Verbunddokumente als Nutzeroberfläche von Software für die Tragwerksplanung

Dipl.-Ing. Daniel Bittrich

Informatik im Bauwesen, Bauhaus-Universität Weimar

1 Motivation

Der Schwerpunkt von Forschung und Entwicklung auf dem Gebiet der Tragwerksplanungs-Software lag in den letzten Jahren auf der Erweiterung des funktionalen Umfangs. In der Folge ist es notwendig, den gestiegenen Funktionsumfang einem möglichst breiten Anwenderkreis durch ingenieurgemäß gestaltete Nutzeroberflächen zugänglich zu machen, so dass ein möglichst effizientes und fehlerarmes Arbeiten ermöglicht wird. Obwohl der Stellenwert ingenieurgemäßer Nutzeroberflächen seit längerem bekannt ist, wurde dies bislang bei der Entwicklung von Tragwerksplanungs-Software nur selten angemessen berücksichtigt. Eine Ursache dafür ist der im Vergleich zur Entwicklung der Kernfunktionalität unverhältnismäßig hohe Aufwand zur Erstellung ingenieurgemäßer Nutzeroberflächen. Im folgenden wird deshalb ein Konzept zur Überwindung dieses sowohl für Anwender als auch für Entwickler unbefriedigenden Zustands vorgestellt.

2 Stand der Technik

Der Markt für Tragwerksplanungs-Software ist gegenwärtig geprägt durch einige allgemeine FE-Systeme und eine unüberschaubare Vielzahl von Anwendungen zur Analyse spezieller Bauteiltypen (Platten, Unterzüge, Stützen, Fundamente, Dächer). Die Nutzeroberflächen sind aus entwicklungsgeschichtlichen Gründen nur bedingt konform zu den aktuellen Gestaltungsrichtlinien. Nur vereinzelt und nicht mit letzter Konsequenz werden innovative Konzepte verfolgt.

Bei der gegenwärtig typischen Arbeitsweise zerlegt der Ingenieur das Tragwerk in überschaubare Bauteile. Für diese werden die Schnittgrößen ermittelt und damit die Bauteile bemessen sowie deren Tragsicherheit und Gebrauchstauglichkeit nachgewiesen. Abschließend werden die Ergebnisse im Tragwerksbericht dokumentiert. Im Regelfall wird für jeden Arbeitsschritt ein eigenes Programm verwendet. Häufig sind zur Bearbeitung eines Bauteils drei verschiedene Anwendungen erforderlich, von denen jede über eine eigene Nutzeroberfläche und eine eigene Datenhaltung verfügt.

Eine Anpassung von Tragwerksplanungs-Software an die individuellen Wünsche des Anwenders ist gegenwärtig entweder gar nicht vorgesehen oder nur begrenzt über proprietäre Makrosprachen oder prozedurale Programmiersprachen möglich.

Mit den Mitteln der objektorientierten Programmierung ließ sich bisher eine Reduzierung des Entwicklungsaufwands nicht im erhofften Umfang erreichen. Eine Wiederverwendung auf der Basis von Klassenbibliotheken erfordert häufig eine Offenlegung des Quellcodes. Dies ist wegen der damit verbundenen Preisgabe von Know-how jedoch nur selten über Organisationsgrenzen hinweg praktiziert worden. Als Konsequenz wird beispielsweise sogar Textverarbeitungsfunktionalität immer wieder durch eigene Implementierungen realisiert.

Die vielfältigen Aufgaben des Tragwerksplaners erfordern ein breites Spektrum von Anwendungen, das nicht von einem Anbieter allein abgedeckt werden kann. Die bisher üblichen monolithischen Systeme erlauben nicht im erforderlichen Umfang eine Erweiterung durch externe Entwickler. Daraus resultieren kostspielige Mehrfachentwicklungen.

