Aufbau der Infrastruktur für die Durchführung der Übungen

Bei der Durchführung der ersten Übungsaufgabe werden Sie mit einem Computersystem konfrontiert, mit welchem Sie auch alle weiteren Übungsaufgaben durchführen müssen. Deshalb behandle ich nun die Struktur, die Sie vor Ihrem geistigen Auge haben sollten, wenn Sie an Ihrem Übungsrechner sitzen.

In Bild 11 sind Rechner gezeigt, die über das Internet verbunden sind. Einer ist Ihr Arbeitsplatzrechner im Übungsraum, mit dem Sie über Bildschirm, Tastatur und Maus verbunden sind. Dieser Rechner tritt als Client von sog. Servern auf. Unten im Bild ist der Serverrechner *wendtstud1* gezeigt, der irgendwo im HPI-Gebäude steht. Auch an diesen Rechner kommt man heran über Tastatur, Maus und Bildschirm, aber dieser Zugang ist Ihnen verwehrt, er steht nur ganz bestimmten Mitarbeitern des HPI zur Verfügung.

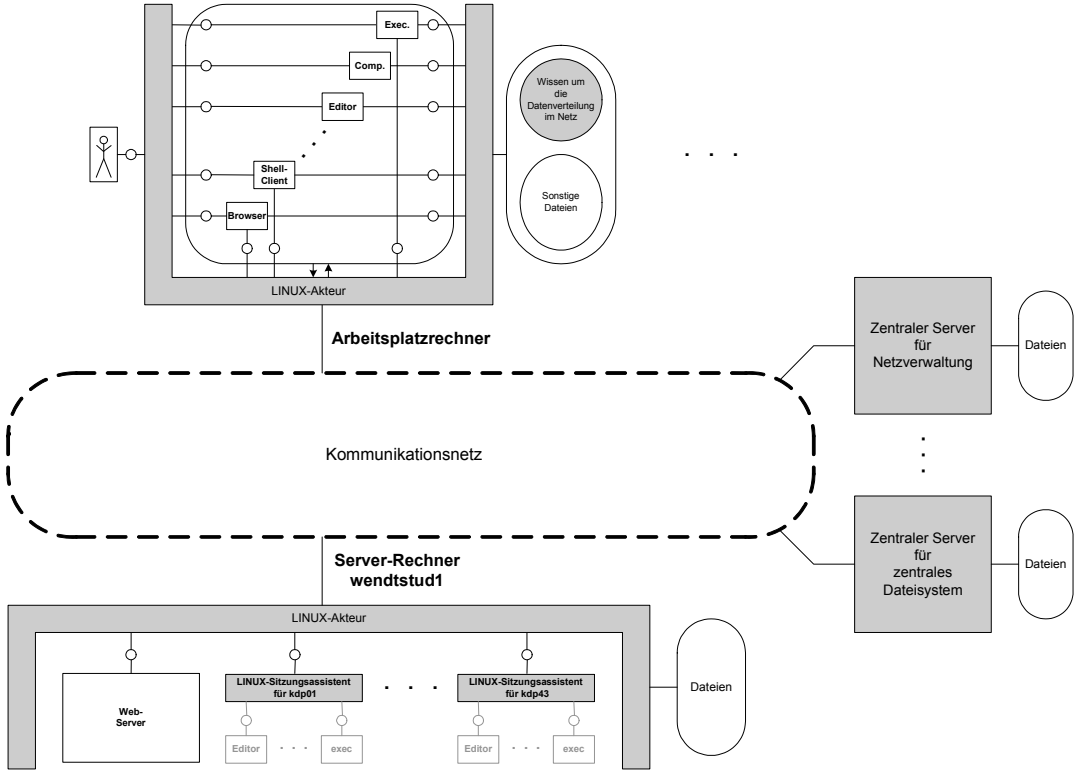

Bild 11 Client-Server-Konfiguration

Wenn Sie Ihren Rechner einschalten, werden Sie aufgefordert, sich mit Benutzernamen und Passwort anzumelden. Es muss also bereits zu Beginn ein Akteur existieren, von dem diese Aufforderung ausgeht. Dieser Akteur ist das Betriebssystem Ihres Arbeitsplatzrechners. Nachdem Sie sich korrekt angemeldet haben, kommunizieren Sie immer noch nur mit dem Betriebssystemakteur, nur wird er Ihnen nun Dinge ermöglichen, die er Ihnen vor der korrekten Anmeldung noch vorenthalten hat. Er wird Ihnen nun nämlich anbieten, Sie mit einem der im Inneren des U-förmigen Rahmen liegenden Akteure zu verbinden. Diese Akteure liegen in einem Aktionsfeld des Betriebssystems, denn sie liegen innerhalb der Kontur eines Rundknotens. Das bedeutet, dass diese Akteure anfänglich noch gar nicht agieren können, sondern als

totes Material betrachtet werden müssen. Denken Sie an die Bibelstelle im Alten Testament, wo es heißt: „Er formte einen Menschen aus Lehm und hauchte ihm den Odem ein.“ Solange der Odem noch nicht eingehaucht wurde, stellt der Akteur nur einen Klumpen totes Material dar, der zwar Speicherplatz belegt, aber noch nicht agieren kann. Wenn er dann später agieren wird, benötigt er immer noch den gleichen Speicherplatz wie vorher, aber er ist nun etwas Lebendiges geworden. In Analogie hierzu sehen wir diese Akteure zuerst einmal als speicherbelegendes Material, dem über das Betriebssystem ein Odem eingehaucht werden kann. So können