1 Einleitung und Zielsetzung der Arbeit

Der Bereich der mathematischen Optimierungssysteme hat in den letzten Jahren erhebliche Leistungssteigerungen und Funktionsverbesserungen erfahren (vgl. [Bixby02]). Die teilweise enormen Steigerungen in Bezug auf Rechengeschwindigkeit und Lösungsqualität wurden durch Verbesserungen der numerischen Kerne erreicht. Hingegen haben die Schnittstellen und Umfeldsysteme (Modellierungssprachen, integrierte Modellierungssysteme etc.) der Optimierungssysteme zwar auch Verbesserungen erfahren, jedoch oftmals in weit geringerem Maße als die numerischen Kerne. Dies gilt insbesondere für Optimierungssysteme aus dem akademischen und semi-kommerziellen Bereich, da hier das Augenmerk der Entwickler auf anspruchsvollen und innovativen numerischen Algorithmen liegt und dem Bereich der Schnittstellen und Umfeldsysteme oft weniger Aufmerksamkeit entgegengebracht wird.

Aus dem mangelnden Fokus auf Schnittstellen- und Integrationsaspekte resultieren diverse Probleme bei der Einbettung von Optimierungssoftware in betriebliche Anwendungssysteme. Solche Probleme, die später noch detaillierter erläutert werden, können beispielsweise sein: Nicht vorhandene oder mangelhafte Anbindung an Modellierungssprachen, Nichtverfügbarkeit von programmiersprachenspezifischen Schnittstellen für die Modellgeneratorentwicklung, mangelhafte Anbindung an Datenbanken und andere Datenquellen, Nichtaustauschbarkeit von Solvern, unzureichende Dokumentation, mangelhafte Debugging-Möglichkeiten, mangelhafte Fehlertoleranz von Optimierungssystemschnittstellen und Einiges mehr.

Angesichts dieser und weiterer Problembereiche verfolgt die vorliegende Arbeit die folgenden Zielsetzungen:

Zunächst soll eine vergleichende Darstellung von Schnittstellenkonzepten verbreiteter Optimierungssysteme und Modellierungssprachen vorgenommen werden. Hierbei handelt es sich jedoch keineswegs nur um eine einführende, rein deskriptive Darstellung. Der detaillierte Vergleich des State-of-the-Art von Optimierungssystemschnittstellen ist vielmehr ein eigenständiges Ziel und eine Leistung dieser Arbeit, da dieses Thema in der Literatur bisher kaum behandelt wird.

Ein weiteres, zentrales Ziel ist die Konzeption einer Architektur für ein umfassendes Middleware-System zur Anbindung unterschiedlicher Solver und Modellierungssprachen in betriebliche Anwendungssysteme unter Berücksichtigung der Erfordernisse unterschiedlicher Programmier- und Systemkontexte. In diesem Zusammenhang werden Callable Libraries, Klassen- und Komponentenbibliotheken konzipiert, die für unterschiedliche Programmierkontexte eine optimal angepasste Modellgenerierung und Solveranbindung ermöglichen sollen. Als

unmittelbare Endnutzerschnittstelle wird darauf aufbauend ein integriertes Modellierungssystem entworfen, das für unterschiedliche Solver und Modellierungssprachen offen ist.
Drittes Ziel ist die Umsetzung von Teilen des zuvor erstellten Konzepts in zwei Bereichen:
Zum einen wird eine COM-Komponentenbibliothek implementiert, die die Modellgenerierung insbesondere aus Visual Basic (VB6, VB.NET, VBA) erleichtern soll. Vorteile eines Komponenten basierten Ansatzes sind beispielsweise die gute syntaktische Integration in die genannten Sprachen, Entwurfszeitfunktionalitäten wie Property Pages, Unterstützung für Automatisierungsinterfaces, wie Enumeratoren, sowie weitere Features, die später ausführlich erläutert werden. Die Bibliothek erlaubt die Anbindung mehrerer Modellierungssprachen und unterschiedlicher Solver.
Zum zweiten wird ein umfangreiches integriertes Modellierungssystem erstellt, das sich durch Offenheit und freie Verwendbarkeit, etwa in der Lehre, auszeichnet und das verschiedene Modellierungssprachen, wie AMPL, MPL, MathProg oder Lindo, integriert und ebenfalls prinzipiell mit unterschiedlichen Solvern arbeiten kann. „Öffentlichkeit“ ist dabei ein explizites Teilziel der Entwicklung, denn während fast alle größeren integrierten Modellierungssysteme einen kommerziellen und oft auch Solver gebundenen Hintergrund haben, ist das hier entwickelte System insbesondere für den akademischen Bereich frei verfügbar und wird später eventuell sogar mit Sourcecode bereitgestellt.
Als Referenzbeispiel und für alle Implementationen wird durchgängig das Optimierungssystem MOPS benutzt, für das ebenfalls diverse Implementationsarbeiten durchgeführt wurden, die teilweise Voraussetzung für die Anbindung an die Komponentenbibliotheken und die Modellierungssprachen, wie z.B. Callbacks, waren.