Reengineering–Werkzeuge für Webseiten

Johannes Martin
University of Victoria
Department of Computer Science
PO Box 3055, Victoria, BC V8W 3P6
Kanada
jmartin@csr.csc.uvic.ca

Ludger Martin
Technische Universität Darmstadt
Fachbereich Informatik
Wilhelminenstr. 7, 64283 Darmstadt
Deutschland
lumartin@gkec.informatik.tu–darmstadt.de

Abstrakt

Anbieter von Informationen im Internet müssen ihren Datenbestand ständig analysieren und aktualisieren. Dieses Papier erklärt Gemeinsamkeiten in der Wartung von Software auf der einen Seite und Informationsangeboten auf der anderen Seite und erläutert inwiefern Software–Engineering–Werkzeuge zur Wartung von Informationsbeständen geeignet sind.

1. Einleitung

Das Internet, wie wir es in seiner heutigen Form kennen, ist nun einige Jahre alt. Webangebote beinhalten eine immer größer werdende Menge von Informationen. Eine Herausforderung für Webanbieter ist die kontinuierliche Bestandsaufnahme und Wartung ihres Informationsangebotes. Um bestehende Kunden zu halten und neue Kunden zu gewinnen, müssen die Daten sowohl in übersichtlicher Weise organisiert und gut erreichbar sein als auch laufend aktualisiert und erweitert werden.

Softwareingenieure kämpfen schon seit langer Zeit mit ähnlichen Problemen: Informationen über Softwaredesign und -Architektur sowie über Quellcode müssen bis ins Detail erfaßt, verarbeitet und dargestellt werden. Viele Software–Engineering–Werkzeuge nutzen Graphen für die Realisation dieser Aufgaben.

Webseiten und ihre Verknüpfungen können in trivialer Weise als Graphen aufgefaßt werden. Nachfolgend präsentieren wir Ansätze zur Analyse von Webangeboten mittels des Software–Engineering–Werkzeugs *Rigi* [5, 10].

2. Überblick über Web–Analysewerkzeuge

Es gibt eine ganze Reihe von Werkzeugen zur Erstellung, Analyse und Wartung von Webseiten. Die meisten allgemein verfügbaren Werkzeuge befassen sich jedoch mit der Erstellung von neuen Inhalten und nicht mit der Wartung und Strukturierung bestehender Angebote. Analysewerkzeuge haben häufig nur begrenzte Funktionalität:

- Webseiten–Prüfer verifizieren die Korrektheit von HTML–Seiten und ihrer Verbindungen zu anderen Webseiten.
- Webstatistik–Werkzeuge analysieren Webserver–Logbücher und erzeugen Statistiken über die Nutzung von Webseiten.

Im Bereich der Forschung wurden einige Versuche gemacht, diese Probleme anzugehen, insbesondere durch die Darstellung von Webangeboten als Graphen. Chung und Lee [2] schlagen den Einsatz des Unified Process [3] und der Unified Modelling Language (UML [1]) für die Wartung von Webangeboten vor. Da die aktuelle Struktur von Webangeboten oft schlecht oder gar nicht dokumentiert ist, verwenden sie Reverse–Engineering Techniken, um die Struktur und Navigationsschemen eines Webangebots zu ermitteln. Ihr Werkzeug zeigt zwei verschiedene Ansichten von Webangeboten, eine, die die Verknüpfungen zwischen Webseiten als Abhängigkeiten zwischen Komponenten darstellt, eine andere, die das Webangebot anhand der Verzeichnisstruktur gliedert. Chung und Lee erwarten vom Einsatz ihres Werkzeuges ein besseres Verständnis des aktuellen Webangebotes, Hilfe bei der Qualitätsverbesserung und ein kleineres Risiko bei der Erneuerung des Webangebotes.

Ricca und Tonella [7, 8] entwickelten ihr Werkzeug *ReWeb* zur Analyse der Struktur und Entstehungsgeschichte von Webangeboten. ReWeb speichert über einen gewissen Zeitraum mehrere Versionen eines Webangebotes und nutzt Graphen, um die Unterschiede (Hinzufügungen, Änderungen und Löschungen von Seiten oder Verknüpfungen) aufzuzeigen. Verschiedene Farben werden verwendet, um die Unterschiede darzustellen. ReWeb benutzt nicht UML sondern eine eigene Graphendarstellung. Ricca und Tonella erörtern auch die Auswirkung von *Rahmen* (frames) auf Webangebote und deren Struktur, sowie *Dominatoren*, *kürzeste Pfade* und *engverbundene Komponenten*. Wir nutzten einige ihrer Ideen in unseren Fallstudien.

