Kurzzusammenfassung

Ausgehend von dem wiederholt in der Musikwissenschaft vorgetragenen Wunsch nach effektiver, flexibler Rechnerunterstützung bei einer Vielfalt von Analysearten, erarbeite ich in dieser Dissertation sowohl softwaretechnische als auch musikdatenverarbeitungsbezogene Lösungen, um die in der Literatur diskutierten Probleme zu überwinden. Die Ergebnisse sind (1) eine einheitliche Beschreibung der Anforderungen an eine Anwendungsfamilie von Analysesystem bezüglich der Entwicklungskontexte *Anwendungsbereich*, *Handhabung & Präsentation* und *verwendete Technik*, (2) ein Ansatz zur bruchlosen Modellierung von Anwendungsfamilien von der Analyse bis hin zur für den Anwender verständlichen sitzungsweisen Konfiguration sowie (3) zwei Konstruktionskonzepte, mit denen komplexe technische Subsysteme – beispielsweise zur Musikdatenverarbeitung – auf harmonische Weise in anwendungsorientierte Softwaresysteme eingegliedert werden können, ohne die fachliche Modellierung aufweichen zu müssen. Entsprechend der Sichtweise von Software als sozial eingebettetem System validiere ich alle drei Ergebnisse konstruktiv anhand des *JRing*-Systems zur Unterstützung der musikwissenschaftlichen Analyse, das ich nach dem Werkzeug- und Material-Ansatz WAM zusammen mit Musikwissenschaftlern bis hin zur Produktreife entwickelt habe.

Zunächst konkretisiere ich den Begriff der Anwendungsfamilie und identifiziere dabei als wesentliche Bestandteile das Kernsystem und Komponenten, mit denen das Kernsystem an festgelegten Anpassungsstellen konfigurierbar ist. Anschließend betrachte ich den Anwendungsbereich der musikwissenschaftlichen Analyse und spezifiziere ausgehend von den Gemeinsamkeiten und Unterschieden in den Anforderungen eine Anwendungsfamilie von Analysesystemen samt Kernsystem, Anpassungsstellen und Komponenten. Dabei arbeite ich heraus, dass sich die Anforderungen von Sitzung zu Sitzung ändern können und somit das Kernsystem vor jedem Start durch die Musikwissenschaftler anpassbar sein muss. Darauf aufbauend untersuche ich die softwaretechnischen Eigenschaften, die Kernsysteme und Komponenten aufweisen müssen, um auf verständliche Weise durch ihre Anwender anpassbar zu sein. Als geeignete Grundlage für eine softwaretechnische Realisierung identifiziere ich eine Kombination von Rahmenwerk- und Komponenten-Ansätzen, da reine Rahmenwerk-Ansätze sich als zu inflexibel und reine Komponenten-Ansätze sich als nicht durch Anwender handhabbar erweisen. Da ich das Konzept der Systemmetapher als zentral für die Verständlichkeit und somit der Handhabbarkeit der Auslieferungseinheiten einer Anwendungsfamilie herausarbeite, zeige ich zunächst die Probleme der in der Literatur vorgeschlagene Metaphern auf, in denen Komponenten u.a. als Legosteine, aktive Dokumente, Server oder Geräte einer Stereoanlage versinnbildlicht werden, und stelle anschließend die Systemmetapher von Einschub und Einschubrahmen vor, welche die zuvor beschriebenen Probleme nicht aufweist. Mit dem anschließend vorgestellten Einschub-Ansatz, können Anwendungsfamilien gemäß dieser Metapher konstruiert, ausgeliefert und durch die Anwender konfiguriert werden. Abschließend zeige ich, wie komplexe technische Subsysteme durch Materialien und fachliche Datentypen sowohl in Verbindung als auch unabhängig vom Einschub-Ansatz gekapselt werden können.

Abstract

Musicologists have repeatedly called for effective, flexible computer tools to facilitate the tasks that arise during various types of musicological analyses. This dissertation presents solutions in the field of software engineering as well as in the field of computer-assisted musicology to overcome the obstacles that have stood in the way of this kind of computer assistance so far. These solutions are threefold: (1) a uniform description of a comprehensive application family of analysis systems regarding the software development contexts *application domain*, *handling & presentation*, and *technology*; (2) a design approach that allows for seamless modeling of application families from analysis all the way to user-driven configuration from session to session; (3) two construction methods for integrating complex subsystems – such as existing musicology software – into user-oriented software systems without generating conflicts between user-oriented and technical design objectives. As the success of software solutions only shows in practice, I implemented the *JRing* family of music analysis systems in close cooperation with musicologists following the tools and materials approach. The *JRing* system incorporates all three solutions and attains the quality level of a commercial software product.

Starting with the basic application family concept, I identify its main constituent parts: the core system and components that can be used to configure the core system at predefined adaptation spots. I then apply this categorization to the domain of musicological analysis in order to specify an analysis system application family with (a) a core systems that represents the commonalties and (b) components that represent the variability of different kinds of analysis requirements. As I can demonstrate that all variable requirements can change from session to session, musicologists must be able to re-configure the systems completely at each startup. Starting from these premises I examine those features that core system and components need to have in order to be re-configurable in such manner: Framework-only solutions are too inflexible and components-only solutions are too difficult to be handled by users, wherefore the focus shifts to approaches that combine frameworks with components. As technical feasibility is not enough when users themselves have to do the configuration, I introduce the “system metaphor” concept to discuss various suggestions for describing the ways in which core system and components can be combined. After showing that known metaphors such as Lego blocks, active documents, servers or stereo components are problematic, the “slide-in and slide-in frame” metaphor is introduced. It avoids the previously described problems and forms the basis for the slide-in approach, making it possible to construct and to deploy user-configurable application families. Focussing on the third initial problem, I demonstrate how existing complex subsystems can be encapsulated by means of materials and domain-specific data types. These two methods can be applied in conjunction with or in isolation from the slide-in approach.