklich notwendigen Umfang eingeschränkt werden.

I.f. diskutieren wir nur Standardeditierfunktionen, die bei beliebigen Graphen anwendbar sind. Für spezielle Arten von Graphen kann es weitere Editierfunktionen geben, für die analoge Betrachtungen anzustellen sind.

Auf alle Fälle werden folgende Standardeditierfunktionen benötigt:

- *anlegen* von Diagrammelementen (Knoten, Kanten, Stützpunkte in Kanten, ggf. Kommentare)
- *löschen* von Diagrammelementen
- *verschieben* von Knoten und von Stützpunkten von Kanten[2]
- *beschriften* von Knoten und von Kanten

Bei den folgenden in CASE-Werkzeugen üblichen Standardeditierfunktionen ist es fraglich, ob sie in Editor-Simulatoren wirklich benötigt werden und ob sie nicht mehr schaden als nutzen, weil das System ihretwegen komplexere Zustände hat, die die Bedienung komplizierter machen und den Erklärungsbedarf erhöhen:

- *kopieren* von Diagrammelementen: Diese Funktion ist nur für Knoten sinnvoll, wobei ggf. die hereinkommenden Kanten mitkopiert werden.

  Sofern man überhaupt eine solche Funktion hat, will man i.d.R. Gruppen von Knoten bilden können, die auf einmal mitsamt der innenliegenden Kanten kopiert werden (s.u.).

  Kopierfunktionen sparen viel Zeit beim Arbeiten mit umfangreichen Beispielen; bei Vorführungen werden aber meist nur kleine Beispiele verwendet, und die eventuellen Modifikationen der Beispiele sind nicht sehr umfangreich. Der Vorteil der Arbeitseffizienz kommt daher nicht zum Tragen.

- temporäre *Gruppierung* für das Löschen und Verschieben (und ggf. Kopieren): Für das Definieren einer Gruppe muß i.a. ein eigener Editierzustand vorgesehen werden, in dem Diagrammelemente selektiert und deselektiert werden können. Die bisher selektierten Diagrammelemente müssen irgendwie erkenntlich sein; dies verkompliziert die Zustandsanzeige.

  Der Gewinn an Arbeitseffizienz gegenüber dem einzelnen Löschen bzw. Verschieben der Diagrammelemente ist bei kleinen Graphen nur gering.

- *Skalierung* von Knoten, die meist als Rechtecke oder Kreise dargestellt werden: Es hängt vom Dokumenttyp ab, ob diese Funktion sinnvoll ist, meist ist sie es nicht.

  Sofern die Schriftgrößen verstellbar ist, wird i.d.R. eine explizite oder implizite Veränderung der Größe der Knotendarstellung erforderlich sein.

[2] Kanten lassen sich i.a. nicht als Ganze sinnvoll verschieben.

– *rotieren*, *spiegeln* usw.: derartige Funktionen sind nur in Ausnahmefällen abhängig vom Dokumenttyp sinnvoll.

– *Raster und Fangmodus*: Diese Funktionsvarianten sind für das Erzeugen und Verschieben von Diagrammelementen relevant. Sie können realisiert werden, ohne die Zustandsanzeige zu verkomplizieren. Ob sie auch inhaltlich sinnvoll sind, hängt vom Dokumenttyp ab.

Im Editiermodus sollten in der Buttonleiste nur die wichtigsten Funktionen angezeigt werden. Weitere Funktionen können in Kontextmenüs angeordnet werden, wobei dann natürlich zunächst das betroffene Diagrammelement selektiert sein muß. Da man in Kontextmenüs "versteckte" Funktionen nicht zielgerichtet finden kann, sollte zusätzlich ein Hauptmenü vorhanden sein, in dem alle Editierfunktionen angewählt werden können. Sofern zunächst die Funktion gewählt wird, sollte der Editor anschließend in einem Zustand sein, in dem nur noch dazu passende Diagrammelemente selektiert werden können.

6 Richtlinien für die Simulationsfunktionen

Die Gestaltung der Simulationsfunktionen hängt natürlich sehr stark von dem zu simulierenden Algorithmus ab. Einige Aspekte treten bei vielen Algorithmussimulationen auf; für diese können allgemeine Richtlinien aufgestellt werden.

