Visualisierung von Softwaresystemen

Bei der Entwicklung grosser Softwaresysteme stehen Entwicklungsteam und Projektleitung vor dem Problem, den Überblick über den Programmcode und seine Struktur zu behalten. Besonders schwerwiegend tritt dieses Problem beim Reengineering oder der Weiterentwicklung und Wartung von Softwaresystemen auf, da hier oft Entwickler beteiligt sind, die das Softwaresystem nicht vom Anfang der Entwicklung her kennen. Als später hinzu stoßende Teammitglieder stehen sie vor der Aufgabe, in hunderttausenden von Zeilen von Code die Teile herauszufinden, die der Überarbeitung bedürfen. Ein einfaches Programmlisting wäre hier ein denkbar unübersichtliches und unhandliches Werkzeug.
Bei der Softwarevisualisierung versucht man, durch geschickte grafische Darstellung die Eigenschaften und Struktur von Softwaresystemen möglichst kompakt darzustellen, so daß der Überblick für den Entwickler erhalten bleibt und große Teile des Systems auf einmal darstellbar sind.
Indem man verschiedene grafische Abbildungen des Systems zu verschiedenen Zeitpunkten seiner Entwicklungsgeschichte als Animation zusammenfaßt, kann man außerdem einen direkten visuellen Eindruck von der Evolution des Systems erhalten.
Es werden im Folgenden zwei unterschiedliche Ansätze vorgestellt. Die Verfahren von G. Eick et al. dienen der Visualisierung von klassischem zeilenorientierten C-Code. Der Schwerpunkt liegt hier auf der geschickten Visualisierung großer Systeme. Die Evolution Matrix von Michele Lanza ist ein unabhängiger Ansatz der versucht , speziell die Entwicklungsgeschichte von objektorientierten, klassenbasierten Systemen darzustellen. Für sehr große Systeme oder die Darstellung einzelner Zustände des Systems ist dieser Ansatz nicht geeignet.

Konkrete Fragen, die sich etwa der Projektleiter bei seiner Arbeit stellt, sind beispielsweise:

- Welche Subsysteme meines Projektes sind besonders groß ?
- Wo ist momentan der Entwicklungsschwerpunkt, d.h. wo sind die meisten Entwickler tätig ?
- Werden im wesentlichen Fehler behoben oder wird das Programm erweitert ?
- Gibt es Subsysteme , in denen besonders viele Fehler auftreten ?
- Wann waren wesentliche Release Daten ?
- Welche Subsysteme wachsen besonders stark ?

Dieses Fragen versuchen G. Eick et al. dem Projektleiter zu beantworten, indem sie die Eigenschaften des Programms in verschiedenen Diagrammarten geeignet darstellen.
Zur Beschreibung der Programmeigenschaften werden dabei verschiedene zeilenorientierte Metriken benutzt.

Software Visualization

In nebenstehendem Beispiel könnte die benutzte Metrik beispielsweise die Programmlaufzeit der Zeile sein, die für jede Zeile durch den Farbcode dargestellt wird. Blaue Zeilen bedeuten geringe Laufzeit, rot symbolisiert eine hohe Laufzeit. Man beachte, das die Wahl der Farbskala erheblichen Einfluß auf die intuitive Verständlichkeit hat. Würde man die Bedeutung der Farben umdrehen , käme es wohl zu Mißverständnissen. Die meisten Betrachter assoziieren blau mit kalt und daher wenig sowie rot mit heiß und daher viel.

