Component Frameworks für Software Generatoren

A. Speck, E. Pulvermüller

Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen
email: [speck|pulvermu]@informatik.uni-tuebingen.de

Keywords: Komponenten, GenVoca Architektur, Aspektorientierte Programmierung

1 Einleitung

Das Konzept, Software aus Komponenten zusammenzusetzen, wurde bereits 1968 von McIlroy auf der NATO-Konferenz in Garmisch-Partenkirchen vorgeschlagen. Software soll wie beim Bau von Maschinen aus einzelnen Komponenten erzeugt werden.
In diesem Abstract wird eine Architektur (Component Framework nach C. Szyperski für komponenten-basierte Software vorgestellt, die durch Generatoren unterstützt wird. Kundenorientierte Software kann aus Standardkomponenten generiert werden. Die Software-Generation orientiert sich an der GenVoca Architektur und dem kollaborationsbasierten Design.
Das Component Framework ist auf Grundlage der Erfahrungen von unterschiedlichen, bestehenden Software-Systemen entwickelt worden [3].

2 Generische Software-Entwicklung

GenVoca Architekturen beschreiben die Zusammensetzung eines aus hierarchischen Schichten bestehenden Systems. Entsprechend dem kollaborationsbasierten Design können die Schichten als *Collaborations* realisiert sein. Von abstrakten Schichten (siehe Abschnitt 2.1) werden neue, spezifischere abgeleitet und damit das System an eine konkrete Anwendung angepaßt (siehe Abschnitt 2.2). Durch den Austausch von kompletten Schichten können von einer abstrakten Architektur unterschiedliche Applikationen erzeugt werden.

2.1 Component Framework

Das Component Framework liefert eine Vorlage (Muster) zur abstrakten Kombination von Komponenten [3].
Die Komponentenkategorien innerhalb dieses Frameworks sind (siehe Abbildung 1):

- *Backbone Component*: Diese Komponente kommt in einem System genau einmal vor. Die *Backbone Component* realisiert (bzw. kontrolliert) die zentralen Funktionen zur Kommunikation zwischen den anderen Komponenten. Damit steuert diese Komponente das Verhalten des Systems.

- *Basic Components*: Diese realisieren die grundsätzlichen Funktionen des Frameworks. Zusammen mit der *Backbone Component*, das die Methoden der *Basic Components* nutzt, bilden sie ein rudimentäres Basissystem.

- *Additional Components*: Diese Komponenten bilden die Erweiterung des Basissystems. Im Gegensatz zu den *Basic Components* sind die *Additional Components* nicht zwingender Bestandteil eines Systems.

Die Beziehungen zwischen den Komponenten sind Aggregationen (d.h. eine Komponente ruft die Methoden der anderen Komponente) sowie die Anpassung einer Komponente ("Hot Spot"-Anpassung nach W. Pree).

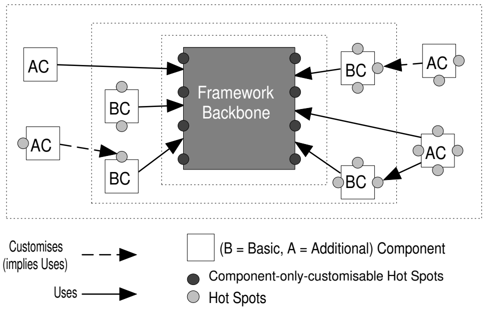

Abbildung 1. Generisches Component Framework

2.2 Kollaborationsbasierte Hierarchie

Entsprechend des GenVoca Vorgehens wird ein System aus verschiedenen Schichten (Kollaborationen) generiert. In einer GenVoca Realisierung nach werden die Komponenten der unterschiedlichen Ebenen durch Mixin-Vererbung verbunden. Diese Verbindung wird in dem hier beschriebenen Ansatz durch Komposition ersetzt. Hierbei ist der Code, der die Komposition realisiert in Aspekten gekapselt. Die Komposition ist als Aggregation realisiert und kann daher auch zur Entwicklung von verteilten Systemen verwendet werden [1].

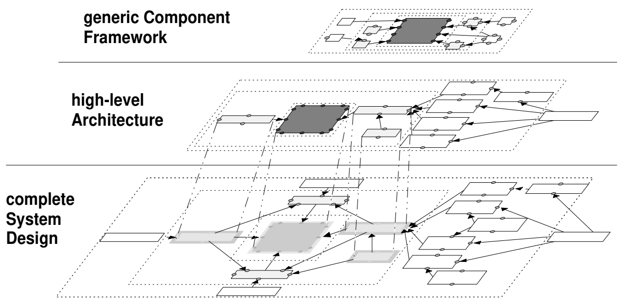

Abbildung 2. Hierarchische generative Architektur

Die Komponenten der oberen Schichten werden entweder durch Komponenten der unterlagerten Ebene verfeinert (spezialisiert) oder übernommen. Aus dem generischen Component Framework wird auf diese Weise ein spezifisches System unter Verwendung wiederverwendbarer Komponenten erstellt.
Die Definition der Beziehungen zwischen den Komponenten der unterschiedlichen Schichten wird durch eine GenVoca Grammatik beschrieben. Diese Beschreibung dient einem Generator als Vorlage zur Generierung der Beziehungen. Die Beziehungen sind dazu in Aspekten lokalisiert. Der Generator wählt die passenden Aspekte aus [2].

Literatur

[1] E. Pulvermüller, H. Klaeren und A. Speck. Aspects in Distributed Environments. In *Proceedings of the GCSE'99*, Erfurt.

[2] E. Pulvermüller, A. Speck und A. Rashid. Implementing collaboration-based Designs using Aspect-Oriented Programming. Generative and Component-based Approaches. TOOLS USA2000.

[3] A. Speck. Component-based Control System. In *Proceedings of ECBS'2000*, Edinburgh, 2000.