3 Gestaltung ingenieurgemäßer Tragwerksplanungs-Software

3.1 Ergonomische Nutzeroberflächen

Aus der Sicht des Anwenders stellt eine ergonomische Nutzeroberfläche ein wesentliches Qualitätsmerkmal dar. Der Anwender erwartet, dass er mit Hilfe von Software seine Aufgaben lösen kann (Effektivität), dabei nicht mehr Zeit als nötig aufwendet (Effizienz) und dennoch nicht sklavisch den Vorgaben der Technik folgen muss (Zufriedenstellung). An ergonomische Nutzeroberflächen werden deshalb die folgenden Anforderungen gestellt:

- die Benutzerführung muss dem spezifischen Arbeitsablauf angepasst sein
- die Informationen müssen konsistent dargestellt werden
- die Informationen müssen übersichtlich dargestellt werden

Von den genannten Anforderungen ist insbesondere eine dem Arbeitsablauf angepasste Nutzerführung wichtig, da diesbezügliche Mängel für den überwiegenden Teil der Nutzungsprobleme verantwortlich sind [1].

Für eine dem Arbeitsablauf entsprechende Nutzerführung bieten sich prinzipiell zwei gegensätzliche Wege an: generalisierte oder spezialisierte Nutzeroberflächen [2]. Der Vorteil generalisierter Nutzeroberflächen liegt darin, dass der Anwender eine Vielzahl von Problemen mit nur einer Nutzeroberfläche bearbeiten kann. Nachteilig ist, dass der Anwender nicht zielgerichtet zur Lösung des Problems geführt werden kann. Statt dessen muss der Nutzer selbst entscheiden, wie er seine Aufgabe mit den von der Anwendung zur Verfügung gestellten Mitteln bearbeiten kann. Im Gegensatz dazu können spezialisierte Nutzeroberflächen optimal an die Lösungsschritte eines Problems angepasst werden. Die Eingabe kann mit hoher Informationsdichte in der vertrauten Terminologie des Anwenders erfolgen und damit sehr effizient gestaltet werden. Durch die konsequente Befolgung plattformweiter und erforderlichenfalls fachspezifischer Gestaltungsrichtlinien kann der Einarbeitungsaufwand für eine große Zahl spezialisierter Nutzeroberflächen auf ein vertretbares Maß reduziert werden. Spezialisierte Nutzeroberflächen sind somit aus der Sicht des Anwenders wegen der damit erschließbaren Effizienzpotentiale wünschenswert.

3.2 Integration individueller Funktionalität in einheitliche Arbeitsumgebungen

Die Arbeitsschritte für Analyse, Bemessung und Nachweis eines Bauteils sowie deren Dokumentation gehören logisch zusammen und sollten zweckmäßiger Weise für jeweils eine Problemstellung zusammengefasst werden. Ein vielversprechender Ansatz zur Lösung dieser Problematik wird in einem sogenannten Verbunddokument gesehen [3]. Derartige Nutzeroberflächen werden als dokumentzentriert bezeichnet. Durch die gemeinsame Darstellung der Informationen in einem einheitlichen Dokument werden Übersichtlichkeit und Arbeitsfluss wesentlich verbessert. Aufgrund der gemeinsamen Datenhaltung können die bei anwendungszentrierten Nutzer- und Datenmodellen erforderlichen arbeitsaufwändigen Import- und Exportvorgänge komplett entfallen, da eine Integration der beteiligten Daten bei diesem Konzept inhärent ist (Abbildung 1).

3.3 Anpassbarkeit der Arbeitsumgebung durch den Anwender

Um eine dem Arbeitsablauf entsprechende Nutzerführung zu erreichen, müssen die zukünftigen Anwender möglichst weitgehend in den Entwicklungsprozess einbezogen werden. Im Idealfall würde der Tragwerksplaner seine Arbeitsumgebung durch Verknüpfen von vorgefertigten Black-Box-Bausteinen selbst zusammenstellen. Dies setzt voraus, dass die benötigte Funktionalität beispielsweise in Form von einfach einsetzbaren Steuerelementen und gewöhnlichen Software-Komponenten vorliegt.