Steuerung der Simulationsschritte. Für die Steuerung der Simulationsschritte sollten die folgenden Funktionen vorhanden sein:

– Einzelschritt
– automatischer Ablauf mit wählbarem Tempo; das Tempo sollte modifizierbar sein, z.B. durch einen graphischen Schieberegler
– Ablauf bis zum Ende ohne Ausgabe der Zwischenzustände

Alle vorstehenden Schritte sind vorwärts gerichtet. In manchen Fällen ist es nützlich, im Vortrag mehrfach die Lage vor und nach einem Rechenschritt anzeigen zu können; hierfür benötigt man eine Funktion, die einen Einzelschritt (oder mehrere) rückwärts ausführt.

Variable Schrittgröße. Bei vielen Algorithmussimulationen sind verschiedene Schrittgrößen sinnvoll. Beispielsweise sind bei der Vorwärtsrechnung in einem Netzplan zwei Schrittgrößen sinnvoll:

– die komplette Bearbeitung eines Knotens in einem Schritt
– Berechnung eines einzelnen Werts innerhalb eines Knotens

Die Menge der Schrittgrößen muß für jede Algorithmussimulation individuell bestimmt werden.

Zwischen den verschiedenen Schrittgrößen sollte man jederzeit umschalten können. Die aktuell gewählte Schrittgröße muß jederzeit erkennbar sein. Daher Bieten sich Radio-Buttons oder eine Combo-Box als GUI-Element für die Auswahl an. Für die Combo-Box spricht, daß man hier mehr Platz hat, um die Schrittgrößen zu benennen bzw. zu beschreiben. Nachteilig ist, daß man die meiste Zeit nicht sieht, welche Alternativen bestehen.

Die Schrittgrößen sollten intuitiv verständliche Einheiten sein, damit man kompakte Benennungen in den GUI-Elementen verwenden kann und damit die Zuschauer die Einheiten leicht verstehen.

Textuelle Erläuterungen zum aktuellen Simulationsschritt. Oft ist es wünschenswert, zum gerade durchgeführten Schritt eine textuelle Erläuterung zu bekommen, was passiert ist. Für die "Sprache", in der die Erläuterung formuliert wird, gibt es zwei Alternativen:

- Umgangssprache
- eine Programmiersprache (oder Pseudocode). Da einzelne Anweisungen aus einem Programm ohne den Kontext oft nicht verständlich sind, liegt es nahe, den kompletten Algorithmus als Programmtext anzuzeigen und die aktuelle Anweisung durch Blinken oder ähnliche Maßnahmen hervorzuheben, ähnlich wie in Debuggern. Wegen des hohen Platzbedarfs ist diese Alternative im Hörsaal wenig brauchbar, für das Heimstudium aber sehr interessant.

In beiden Fällen entsteht das Problem, wie die Operanden, die in dem Rechenschritt benutzt werden, erkennbar gemacht werden. Ein Rechenschritt bearbeitet i.d.R. einen Knoten oder eine Kante des Graphen bzw. einzelne dort vorhandene Werte; ohne explizite Hinweise sind die Operanden meist nicht schnell genug erkennbar. Der Simulator muß also solche Hinweise liefern. Möglichkeiten hierfür sind, die Operanden einzufärben oder blinken zu lassen, die Operandenwerte in der Textzeile zu wiederholen oder Erläuterung in einer Sprechblase anzuordnen; diese Ansätze können auch kombiniert werden. Infrage kommen aber nur Methoden, bei denen die Identifizierungshinweise berechnet werden können, denn der Simulator muß ja mit beliebigen Graphen arbeiten können, d.h. die Menge der Operanden und ihre Anordnung im Graphen ist nicht konstant.

Literatur

[DIN66234] DIN 66234, Teil 8: Bildschirmarbeitsplätze - Grundsätze ergonomischer Dialoggestaltung; 1988

[Mu02] BMBF-Projekt MuSofT - Multimedia in der Softwaretechnik; http://www.musoft.org; 2002

[Sc02] Schulmeister, Rolf: Grundlagen hypermedialer Lernsysteme. Theorie - Didaktik - Design (3. Auflage); Oldenburg; 2002