Sie dem Betriebssystem beispielsweise sagen: Ich würde gerne mit einem Browser kommunizieren. Daraufhin wird das Betriebssystem diesem Browser den Odem einhauchen, und danach wird sich dieser Browser über ein Bildschirmfenster bei Ihnen melden. Sie können nacheinander mehrere Akteure aus diesem Aktionsfeld des Betriebssystems aktivieren lassen, und zu jedem aktivierten Akteur wird ein Bildschirmfenster auf dem Monitor zu sehen sein.

Ihr Arbeitsplatzrechner alleine genügt aber für die Durchführung Ihrer Übungen nicht. Die Aufgabentexte, also sowohl die vorgegebenen Programmsourcen als auch die aufgabenbeschreibenden Texte und Bilder liegen im Dateispeicher des zentralen Server-Rechners *wendtstud1*, der für alle Übungsgruppen nur einmal vorhanden ist, wogegen jede Übungsgruppe ihren eigenen Arbeitsplatzrechner hat. Wenn Sie also die Aufgabentexte anschauen wollen, dann müssen Sie in Ihrem Client einen Browser aktivieren und diesem Browser sagen, welche Datei er aus dem Server zur Anzeige bringen soll. In der Struktur des Bildes 11 genügt es aber nicht, dass der Browser Ihnen nur die Datei anzeigt, sondern Sie müssen auch verlangen, dass diese Datei im Dateispeicher des Client abgelegt wird, damit sie anschließend die Akteure *Editor*, *Compiler* und *Exec* auf den Clientdateien arbeiten lassen können. Mit dem Editor können Sie die Programmtexte verändern. Mit dem Compiler können Sie die Programmtexte übersetzen lassen, wobei die ursprünglich gegebene Programmdatei vom Compiler nur gelesen wird. Das Übersetzungsergebnis legt der Compiler in einer neuen Datei im Clientdateispeicher ab.

Damit Sie Dateien zwischen dem Client und dem Server hin und her transferieren können, müssen Sie explizit eine Verbindung zwischen dem Client und dem Server herstellen. Dies geschieht, indem Sie im Client einen sog. Shell-Client aktivieren und diesem sagen, er solle eine Verbindung zu einem bestimmten Shell-Server im Rechner wendtstud1 herstellen. Der für Sie zuständige Shell-Server ist nach Ihrer Übungsgruppe benannt, also beispielsweise kdp14. Für den Verbindungsaufbau fragt Sie der LINUX-Akteur im Serverrechner nach Ihrem Benutzernamen und Ihrem Passwort. Diese sind nicht die gleichen, die Sie beim Einloggen in den Clientrechner angegeben haben. Ihr Benutzername für die Server-Sitzung ist gleich Ihrer Übungsgruppenbezeichnung, also beispielsweise kdp14.