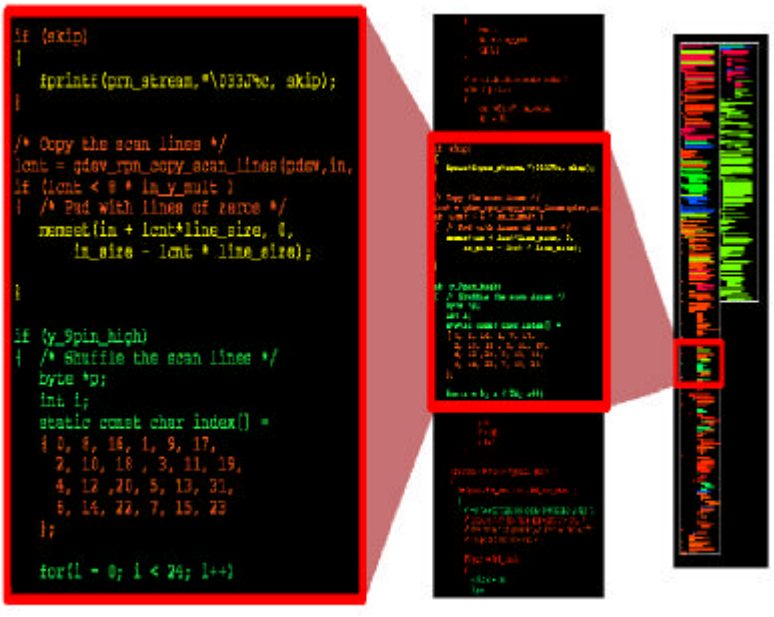

Abbildung 1

Dabei wird vom Programmtext zunehmend abstrahiert bis man im Diagramm ganz rechts die symbolische Zeilendarstellung nur noch die Strukturierung des Programmtextes und das Laufzeitverhalten erkennen läßt. Gewonnen hat man dafür Übersichtlichkeit, da im Vergleich zum Diagramm ganz links ein größerer Teil des Programms überblickt werden kann, ohne daß aus der Perspektive des Projektleiters wichtige Information verloren geht.
Je nach Auswahl der Metrik und Zusammenstellung der Diagramme erhalte ich Aussagen über verschiedene Aspekte meines Programmsystems:

- Entwicklungsgeschichte
- Versionsunterschiede
- Statische Eigenschaften des Codes
- Dynamische Eigenschaften des Codes, Profiling Ergebnisse

Abbildung 1 stellt z.B. die dynamischen Eigenschaften eines bestimmten Systemteils dar. Würde ich dieses Diagramm für verschiedene Zeitpunkte in der Entwicklungsgeschichte anfertigen und diese Abfolge von Diagrammen als Animation darstellen, könnte ich direkt den Erfolg der Tuningmassnahmen in der Entwicklung beobachten.
Weitere interessante Metriken sind:

- Modulgrössen
- Programmlaufzeit
- Zahl der Änderungen
- Zahl der Bugfixes
- Zahl der Programmierer, die Änderungen vornehmen
- Verschachtelungstiefe
- Fehlertyp

Bei der Visualisierung werden diese Metriken jeweils mittels Farbcodierung dargestellt. Durch die Wahl unterschiedlicher Diagrammtypen erhält man Übersichten mit unterschiedlicher Auflösung.
Das einfachste Diagramm ist die farbcodierte textuelle Darstellung des Programmcodes.
Mehr Information läßt sich gleichzeitig darstellen, indem einzelne Textzeilen nur schematisch durch eine farbcodierte Linie dargestellt werden. (s. Abbildung 1)

Im nächsten Schritt verwendet man für eine Zeile nur noch ein oder mehrere Pixel. (s. Abbildung 2)

Im letzten Schritt hebt man die proportionale Darstellung der Filegrößen auf und repräsentiert jedes File nur noch durch eine Box in einer von 4 Größen, je nachdem in welches Größen Quartil das File fällt. So erreicht man, das Files mit 10 Zeilen und Files mit 10000 Zeilen gleichzeitig auf dem Bildschirm sichtbar sind. Bei reiner Pixeldarstellung wäre das 10 Zeilen File so klein dargestellt, daß es unsichtbar wäre. Abbildung 3 zeigt eine solche zusammenfassende Darstellung.