Knotentyp	Beschreibung
Host	ein Webserver.
HTML	eine HTML–Webseite.
Forward	eine Webseite, die automatisch an eine andere weiterleitet.
Text	eine Webseite mit Textinhalt (nicht HTML).
Image	eine Graphikdatei.
Other	eine Webseite mit unbekanntem oder anderem Inhalt.
Email	ein `mailto:` URL.
Applet	ein von einer HTML–Webseite aufgerufenes Java Applet.
CGI	ein aus einem HTML–Formular aufgerufenes CGI Skript.
MalformedURL	ein syntaktisch inkorrektes URL.
DeadURL	ein URL, das zur Zeit nicht erreichbar ist.
Client	ein Webbenutzer (genaugenommen der Computer, der die Verbindung zum Server aufbaut).

Kantentyp	Beschreibung
residesOn	verbindet eine Webseite mit dem anbietenden Server.
linksTo	repräsentiert einen HTML–Link zwischen zwei Webseiten.
runsApplet	verbindet ein Java–Applet mit der aufrufenden Webseite.
runsCGI	verbindet ein CGI–Skript mit der aufrufenden Webseite.
forwardsTo	verbindet eine Webseite mit automatischer Weiterleitung und ihr Ziel.
refersTo	zeigt an, daß einer Verknüpfung zwischen Webseiten gefolgt wurde (dynamische Analyse).
accessed	zeigt an, daß ein Webbenutzer eine Webseite anforderte (dynamische Analyse).

Attributtyp	Beschreibung
contentType	der MIME–Typ der Webseite.
nodeurl	das URL der Webseite.
usesScript	gibt an, ob eine Webseite Skripte enthält (z.Bsp. Java–Skript).
error	erläuternde Fehlermeldung für syntaktisch inkorrekte oder unerreichbare URLs.
parseError	erläuternde Fehlermeldung für syntaktisch inkorrekte HTML–Seiten.
referenced	gibt an, wie oft ein URL angefordert wurde (dynamische Analyse).
accessCount	gibt an, wie oft *ein* Benutzer ein URL anforderte (dynamische Analyse)
referralCount	gibt an, wie oft einer Verknüpfung zwischen zwei URLs gefolgt wurde (dynamische Analyse)

Abbildung 1. Definition der Graphklasse

3. Rigi als Web–Analysewerkzeug

Das Software Reverse–Engineering–Werkzeug Rigi wurde von Anfang an mit dem Ziel konzipiert, die Anpassung an neue Aufgaben zu ermöglichen [9]. Schon frühzeitig wurde diese Flexibilität genutzt, um Rigi auch in Bereichen außerhalb der Softwareentwicklung einzusetzen. Sie wurde unter anderem durch folgende Eigenschaften Rigis erreicht:

- Graphkonzept: Rigi nutzt gerichtete Graphen zur Verwaltung und Darstellung der zu bearbeitenden Daten. Sowohl Knoten als auch Kanten haben Typen und können mit Attributen versehen werden. Diese Typen und Attribute sind frei wählbar, und können sogar während der Arbeit mit Rigi verändert werden.

- Trennung von Informationsgewinnung und Informationsauswertung: Parser extrahieren Artifakte aus den zu analysierenden Datenbeständen (normalerweise Software–Quelltexte); der Rigi Graph–Editor ermöglicht die Bearbeitung der Daten. Die Kommunikation geschieht über ein gemeinsames Dateiformat [4].

- Programmierbarkeit des Graph–Editors: Rigi kann nicht nur interaktiv bedient werden; viel benutzte oder komplexe Operationen können durch eine interpretierte Programmiersprache automatisiert werden. Die Programmiersprache erlaubt den Zugriff auf den gesamten Datenbestand und die Anpassung der Benutzungsoberfläche.

Um Rigi als Analysewerkzeug für Webangebote einzusetzen, mußte der Definitionsbereich (d.h. die Typen und Attribute der Knoten und Kanten) des Graphen festgelegt und ein Parser für Webseiten entwickelt werden. Die folgenden Abschnitte beschreiben dies.