Nachdem die Datei mit dem ausführbaren Programmcode erzeugt ist, können Sie den Akteur Exec mit der Programmausführung beauftragen. Dann wird dieser Akteur all das tun, was Sie in Ihrem Programm verlangt haben. Er wird also auch über Ihre Ein- und Ausgabekanäle mit Ihnen kommunizieren, damit Sie die gewünschten Eingaben machen können und damit Ihnen die Ergebnisse mitgeteilt werden.

Die offizielle Abgabe einer Lösungsdatei als gemachte Übung erfolgt dadurch, dass Sie die Datei aus dem Dateispeicher Ihres Arbeitsplatzrechners in den Dateispeicher des Servers wendtstud1 an eine bestimmte Stelle des dortigen Dateienbaums kopieren.

In Bild 11 kommen neben dem gezeigten Menschen viele unterschiedliche softwaretechnische Akteure vor:

der Client-LINUX-Akteur,
der Server-LINUX-Akteur,
der Browser,
der Webserver,
der Editor,
der Compiler,
der Exec-Akteur,
der Shell-Client,
die Sitzungsassistenten im Server.

Nicht alle, aber die meisten dieser Akteure kommunizieren direkt mit dem Menschen und haben dabei jeweils eine eigene Art und Weise, mit dem Benutzer zu kommunizieren. Dies bedeutet anschaulich, dass der Benutzer viele individuelle technische „Fremdsprachen“ lernen muss.

Ein Clientbetriebssystem, welches unmittelbaren Zugang zur Kommunikationsperipherie des Benutzers hat, kann eine komfortablere Benutzerschnittstelle anbieten als ein Betriebssystem, welches nur über einen Shell-Client hindurch mit dem Benutzer kommunizieren kann. Das Clientbetriebssystem kann graphische Benutzerelemente anbieten. Dies bedeutet, dass etliche Identifikationen durch Zeigen realisiert werden können. Alle sogenannten Icons, die das Betriebssystem auf den Bildschirm spielt, sind zeigbare Einheiten. Durch Anklicken eines solchen Icons spricht man nicht den mit dem Icon gemeinten Akteur an, denn dieser Akteur ist noch gar nicht aktiviert und kann deshalb den Anstoß gar nicht wahrnehmen. Durch das Anklicken eines Icons teilt man lediglich dem Betriebssystem mit, dass es den durch das Icon symbolisierten Akteur aktivieren soll.

Die Kommunikation mit dem Serverbetriebssystem geht in unserem Falle nur alphanumerisch. Das bedeutet, dass man bestimmte Kommandokürzel benutzen muss und dass man nicht den Mauszeiger zum Identifizieren irgendwelcher Individuen benutzen kann. Man muss zum Verschieben des Cursors die Cursorbewegungstasten der Tastatur benutzen.

Identifikation von Dateien

Bei der Kommunikation mit Akteuren im Rechner muss man immer wieder bestimmte Dateien identifizieren, die in den jeweiligen Dateispeichern liegen. Es ist heute allgemein üblich, diese Dateien über einen sogenannten Dateienbaum identifizierbar zu machen. Bild 13 zeigt ein Beispiel eines solchen Baumes. Die einzelnen Dateien bilden die Blätter in diesem Baum. Um ein solches Blatt zu identifizieren, muss man ausgehend von der Wurzel jeweils angeben, wie man absteigen soll. Die im Bild 13 schattiert gezeigte Datei wird also dadurch identifiziert, dass man den Pfad, der von der Wurzel ausgehend zum Blatt führt, als Folge von Richtungsentscheidungen beschreibt, wie es im Bild gezeigt ist.