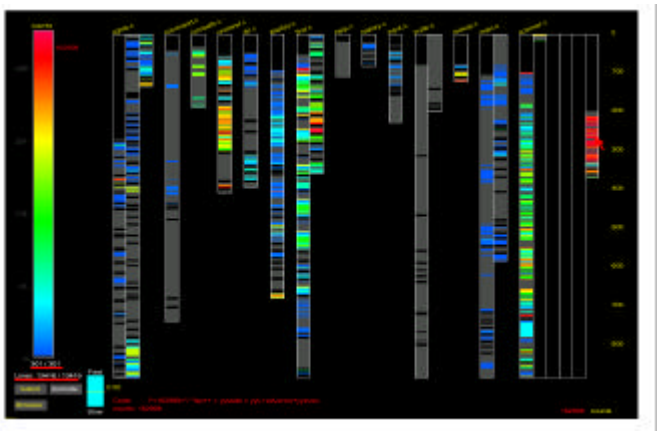

Abbildung 2 Darstellung von Profilerdaten - Laufzeit/Zeile

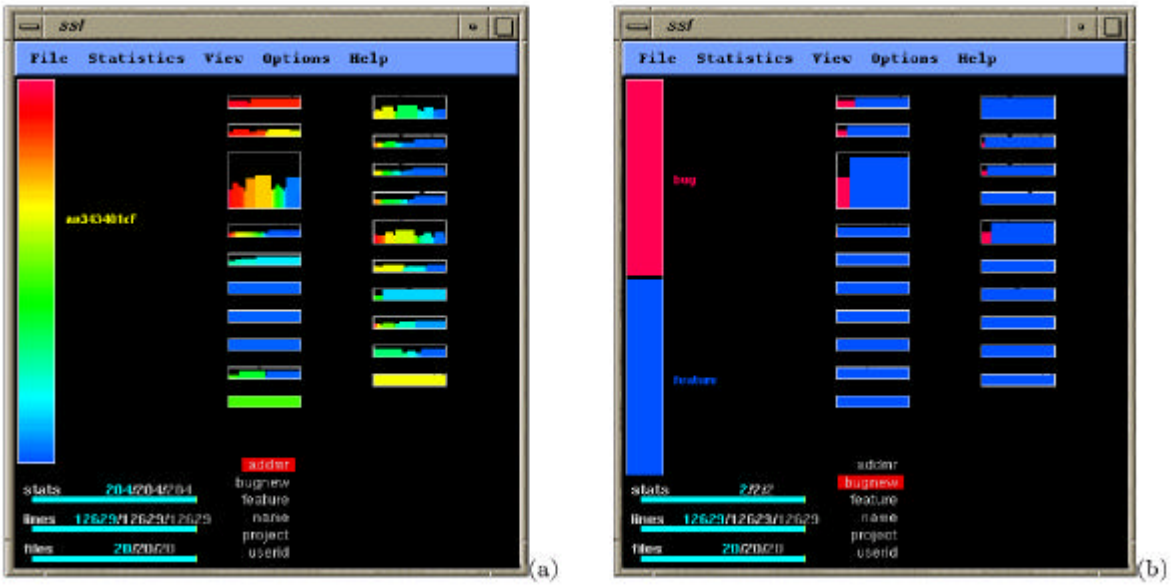

Abbildung 3 - Code Age und Bug / Feature Verteilung . Code Age einer Zeile ist die seit ihrer letzten Modifikation verstrichene Zeit. Das Diagramm b) zeigt die Anteile der Zeilen, die als Bugfix hinzugefügt wurden

Evolution Matrix

Bei der „Evolution Matrix“ genannten Visualisierung ist die Grundeinheit der Visualisierung eine Klasse. Jede Klasse wird durch eine Box dargestellt, die Größe der Box hängt von zwei beliebigen die Klassen beschreibenden Metriken ab. Im zugrundeliegenden Paper wurde Methodenzahl für die Breite und Zahl der Membervariablen für die Höhe benutzt.

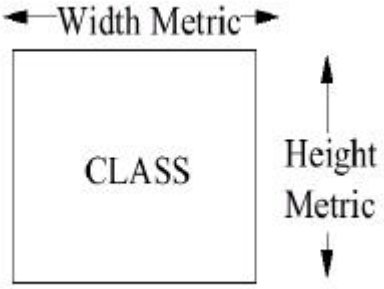

Abbildung 4 Darstellung einer Klasse

Die eigentliche „Evolution Matrix“ ist die Darstellung des Systemzustandes , d.h. der Klassen des Systems für verschiedene Systemversionen.
Anhand der Evolution Matrix kann man die

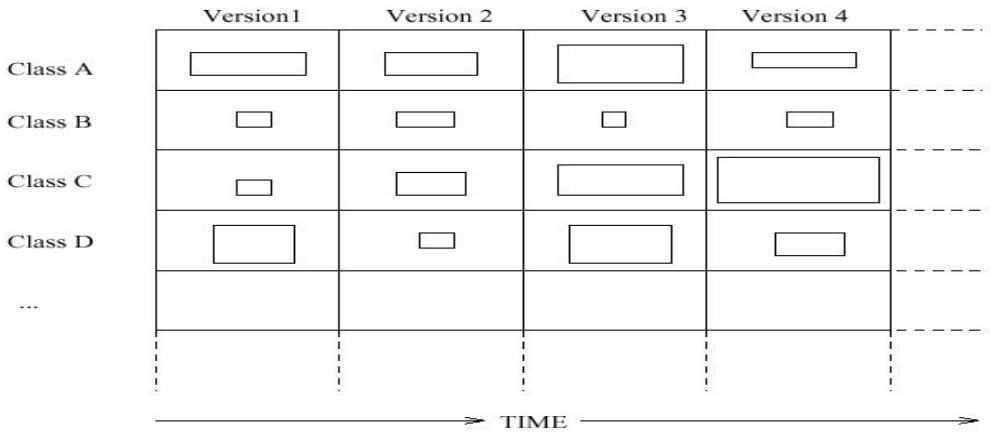

Abbildung 5 Evolution Matrix

Entwicklung einzelner Klassen und des Gesamtsystems erkennen.
Für das Gesamtsystem kann man erkennen, wie sich die Größe des Systems entwickelt hat, wann Klassen entfernt oder hinzugefügt wurden und wann die Hauptwachstumsphasen des Systems waren.