3.1. Definitionsbereich der Graphen

Graphklassen in Rigi werden durch die Typen und Attribute ihrer Knoten und Kanten charakterisiert. Im Graphmodell für Webseiten werden URLs als Knoten und Verbindungen zwischen URLs als Kanten gezeigt. Abbildung 1 zeigt die Klassifizierung der Knoten, Kanten und ihrer Attribute, für die wir uns entschieden.

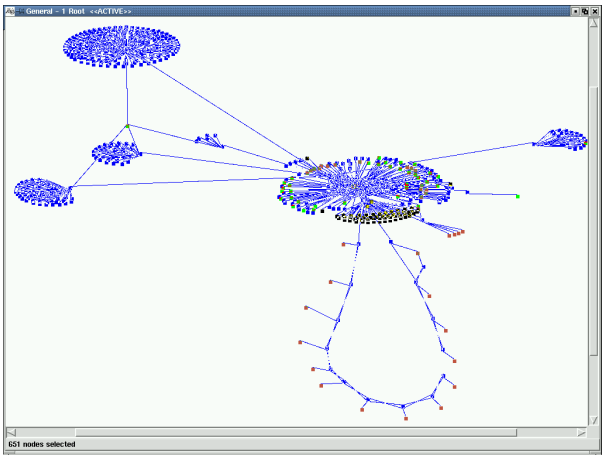

Abbildung 2. Rigi–Webseiten — Unbearbeitete Visualisierung

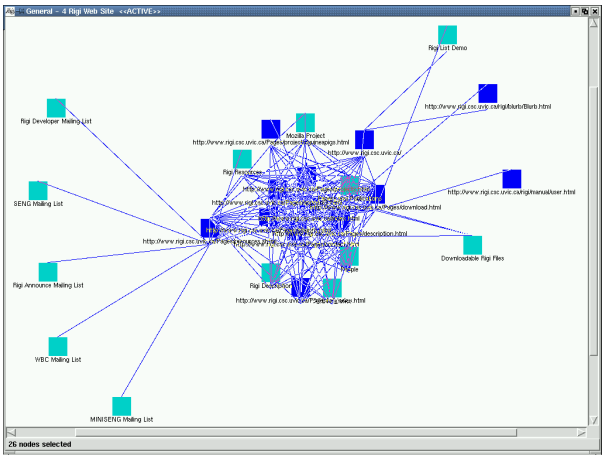

Abbildung 3. Rigi–Webseiten — Bearbeitete Visualisierung

3.2. Parser: Web2Rsf

Web2Rsf analysiert Webseiten beginnend mit einem anzugebenden Start–URL. Dazu bestimmt es zuerst den MIME-Typen eines URLs. Falls es sich um ein HTML–Dokument handelt, extrahiert es daraufhin alle Verknüpfungen zu anderen Webseiten, und untersucht diese dann in gleicher Weise wie das Start–URL.

Web2Rsf bietet mehrere Möglichkeiten, auf die Webseiten–Analyse Einfluß zu nehmen. Es kann bestimmt werden, wie tief Verknüpfungen zu anderen Dokumenten verfolgt werden sollen, z.B. kann erreicht werden daß nur direkt mit der Startseite verknüpfte Seiten analysiert werden. Des weiteren kann die Analyse auf URLs begrenzt werden, die mit einem bestimmten Namensmuster übereinstimmen oder auf einer Auswahl von Webservern gelagert werden.

Als Programmiersprache für Web2Rsf bot sich Java aufgrund seiner integrierten Unterstützung von URLs und Netzwerkprotokollen an. Da die Übertragungszeiten von Internetseiten manchmal länger sein können, kann Web2Rsf auf Wunsch mehrere Threads starten, um gleichzeitig mehrere Seiten zu bearbeiten.

4. Fallstudien

Die nachfolgenden Abschnitte zeigen Ergebnisse, die in der Analyse einiger Webangebote mit Rigi gewonnen wurden.