Da die Bäume im Allgemeinen wegen der großen Anzahl von Dateien eine verhältnismäßig große Tiefe bekommen können, werden die Identifikationsausdrücke verhältnismäßig lang, wenn man jedesmal den ganzen Weg ausgehend von der Wurzel beschreiben muss. Glückli-

cherweise arbeitet man meistens in einem bestimmten Kontext, bei dem man nur Dateien eines bestimmten Teilbaums identifizieren muss. Deshalb gibt es in diesen Systemen immer eine Speicherzelle, worin man die Identifikation des Knotens ablegen kann, der für die folgenden Dateiidentifikationen als Wurzelknoten betrachtet werden soll. In Bild 13 könnte beispielsweise die schattierte Teilbaumwurzel als Bezugsknoten angegeben worden sein; dann genügt zur Identifikation der schattierten Datei die Angabe des kürzeren Pfades, der vom Referenzknoten zum Blatt führt.

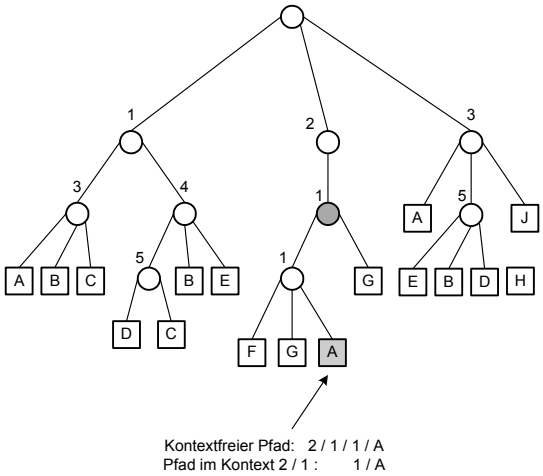

Bild 13 Beispiel eines Dateienbaums

****

Notwendigkeit der Berücksichtigung der Programmleser

Meine Beobachtungen aus dem Übungsbetrieb geben Anlass, Ihnen noch einmal eindringlich den Unterschied zwischen einem Softwareingenieur und einem Programmierkünstler klar zu machen. Ihre jeweiligen Übungsaufgaben bestehen darin, dass Sie ein vorgegebenes Programm in ganz bestimmter Weise abändern sollen. Ich habe nun festgestellt, dass viele von Ihnen sich bei diesen Abänderungen nicht auf das geforderte Minimum beschränken, sondern die Funktionalität der Programme erweitern. Ich kann ja verstehen, dass Sie mit der vorgegebenen Funktionalität unzufrieden sind und versuchen, das Programm für die Anwender attraktiver zu machen. Aber im vorgegebenen Falle schreiben Sie diese Programme weder für sich selbst noch für irgendwelche Anwender. Sie schreiben sie für mich als den Leser Ihrer Programme. Sie selbst werden später als Softwareingenieure auch in der Rolle von Programmlesern sein und viel weniger in der Rolle von Programmschreibern. Was ich mir selbstverständlich wünsche ist, dass alle Übungsteilnehmer möglichst gleiche Ergebnisprogramme abliefern, und das kann nur dadurch erreicht werden, dass sich jeder auf das geforderte Minimum beschränkt. Es ist auch nicht Ihre Aufgabe, Variablenbezeichner zu ändern, auch wenn Sie berechtigterweise der Überzeugung sein können, dass es viel anschaulichere und leichter verständliche Bezeichner gibt. Wenn aber jeder von Ihnen eine individuelle Bezeichnerwahl vornimmt und außerdem noch eine individuelle Funktionserweiterung, dann brauche ich als Programmleser nicht einige Stunden, sondern einige Wochen zur Durchsicht aller abgelieferten Programme. Wir wollen Sie hier nicht zu Programmierkünstlern ausbilden, denn Programmierkünstler gibt es weltweit inzwischen mehrere Millionen. Wir wollen Sie zu Softwarein-

genieuren machen, und die können ihre Arbeit nur dann erfolgreich durchführen, wenn sie streng die Einhaltung von Spezifikationen überwachen.

