Algorithmenerkennung und Rekonstruktion

(abstract)

1. Softwarearchäologie und Softwarewartung

Der Begriff ‚Softwarearchäologie' wurde von H. Sneed als abschreckendes Synonym für Softwarewartung eingeführt, da man sich bei der Softwarewartung analog zur Archäologie oft durch Verkrustungen und Deckschichten durcharbeiten muss, um zum Kern der Dinge vorzustoßen.
Die Gesellschaft für Softwarearchäologie akzeptiert zwar die Methodenverwandtschaft zur Softwarewartung sieht aber die Softwarearchäologie als eigenständige Disziplin, bei der es u.a. um folgende Schwerpunkte geht

- Sammlung von beispielhaften Softwareartefakten aus allen Bereichen der Softwareentwicklung
- Entwicklung bzw. Zusammenstellung einer Vorgehensweise und eines Methodensets zur Softwareanalyse bezüglich zugrundeliegender Algorithmen
- Sammlung und Beschreibung algorithmischer Pattern und Signaturen
- Erarbeitung eines Vorgehens zur Ableitung der Code-Semantik aus Syntax und Kontext

Damit kann die Softwarearchäologie durchaus der Softwarewartung helfen indem sie Struktur und Bedeutung von Programmalgorithmen zur Verfügung stellt.
Ein vereinfachtes Beispiel ist der normierte Algorithmus einer Gruppenverarbeitung, der jungen Softwareentwicklern häufig völlig unbekannt ist, da sie ihn nicht (mehr) gelehrt bekamen und ihn bei datenbankgestützten Verarbeitungen nicht brauchten.

2. maschinelle Programmanalyse

Die verfügbaren Tools zur maschinellen Programmanalyse beschränken sich im wesentlichen auf folgende Schwerpunkte

- Suche und Erkennung von Begriffen bzw. Begriffskombinationen in Quelltexten
- Erkennung und grafische Aufbereitung modulinterner Strukturen gemäß der zugrundeliegenden Syntax (if-then-else, Schleifenkonstrukte etc.)
- Erkennung und grafische Aufbereitung von Aufruf- und Vererbungsstrukturen (cross-reference)

Diskutiert werden im folgenden die Möglichkeiten, Probleme und Grenzen von Programmanalysen unter Nutzung der Methoden, die von Virenschutzsoftware zur Erkennung von Virensignaturen eingesetzt werden.

3. Algorithmendarstellung in Pattern und Signaturen

Um bei der Programmanalyse brauchbare Ergebnisse zu erhalten, müssen die Algorithmen in geeigneter Weise dargestellt werden.
An einigen praktischen Fällen werden mögliche Algorithmen-Pattern bzw. Signaturen gezeigt und diskutiert. Betrachtet wird dabei auch das Problem der unscharfen Erkennung unter Berücksichtigung von Ähnlichkeiten und die Definition von Ähnlichkeitsmaßen, da es ja bekanntermaßen bei der Programmierung ‚viele Wege nach Rom' gibt.

4. Erkennung der Semantik

Nach der technischen Analyse ‚wie arbeitet die Software' ist die Frage nach dem ‚was tut die Software' für die Softwarearchäologie und die Softwarewartung gleichermaßen interessant.
Bekanntlich lässt sich aus dem Quellcode allein die Semantik nicht vollständig ableiten.
Untersucht wird in dem Vortrag u.a.,

- welche weiteren Informationen (Kontext) nötig sind, um mit hinreichender Sicherheit semantische Aussagen machen zu können

- welche Möglichkeiten und Grenzen es für den Check der Konsistenz von Syntax und ermittelter Semantik gibt

5. praktische Verwertung der Ergebnisse der Softwarearchäologie

Die Ergebnisse

- Vorgehensweise und Methodenset
- Pattern-/Signaturdatenbank

lassen sich verwenden, um bei schwierigen Fällen der Softwarewartung bzw. der Beurteilung von Notwendigkeiten/Möglichkeiten des Reengineering bzw. der Schätzung von Migrationsaufwänden qualitativ bessere Aussagen zu erhalten.
In bestimmten Fällen kann damit auch nach besseren Realisierungsvarianten für schlecht realisierte Algorithmen gesucht werden bzw. können degenerierte (‚verschlimmbesserte') Algorithmen in Anlehnung an den ursprünglich beabsichtigten Zustand rekonstruiert werden.
Last but not least können mit den Methoden der Softwarearchäologie Fragen der DV-Revision zu historischen Softwareständen beantwortet werden.
Beispiele werden aufgezeigt und diskutiert.