4.1. Rigi–Webseiten

Um den neuentwickelten Parser mit Rigi zu testen, begannen wir, die Rigi–Webseiten [6] zu analysieren. Der Parser war mühelos in der Lage, die ca. 550 auf dem Server gelagerten Seiten zu untersuchen. Wir luden den erzeugten Graphen in den Rigi–Grapheditor und benutzten zuerst Knotenfilter, um Verknüpfungen zu nicht erreichbaren Webangeboten zu finden. Die Fluktuation der Angebote im Internet führt immer wieder dazu, daß Angebote von der Bildfläche verschwinden. Daher ist es wichtig, ein eigenes Angebot auf nicht mehr aktuelle Verknüpfungen zu untersuchen und entsprechend zu aktualisieren. Schon allein zu diesem Zweck lohnte sich die Entwicklung von Web2Rsf und der Einsatz von Rigi.

Daraufhin nutzten wir Rigis Spring–Layout Algorithmus, um einen besseren Überblick über die Struktur der Rigi–Webseiten zu gewinnen. Das Ergebnis ist in Abbildung 2 zu sehen. Es überraschte auf den ersten Blick nicht nur durch seine ästhetische Schönheit, sondern auch durch die klar sichtbaren Gruppierungen von Webseiten. Ein genauerer Blick auf die einzelnen Glieder dieser Gruppierungen erklärte ihren Ursprung: auf dem Rigi–Webserver werden einige Archive gelagert, z.B. von Mailing–Listen und Publikationen zu Rigi. Die halbmondartige Kette im unteren Teil der Abbildung stellt eine Online–Präsentation dar: aufeinanderfolgende Seiten sind miteinander verbunden, aber nur am Anfang und am Ende der Präsentation besteht eine Verbindung zum Rest der Webseiten.

Wir benutzten den Grapheditor, um diese Gruppierungen zusammenzulegen und so den Graphen zu vereinfachen. Außerdem identifizierten wir alle Knoten, die Webseiten auf externen Webservern repräsentieren, und legten sie in eine eigene Gruppe, da sie für unsere erste Betrachtung unerheblich waren. Des weiteren verbargen wir alle Kanten, die Bilddateien auf dem Webserver entsprechen. Das Ergebnis dieser Bearbeitung ist in Abbildung 3 zu sehen. Die Gruppierungen, die in Abbildung 2 zu sehen waren, sind nun zu einzelnen Knoten geworden, die am Rand der Abbildung stehen. Im Zentrum ist ein dichter Haufen von eng verbundenen Webseiten.

Ein Blick auf die Rigi Webseiten erklärt diese Struktur. Ein Großteil der Rigi Webseiten setzt ein einheitliches Navigationsmenü ein, um den raschen Wechsel von einem

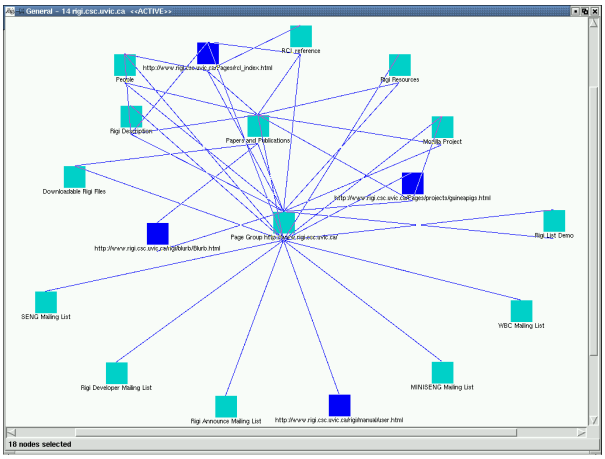

Abbildung 4. Rigi–Webseiten — Nach Menü–Analyse

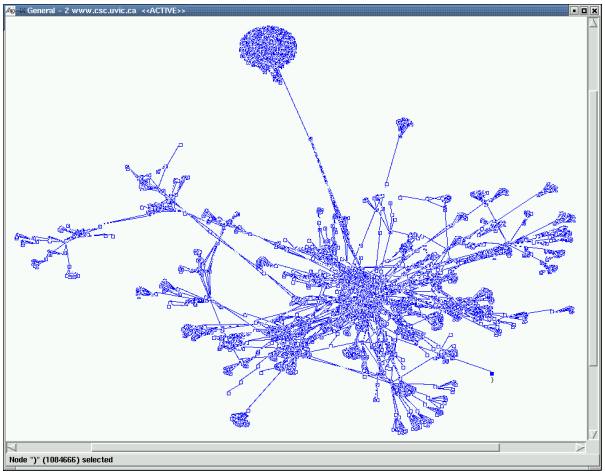

Abbildung 5. Fachbereichswebseiten — Überblick

Themengebiet in ein anderes zu ermöglichen. Daher sind alle diese Seiten eng miteinander verbunden. Andere Teile, wie zum Beispiel die zuerst identifizierten Archive, nutzen dieses Navigationsmenü nicht — aus Erfahrung mit der Wartung des Rigi Webangebotes wissen wir, daß diese bei der Neukonzeption der Webseiten vor zwei Jahren unverändert vom alten Webangebot übernommen wurden. Um das einheitliche Erscheinungsbild des Webangebotes zu wahren, hätten diese Seiten in die Neukonzeption einbezogen werden sollen.