Die erste Übungsaufgabe als Beispiel eines Algorithmus

Die bisher gemachten allgemeinen Aussagen werden nun veranschaulicht durch Betrachtung der konkreten ersten Übungsaufgabe. In dieser Übungsaufgabe wird Ihnen ein Programm vorgegeben, welches der Berechnung einer Näherungslösung der Quadratwurzel aus einem einzulesenden Argument dient. Dieses Programm sollen Sie so abändern, dass es nicht die Quadratwurzel sondern die Kubikwurzel berechnet. Sowohl der Algorithmus zur Bestimmung einer Näherungslösung für die Quadratwurzel als auch der Algorithmus zur Bestimmung einer Näherungslösung für die Kubikwurzel wird Ihnen vorgegeben. Sie müssen also keinen Algorithmus erfinden. Ich zeige Ihnen aber nun, welche Überlegungen zu diesen Algorithmen geführt haben.

Seit langem ist in der Mathematik das sogenannte Newtonsche Näherungsverfahren zur Bestimmung einer Nullstelle einer stetig differenzierbaren, monotonen und wendepunktfreien Funktion f(x) bekannt. Eine monotone Funktion f(x) ist dadurch charakterisiert, dass das Vorzeichen ihrer ersten Ableitung, also ihrer Steigung für alle Werte von x das gleiche ist. Entweder also steigt die Funktion mit steigenden x-Werten dauernd an, oder sie fällt mit wachsenden x-Werten dauernd ab. Eine Funktion f(x) ist wendepunktfrei, wenn das Vorzeichen ihrer zweiten Ableitung für alle x-Werte das gleiche ist. Das bedeutet, dass die Steigung einer wendepunktfreien Funktion mit wachsenden x-Werten entweder immer nur größer wird oder immer nur kleiner wird.
Das Newtonsche Näherungsverfahren besteht nun darin, dass man irgendeinen willkürlichen Wert für x wählt als den ersten Näherungswert für die gesuchte Nullstelle. An dieser x-Stelle bestimmt man die Tangente an die Funktionskurve, und der Schnittpunkt dieser Tangente mit der x-Achse ist der nächste Näherungswert, falls nicht schon der erste Näherungswert mit der gesuchten Nullstelle zusammen fiel. Dass die Folge der so bestimmten Näherungswerte zu der gesuchten Nullstelle hin konvergiert, ist eine Konsequenz der Monotonie und der Wendepunktfreiheit der betrachteten Funktion. Dies sieht man leicht ein, indem man die Zeichnung in Bild 14 betrachtet.

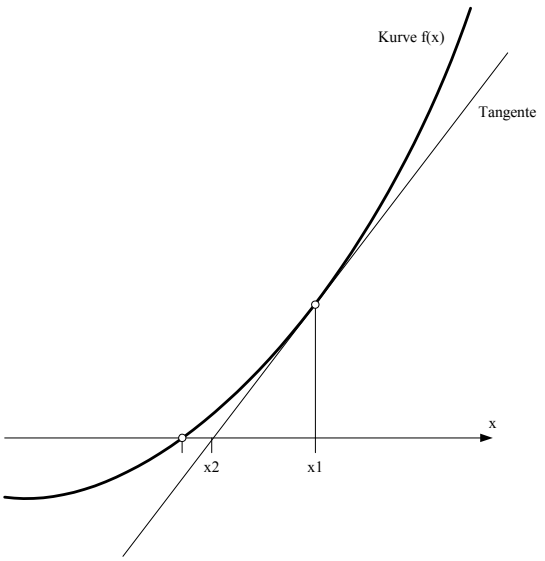

Bild 14 Veranschaulichung des Newtonschen Näherungsverfahrens

In unserem konkreten Falle, wo wir eine Quadratwurzel berechnen sollen, ist die Funktion, deren Nullstelle wir suchen müssen, gegeben als

$f(Näherungswert) = Näherungswert^2 - Radikand.$

An der Nullstelle dieser Funktion gilt

$Näherungswert^2 = Radikand$

d.h. die Nullstelle ist genau dort, wo der Näherungswert die Quadratwurzel aus dem Radikanden ist.