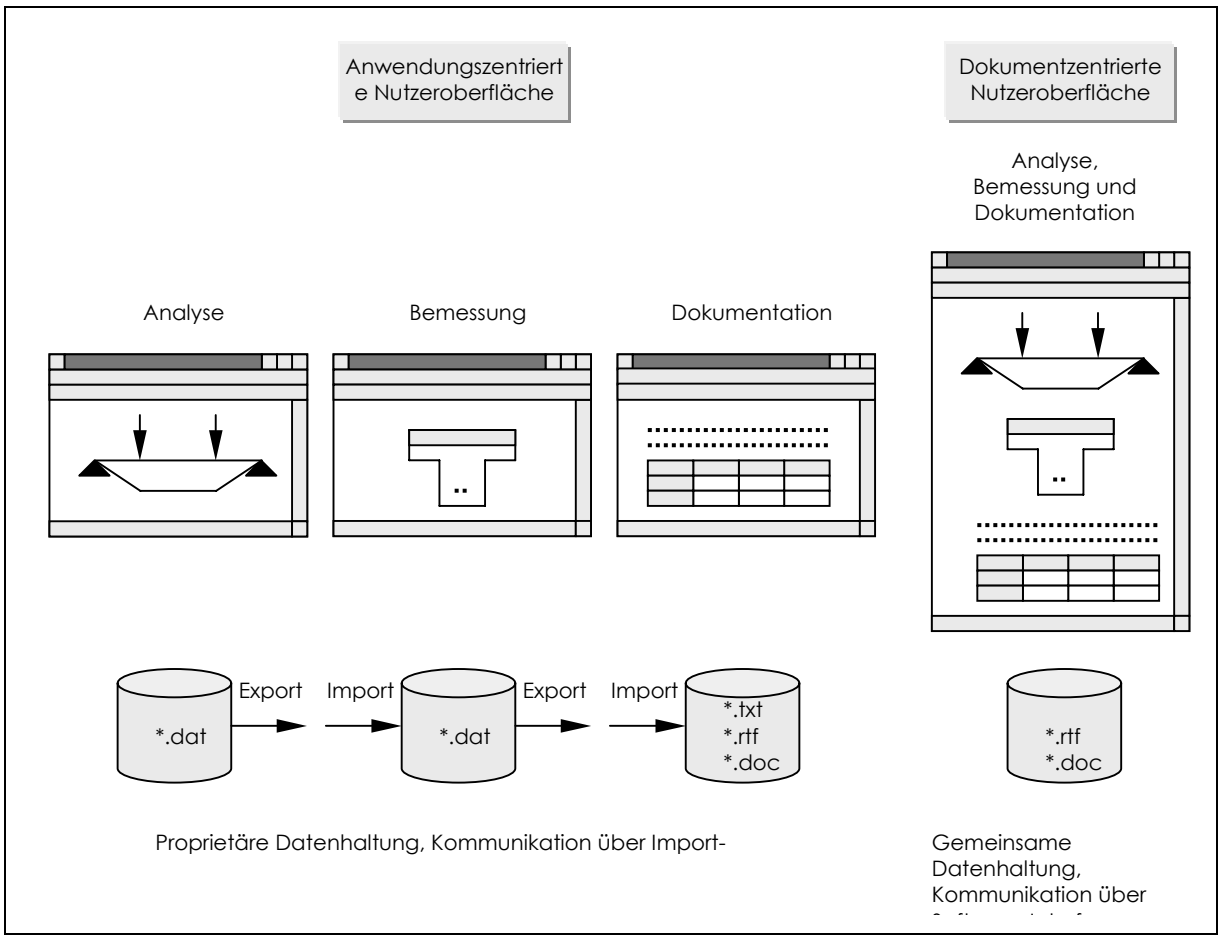

Abbildung 1: Anwendungs- versus dokumentzentrierte Nutzeroberflächen

4 Strategien zur software-technischen Umsetzung

4.1 Wiederverwendung von Funktionalität

Um die Rentabilität eines Software-Projektes sicherzustellen, müssen die Anforderungen der zukünftigen Nutzer mit möglichst geringem Entwicklungsaufwand erfüllt werden. Die gegenwärtig leistungsfähigste Technologie zur Begrenzung des Entwicklungsaufwandes stellt die Wiederverwendung von Funktionalität in Form von Software-Komponenten dar. Software-Komponenten sind binäre Black-Box-Bausteine, deren Implementierung vollständig verborgen ist. Eine Kommunikation ist nur über explizit definierte Schnittstellen möglich [4]. In der Tragwerksplanung wurden Software-Komponenten bereits erfolgreich zur Ausgliederung von Funktionalität für numerische Berechnungen eingesetzt. Damit ist eine Wiederverwendung der sogenannten Rechenkerne für mehrere Anwendungen möglich. Erfahrungsgemäß liegt jedoch der Codeanteil zur Implementierung von Rechenkernen wesentlich niedriger als der zur Erstellung von Nutzeroberflächen. Aufgrund des relativ geringen Codeanteils kann damit keine wesentliche Verringerung des Entwicklungsaufwandes für neue Anwendungen erreicht werden.

In [5] wird deshalb versucht, den Anteil an wiederverwendeter Software durch Einsatz von Codegeneratoren und herkömmlichen objektorientierten Technologien wie Class Frameworks zu erhöhen. Das Ergebnis sind jedoch wiederum quasi-monolithische Anwendungen. Damit bleiben die Unzulänglichkeiten der bisherigen Konzepte und Umsetzungen weitgehend erhalten.