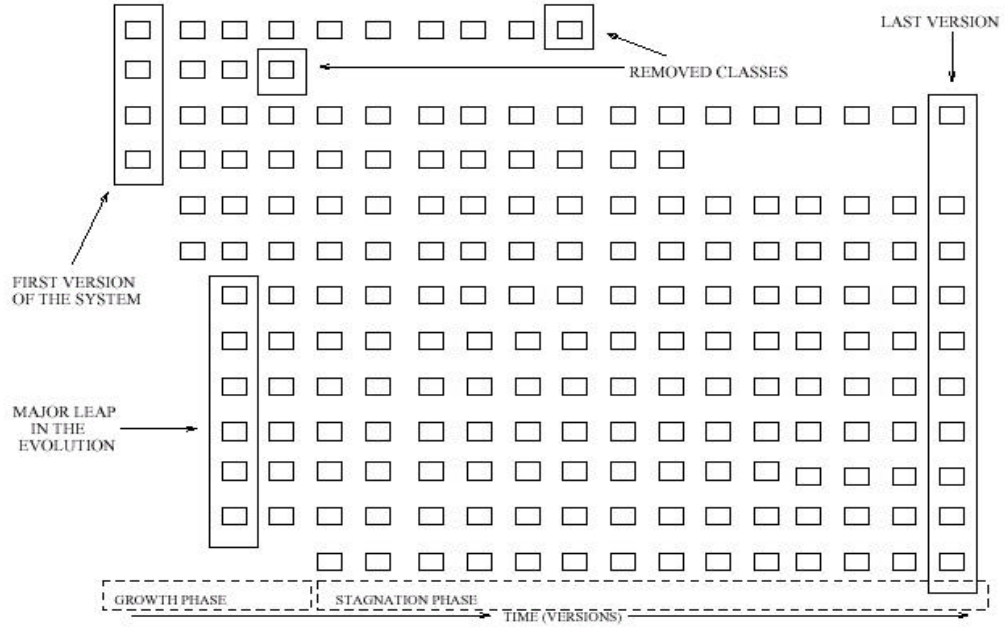

Abbildung 6 Evolution Matrix eines Systems

Aussagen über einzelne Klassen erhofft man sich von der Beobachtung und Klassifikation des Musters der Entwicklung dieser Klasse.

Dieses „Pulsar“ getaufte Muster für die Entwicklung einer Klasse läßt sich wie folgt interpretieren:

- Funktionserweiterungen führen zu größerem Volumen
- Restrukturierung der Klasse läßt Volumen wieder sinken

Vermutlich stehen diese Klassen oft im Zentrum der Entwicklungsaktivität.

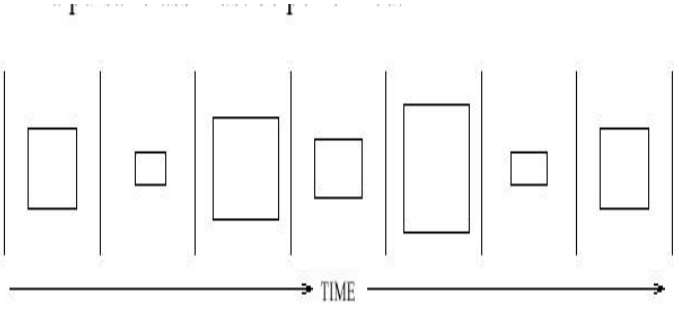

Abbildung 7 Pulsar Muster

Dieses Muster trägt den Namen „Supernova“. Sein Auftreten deutet auf folgendes hin:

- Kann Folge eines System Refactoring sein
- reine Datenklasse, kann aufgrund der einfachen Struktur sehr groß werden
- Implementierung wurde erst kürzlich hinzugefügt, bisher nur Klasse definiert
- Kann Anzeichen für Design Unklarheiten sein