Falls wir bei der Wahl des ersten Näherungswertes nicht schon zufällig die exakte Nullstelle getroffen haben, kann keiner der folgenden Näherungswerte exakt auf die Nullstelle fallen. Die Folge der Näherungswerte führt lediglich immer näher an die Nullstelle heran. Man muss also ein Kriterium festsetzen für die Entscheidung, nach welchem Näherungsschritt man die Berechnung abbrechen will. Allgemein wird man sagen, dass man die Schrittfolge abbricht, wenn man mit der erreichten Näherung zufrieden ist. Zufriedenheit könnte beispielsweise gegeben sein, wenn der erreichte Näherungswert in den ersten 5 Dezimalstellen bereits mit der Nullstelle übereinstimmt.

Wie das hier beschriebene Verfahren im konkreten Falle in Formeln der analytischen Geometrie umzusetzen ist, wird als Teil der Aufgabenstellung zur ersten Übungsaufgabe mitgeliefert.

Bild 15 zeigt den auszuführenden Algorithmus in Form eines Petrinetzes. Ganz zu Beginn des Algorithmus werden drei Konstante festgelegt, die wir auch hätten einlesen können; der Einfachheit halber wurde hier aber auf die Möglichkeit der Eingabe verzichtet. Die erste Konstante gibt die Untergrenze für den Radikanden an, die nicht unterschritten werden soll. Diese Untergrenze muss ein kleiner positiver Wert sein. Die zweite Konstante dient der Bestimmung des Abbruchkriteriums bezüglich der erreichten Genauigkeit des Näherungswertes. Die dritte Konstante gibt an, wie viele Näherungsschritte maximal ausgeführt werden sollen. Die Berechnung soll nämlich auch dann abgebrochen werden, wenn nach dieser maximalen Schrittzahl die gewünschte Genauigkeit immer noch nicht erreicht ist.

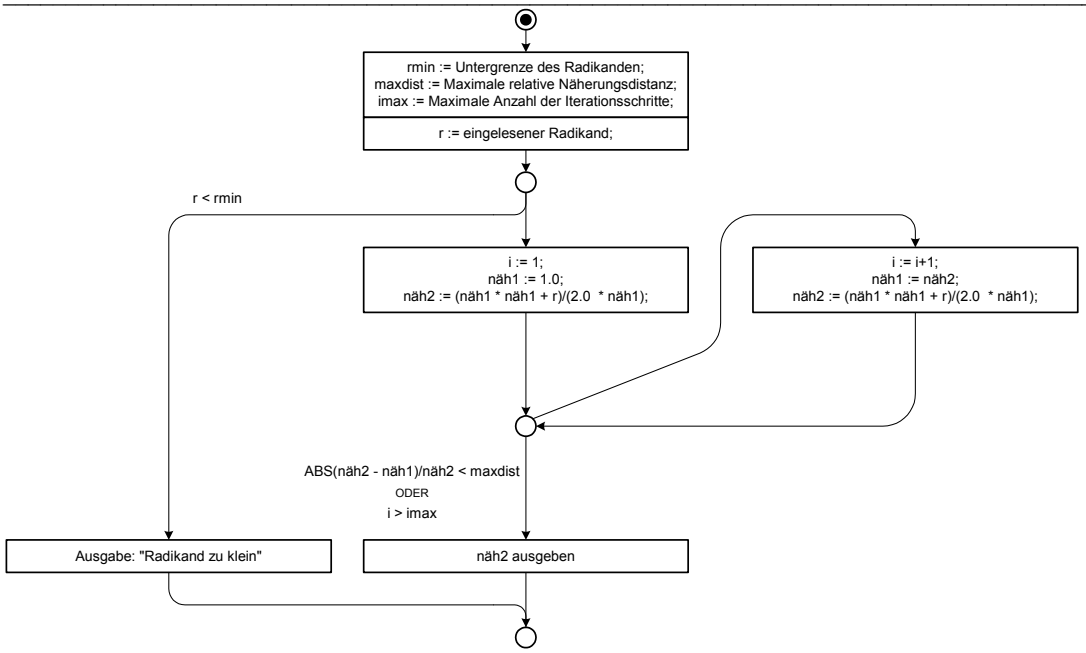

Bild 15 Petrinetz-Darstellung des Algorithmus der Quadratwurzelberechnung