Um den Komponentenanteil über das bisher erreichte Niveau zu erhöhen, müssen neben der bereits praktizierten Ausgliederung problemspezifischer Funktionalität in Software-Komponenten ohne Nutzeroberfläche auch Möglichkeiten genutzt werden, um Funktionalität in Software-Komponenten mit eigenen Nutzeroberflächen auszugliedern. Solche sogenannten Steuerelemente können zur Realisierung intelligenter Dokumentbausteine verwendet werden, die beispielsweise Funktionalität zur Darstellung von statischen Systemen, Schnittgrößenverläufen sowie zur Durchführung von Nachweisen bereitstellen.

Die zunächst völlig unabhängigen Dokumentbausteine müssen in einem sogenannten Container zu einer Anwendung zusammengefasst werden. Zu diesem Zweck wird als pragmatischer Lösungsansatz die Einbindung kompletter Verbunddokumentfenster einer kommerziellen Textverarbeitung gewählt. Damit können mit geringem Entwicklungsaufwand auch Merkmale wie die Erstellung von Inhaltsverzeichnissen oder eine Rechtschreibprüfung angeboten werden.

4.2 Unabhängig erweiterbare Systeme

Die Möglichkeit der Erweiterung durch externe Entwickler ist für eine Reduzierung des Entwicklungsaufwands noch wichtiger als die Wiederverwendung. Die dazu benötigte Technologie bilden unabhängig erweiterbare Systeme. Ein System gilt als unabhängig erweiterbar, wenn es erweitert werden kann und wenn es die Integration von unabhängig voneinander entwickelten Elementen erlaubt. Das Konzept der unabhängigen Erweiterbarkeit ist nicht neu: Jedes Betriebsystem erlaubt zur Laufzeit die Installation von unabhängig voneinander entwickelten Anwendungen, ermöglicht diesen, miteinander zu kommunizieren, und reguliert die Nutzung gemeinsamer Ressourcen. Eine Interoperabilität von unabhängig voneinander entwickelten Software-Komponenten ist jedoch nur möglich, wenn diese bestimmten Konventionen folgen. Offensichtlich sind derartige Konventionen vom jeweiligen Anwendungszweck abhängig. Die Einhaltung dieser domänenspezifischen Designprinzipien kann durch sogenannte Component Frameworks sichergestellt werden [6].