Da diese engverbundenen Webseiten (Ricca und Tonella [8] nennen sie *strongly connected components*) eine Einheit bilden, sollten sie auch als Einheit im Graph dargestellt werden. Wir entwickelten einen Algorithmus, der nach Häufungen von engverbunden Seiten sucht und aus ihnen Komponenten bildet. Nach Anwendung dieses Algorithmus' auf den Graphen erhielten wir den Graph in Abbildung 4.

4.2. Fachbereichswebseiten

Für unsere zweite Fallstudie wählten wir ein größeres Webangebot, nämlich das des Computer Science Fachbereichs der University of Victoria. Der Webserver bietet allgemeine Informationen zum Fachbereich, Informationen zu Vorlesungen (zum Beispiel Skripte und Übungen), und beheimatet auch Webseiten von Studenten. Web2Rsf analysierte alle Webseiten dieses Webangebots, die von der Startseite des Fachbereichs erreichbar waren, insgesamt ungefähr 3500, davon ungefähr 1700 HTML–Seiten. Wiederum zeigte der Parser keine Probleme bei der Analyse.

Die Probleme begannen, als wir den Graphen in den Rigi Graph–Editor luden: wie in der ersten Fallstudie versuchten wir mit Hilfe des Spring–Layouts einen Überblick über den Graphen zu gewinnen. Da das Spring–Layout aber eine Komplexität von $O(n^3)$ hat, hatte der Größenunterschied der Graphen eine sehr merkbare Auswirkung: die Berechnung des Layouts in Abbildung 5 dauerte mehrere Minuten. Außerdem stellt die Struktur, die aus dem Graphen ersichtlich wird, nur die Verknüpfungen der Webseiten dar; sie ist nicht die logische Struktur des Angebotes: eine genauere Betrachtung des Graphen zeigte, daß die gut erkennbaren Gruppierungen von Knoten an den Seiten des Graphen oft lokale Kopien von Handbücher im HTML–Format oder teilweise auch Webseiten von Studenten sind.

Ein anderer Weg, das Problem anzugehen, scheint in diesem Fall vernünftiger. Sowohl Chung und Lee [2] als auch Ricca und Tonella [8] schlagen vor, Verzeichnisnamen in URLs als Indikatoren der logischen Struktur eines Webangebotes zu verwenden. Dies erscheint insofern plausibel, da Webentwickler die Verzeichnisstruktur oft als Mittel zur Organisation von Inhalten verwenden. Daher entwickelten und implementierten wir Algorithmen in Rigi, um den Graphen anhand der Verzeichnisstruktur der URLs zu gliedern. Wir nahmen auch unser Wissen um Namenskonventionen auf dem Webangebot zur Hilfe, um Fachbereichsinformationen von Vorlesungsinformationen und Studentenwebseiten zu trennen. Das Ergebnis der Bearbeitung des Graphen ist in Abbildung 6 zu sehen.

Da Rigi es ermöglicht, Fragen über die Struktur des Graphen zu stellen, kann ein Webangebot-Verwalter Rigi zum Beispiel verwenden, um die Einhaltung von Stilrichtlinien des Fachbereichs auf Vorlesungs- und Studentenwebseiten zu prüfen. Zum Beispiel könnte er kontrollieren, ob alle Vorlesungswebseiten eine Verknüpfung zur Startseite des Fachbereichs haben, und Seiten finden, die diese Anforderung nicht erfüllen.