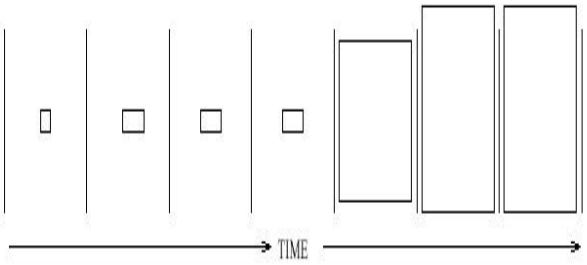

Abbildung 8 Supernova Muster

Anmerkungen

Softwarevisualisierung kann ein mächtiges Werkzeug zur Unterstützung des Entwicklungsprozesses sein. Wer schon als Entwickler in ein nur etwas größeres Projekt „nachgerückt“ ist, wird sich sicher wünschen ein solches Hilfsmittel auch bei seinem Projekt zu haben.
Softwarevisualisierung braucht Unterstützung von weiteren Tools. Sind Voraussetzungen wie die Verwaltung von verschiedenen Codeständen mittels Versionsverwaltung und Erfassung weiterer Daten z.B. via Bug Tracking Systemen nicht erfüllt ist eine Visualisierung nur eingeschränkt oder überhaupt nicht möglich.
Man kann keine „Evolution Matrix“ erstellen, wenn man nur den Code des aktuellen Systems hat.
Im Vergleich der beiden Techniken sind die Verfahren von G. Eick sicher die praxistauglicheren. Seine Verfahren entstammen auch der praktischen Anwendung bei einem viele Jahre laufenden Grossprojekt. (vgl. den nachfolgenden Vortrag)
Die Evolution Matrix hat interessante Ansätze , gerade bei der Klassifikation und Interpretation von Mustern wie „Pulsar“ etc ist aber noch viel Forschungsbedarf.
Auch die Anwendbarkeit auf große Projekte ist wohl nicht gegeben.

Die wirkliche Mächtigkeit von Visualisierungstechniken erschließt sich aber nicht aus einem Paper, sondern nur aus dem Umgang mit dem System. Der interaktive schnelle Wechsel zwischen verschiedenen Aspekten der Visualisierung , unterschiedlichen Auflösungen und manipulieren der Zeitskala verschafft einen tieferen Einblick in das System als Abbildungen es vermitteln können.

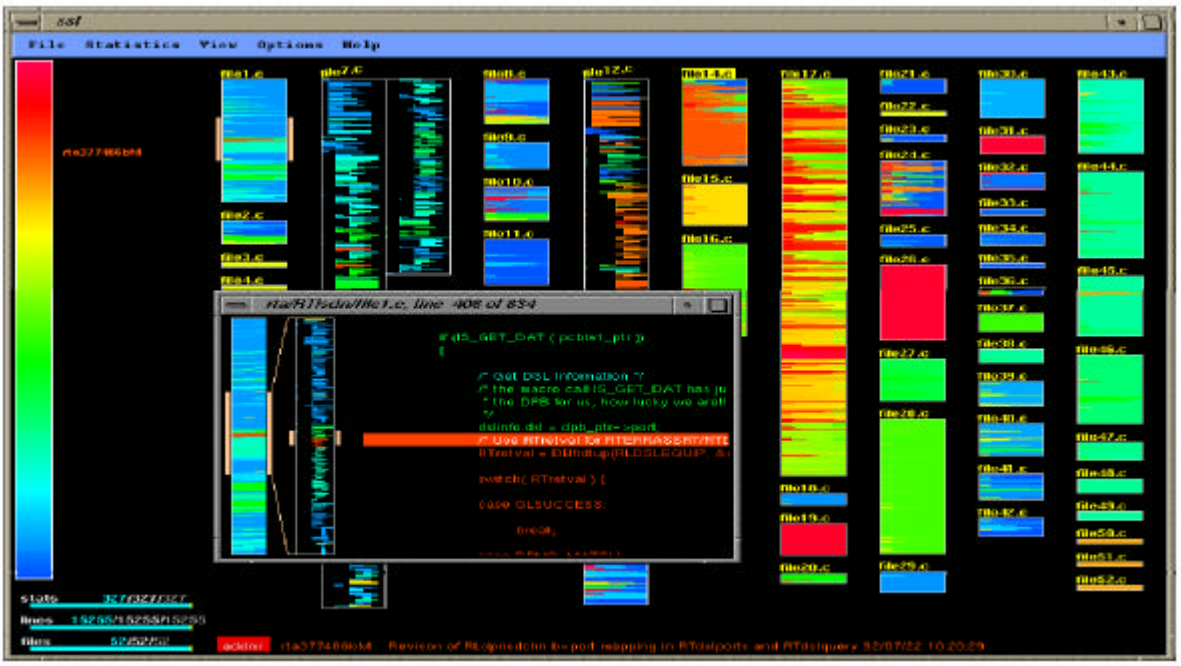

Abbildung 9 Navigieren per Visualisierung durch den Code

Literatur

Visualizing Software Systems; Marla J. Baker, Stephen G. Eick

Software Visualization in the Large; Thomas Ball, Stephen G. Eick

The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques; Michele Lanza