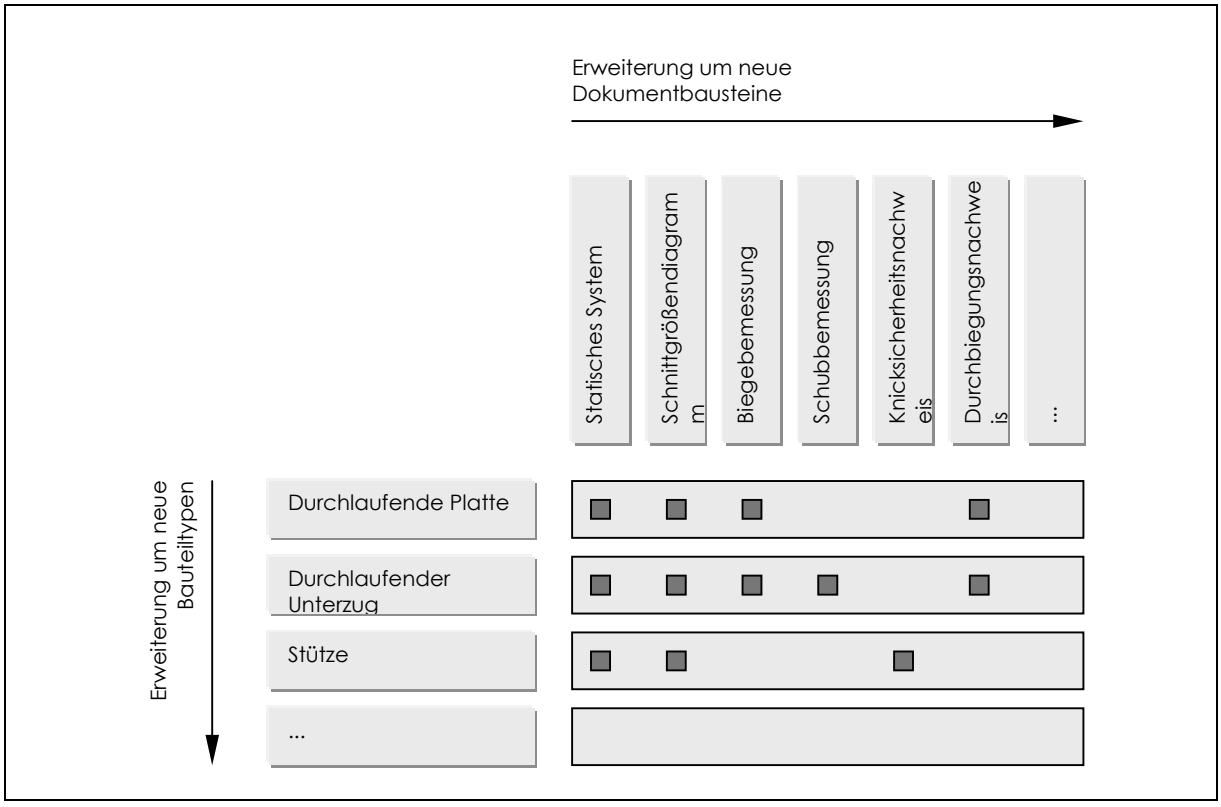

Abbildung 2: Dimensionen der Erweiterbarkeit des Component Frameworks

Ein Component Framework erlaubt die unabhängige Erweiterung eines Systems innerhalb vordefinierter Dimensionen. Ein System für die bauteilorientierte Tragwerksplanung muss die Erweiterung um Dokumentvorlagen für neue Bauteiltypen ermöglichen. Indirekt ist damit auch eine Erweiterung über die in den Dokumentvorlagen enthalten Dokumentbausteine möglich (Abbildung 2).

Das Prinzip der unabhängigen Erweiterbarkeit bedeutet aber auch, dass das System niemals global analysiert und somit kein allgemeingültiges, vollständiges Tragwerksmodell definiert werden kann. Da auch die Definition eines standardisierten Tragwerksmodell bislang nicht gelungen ist, kann eine Datenübernahme zwischen den Dokumenten nur durch ein semantikfreies Integrationskonzept erfolgen, welches das Fachwissen des Anwenders nutzt.