5. Dynamische Analyse

Ein Webangebot kann noch so gut organisiert und aktuell sein, es besteht doch immer die Gefahr, daß das Design den Anforderungen seiner häufigsten Nutzer nicht genügt. Daher muß ein Webentwickler auch kontrollieren, wie die

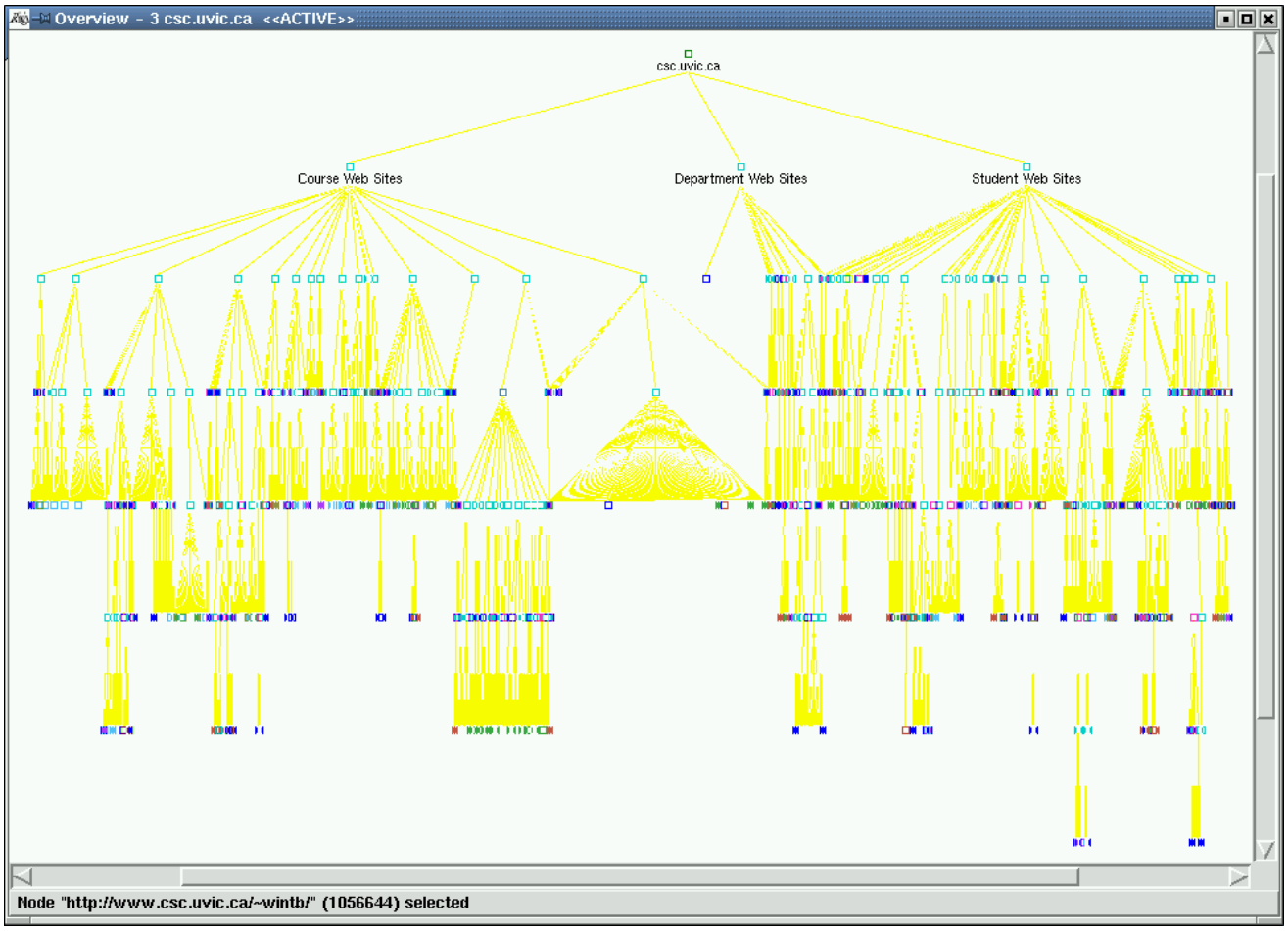

Abbildung 6. Fachbereichswebseiten — Verzeichnisstruktur

Besucher das Angebot nutzen, d.h. welche Seiten sie ansehen, und auf welchen Wegen sie zu diesen Seiten gelangen.

Web2Rsf kann Webserver–Logdateien untersuchen und sammelt dabei Informationen über die Nutzung des Webangebotes. Zu jeder Anfrage ermittelt es die Seite, die angefordert wurde, den Anforderer, und die Webseite, die die Verknüpfung zu dieser Seite bereitstellte. Web2Rsf erläutert und erweitert den zuvor bei der Untersuchung der Webseiten erstellten Graphen mit diesen Informationen.