5 Pilotimplementierung

Die Umsetzung des Konzepts erfolgt auf der Windows-Plattform. Als Komponentenmodell wird das Component Object Model (COM) eingesetzt. Den Kern der Pilotimplementierung bildet der Tragwerkseditor, der zur Verwaltung aller zu einem Projekt gehörenden Positionen dient. Innerhalb des Tragwerkseditors können über die Technologie der Active Documents reguläre Textverarbeitungsdokumente angezeigt und bearbeitet werden (Abbildung 3).

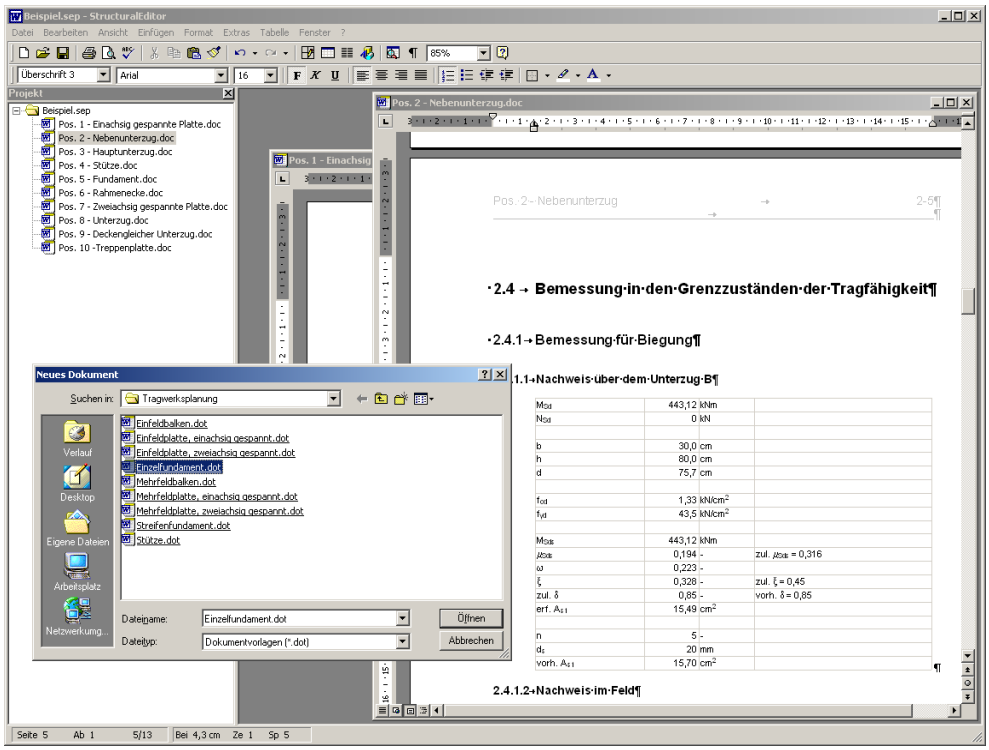

Abbildung 3: Pilotimplementierung des Tragwerkseditors

Für jeden Bauteiltyp wird eine Dokumentvorlage definiert. Innerhalb dieser Dokumentvorlage sind die vorgefertigten Dokumentbausteine platziert. Für jeden Bauteiltyp wird der Informationsfluss zwischen den Dokumentbausteinen durch ein mit der Dokumentvorlage verknüpftes Skript geregelt. Weiterhin kann der Anwender das Verbunddokument beliebig mit zusätzlichen Erläuterungen, Kommentaren und Skizzen ergänzen.

Die für die Berechnungsaufgaben verwendeten Dokumentbausteine basieren auf einem speziell entwickelten Tabellensteuerelement. Dieses passt seine Darstellung (z.B. Schriftart und –größe) selbständig der umgebenden Formatierung des Containers an und stellt umfassende Formatierungsmöglichkeiten (z.B. Hoch- und Tiefstellung, griechische Symbole und Runden) bereit.

Der Anwender kann dem System jederzeit weitere Bauteiltypen hinzufügen, indem er die entsprechende Dokumentvorlage auf den Rechner kopiert. Zur Laufzeit kann die gewünschte Funktionalität aus der Liste der installierten Bauteiltypen ausgewählt werden.

Die Datenhaltung der Dokumentbausteine basiert auf dem im Rahmen von COM definierten Structured-Storage-Konzept. Damit werden innerhalb eines Dokuments virtuelle Ordner angelegt, in denen beliebige Daten als binäre Streams abgelegt werden können. Somit werden keine proprietären Dateiformate benötigt, da alle relevanten Informationen automatisch durch die Containeranwendung in deren Dokumentformat gespeichert werden.

6 Bewertung

Das vorgestellte Konzept für ingenieurgemäße Tragwerksplanungs-Software vereinfacht die Bearbeitung alltäglicher Aufgaben. Durch die spezialisierten Nutzeroberflächen wird eine optimale Anpassung an den Arbeitsablauf ermöglicht, gleichzeitig kann der Eingabeaufwand infolge gesteigerter Informationsdichte reduziert werden. Die auf Verbunddokumenten basierende Nutzerführung erlaubt einen verbesserten Arbeitsfluss, indem unnötige und nachbearbeitungsintensive Import- und Exportvorgänge vermieden werden. Der Anwender findet sich aufgrund der vertrauten Nutzeroberfläche der Textverarbeitung intuitiv zurecht.

Aus der Sicht der Software-Industrie liefert der verfolgte Ansatz einen Beitrag, um die Entwicklung zeitgemäßer Tragwerksplanungs-Software wieder beherrschbar und rentabel zu machen. Der vorgeschlagene Ansatz führt allerdings nicht dazu, dass ein neues Software-System kurzfristig mit erheblich verringertem Aufwand erstellt werden kann. So erfordert die Entwicklung der Infrastruktur (Tragwerkseditor sowie die allgemeinen Steuerelemente) zunächst beträchtliche Vorleistungen. Aufgrund des hohen Wiederverwendungspotentials und der hohen Leistungsfähigkeit ist dieser Aufwand aber langfristig gerechtfertigt. Wesentlich ist, dass Bestandteile mit geringem und mittlerem Wiederverwendungspotential mit geringem Aufwand und durch Entwickler mit Ingenieurausbildung erstellt werden können (Tabelle 1).

Bestandteil	Anzahl	Aufwand	Wiederver-wendung	Entwickler
Bauteiltypen (Dokumentvorlagen)	sehr groß	sehr gering	gering	Bauingenieur
Bauteiltypen (Skripte)	groß	mittel	mittel	Bauinformatiker
Dokumentbausteine (spezifisch)	groß	mittel	hoch	Bauinformatiker
Dokumentbausteine (allgemein)	gering	sehr hoch	sehr hoch	Informatiker
Tragwerkseditor	1	sehr hoch	sehr hoch	Informatiker

Tabelle 1: Gegenüberstellung von Entwicklungsaufwand und Wiederverwendungspotential

[1] "When GUIs Fail: Usability is more than presentation", System Concepts, London, 1997

[2] Holzer, Stefan, "Gestaltung ingenieurgemäßer Statiksoftware", Bauingenieur 3/1997, Springer Verlag, Berlin

[3] Beucke, Karl, "Stand der Integration von Statik und Bemessung im Entwurfs- und Konstruktionsprozeß", Bauingenieur 2/1996, Springer Verlag, Berlin

[4] Szyperski, Clemens, "Component Software: Beyond Object-Oriented Programming", Addison-Wesley, Essex, 1998

[5] Hinz, Olav, "Objektorientierte Modelle und Werkzeuge für den Einsatz von Komponentensoftware in der Tragwerksplanung", Dissertation, TU München, Shaker-Verlag, Aachen, 1998

[6] Weck, Wolfgang, "Independently Extensible Component Frameworks", Proceedings, International Workshop on Component-Oriented Programming, in Mühlhäuser, M., (ed.), Special Issues in Object-Oriented Programming – ECOOP96, Workshop Reader, dpunkt Verlag, Heidelberg, 1997