Rigis Filter können dann genutzt werden, um zum Beispiel nur Webseiten zu zeigen, die mehr oder weniger oft als ein Schwellenwert besucht wurden, oder um die Pfade zu zeigen, über die eine Webseite am meisten erreicht wird. Der Webentwickler kann diese Informationen nutzen, um Webseiten zu ordnen und oft angeforderte Seiten besser erreichbar zu machen, oder den Besuchern Hilfen oder zusätzliche Informationen anzubieten. Da Web2Rsf diese Information nach Anforderern getrennt sammelt, können Besuchsverhalten verschiedener Nutzergruppen ausgewertet werden.

6. Zusammenfassung und Ausblick

Unsere Fallstudien zeigen, daß die Visualisierung von Webangeboten mittels Graphen möglich und nützlich ist. Rigis Anpassungsfähigkeit an neue Anforderungen durch die Definition von neuen Graphklassen und die Möglichkeit, Aufgaben durch seine Programmiersprache zu automatisieren, machten es recht einfach, Webangebote zu untersuchen.

Obwohl der Parser mit der Absicht entwickelt wurde, eine weite Palette von Webangeboten untersuchen zu können, besteht hier noch Bedarf zur Verbesserung. Zur Zeit versteht der Parser nur HTML 3.2 — zur Bearbeitung vieler moderner Webseiten sollte der Parser jedoch auch Rahmen, XML, JavaScript, und HTML 4.0 verarbeiten können. Die Wahl von Java als Programmiersprache für den Parser muß möglicherweise revidiert werden: die Entwicklung des Parsers wurde zwar durch Javas hochentwickelte Funktionsbibliotheken sehr erleichtert, jedoch scheitert der Parser manchmal an Hauptspeichergrenzen, wenn größere Webangebote untersucht werden sollen.

In unseren bisherigen Fallstudien haben wir die dynamische Analyse von Webangeboten noch nicht in Betracht genommen. Wir haben vor, einen Satz von Algorithmen zu

erarbeiten, der die Automatisierung der dynamischen Analyse ähnlich wie die der statischen Analyse ermöglicht. Eine weitere Fallstudie wird sich mit der Analyse der Webangebote einer ganzen Universität beschäftigen (d.h. aller Server in einer Domäne).

Literatur

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modelling Language User Guide*. Addison Wesley Longman, Inc, Reading, MA, 1999.

[2] S. Chung and Y.-S. Lee. Reverse Software Engineering with UML for Web Site Maintenance. In *Proceedings of WISE 2000, First International Conference on Web Information System Engineering*, pages 157–161, Hong Kong, China, June 2000.

[3] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley Longman, Inc, Reading, MA, 1999.

[4] J. Martin. RSF file format. Technical report, University of Victoria, Aug. 1999. http://www.rigi.csc.uvic.ca/list-archives/rigi-developer-archive/2000-02-10-15:26:16-19165.

[5] H. A. Müller and K. Klashinsky. Rigi — A system for programming-in-the-large. In *Proceedings of the 10th International Conference on Software Engineering*, pages 80–86, 1988.

[6] U. of Victoria. Rigi Web Server. http://www.rigi.csc.uvic.ca (April 2001).

[7] F. Ricca and P. Tonella. Visualization of Web Site History. In *Proceedings of WSE 2000, International Workshop on Web Site Evolution*, pages 30–33, Zürich, Switzerland, Mar. 2000.

[8] F. Ricca and P. Tonella. Web Site Analysis: Structure and Evolution. In *Proceedings of ICSM 2000, International Conference on Software Maintenance*, pages 76–86, San Jose, CA, Oct. 2000.

[9] S. R. Tilley. *Domain-retargetable reverse engineering*. PhD thesis, Department of Computer Science, University of Victoria, 1995. http://www.rigi.csc.uvic.ca /Pages/publications/dtre.pdf.

[10] K. Wong, S. R. Tilley, H. A. Müller, and M.-A. D. Storey. Programmable Reverse Engineering . *International Journal of Software Engineering and Knowledge Engineering*, 4(4):501–520, Dec. 1994. http://www.rigi.csc.uvic.ca /Pages/publications/progreveng.pdf.