Model Based Design und Systemsimulation mit SimulationX und Toolintegration mit optiSLang

Dr. Andreas Uhlig[1*], Uwe Grätz [1]

[1] ITI GmbH, Webergasse 1, 01067 Dresden

{andreas.uhlig, uwe.graetz}@itisim.com

Zusammenfassung

Der routinemäßige Einsatz von leistungsfähigen Optimierungswerkzeugen wie optiSLang gehört bei FEM-Tools und zunehmend bei Systemsimulation zum integralen Bestandteil der Produktentwicklung. Es wird vorgestellt, welchen besonderen Anforderungen Systemsimulationstools im Hinblick auf das Model Based Design genügen müssen. Zusammen mit der Modellierungssprache Modelica werden Nutzen für den Anwender und Konsequenzen einer gleichungsbasierten Modellierung für den Toolentwickler besprochen. Am Beispiel der Sensitivitätsanalyse eines Membranzylinders wird gezeigt, wie die Toolkopplung von SimulationX mit optiSLang 4 effizient eingesetzt werden kann.

Keywords: Systems engineering, equation based modeling, Modelica, SimulationX, cylinder drive, OptiSLang, optimization tool chain

1 Einleitung

Der routinemäßige Einsatz von leistungsfähigen Optimierungswerkzeugen wie optiSLang gehört im Rahmen des Designprozesses mit FEM- und CFD-Tools zum integralen Bestandteil der Produktentwicklung. So ist es nicht überraschend, dass über Anwendungen von Optimierungstools vor allem im Umfeld von FE-Simulatoren berichtet wird. In diesem Beitrag soll eine andere Disziplin, *Systemsimulation* (als ein Teil von Systems Engineering), vorgestellt werden, der im modernen Designprozess eine wachsende Bedeutung zukommt. Zunächst werden kurz wesentliche Anforderungen an Modellierungsmittel und Simulationstools genannt. Zusammen mit der Modellierungssprache Modelica

[*] Kontakt: Titel, Name, Firma/Institute, Adresse, E-Mail

werden Nutzen für den Anwender und Konsequenzen einer gleichungsbasierten Modellierung für den Toolentwickler besprochen.

Wie Designoptimierung mit Systemmodellen von SimulationX ablaufen kann, wird am Beispiel eines Pneumatikzylinders gezeigt. Hierbei konnte die API von SimulationX von den Entwicklern von optiSLang 4 effizient eingesetzt werden, um eine komfortable Integration der Simulationssoftware in den Optimierungsprozess zu implementieren.

2 Anforderungen an Software für Systems Engineering

2.1 Model Based Design

Die Produktentwicklung steht vor großen Herausforderungen. Das betrifft Zeitdruck, Kosten, Qualität und Funktionssicherheit. Und damit sind nicht nur die Entwickler von Hightech-Produkten wie etwa Smartphones und Flugzeugen konfrontiert, sondern die Entwicklung (darüber hinaus auch die Fertigung und ggf. Betrieb) neuer Produkte ändert sich qualitativ. Computersimulationen werden eingesetzt, um möglichst früh wesentliche Eigenschaften des Produktes bewerten und optimieren zu können. Hochentwickelte Verfahren wie FEM und CFD sind in die Entscheidungsprozesse der Konstruktion integriert und sichern damit die Funktionalität von Bauteilen ab, bevor Prototypen zur Erprobung bereitstehen. Es soll im Folgenden nicht um „einfache“ Bauteile, sondern ausschließlich um Produkte gehen, die man als *System* bezeichnen kann - ohne dass das hier streng definiert werden soll; vgl. zum Systembegriff etwa [JANSCHEK]. Systeme bestehen aus Teilsystemen, diese aus Komponenten, die insgesamt eine Funktion erfüllen müssen. Während man bei der Entwicklung von Komponenten/Bauteilen die Randbedingungen und Einflüsse der Umgebung berücksichtigt, sind bei Untersuchung von Systemen zusätzlich die Wechselwirkungen der Teilsysteme untereinander einzubeziehen.
Teilsysteme erfüllen Teilfunktionen und werden oft von verschiedenen Entwicklungsteams konstruiert. Wie optimal kann dann das „zusammengesetzte“ Gesamtsystem sein? Kann ausgeschlossen werden, dass sich die Teile gegenseitig negativ beeinflussen? Um den notwendigen Test mithilfe von aufwändigen, realen Prototypen zu vermeiden, müssen die Funktionen des Systems entwicklungsbegleitend, virtuell getestet werden können. Dazu benötigt man Modelle und Programme, die sie bewerten, also simulieren. Simulationen sind in diesem Sinne an Modellen ausgeführte Experimente. Die Anforderungen an Modelle bzw. Simulationen steigen rasant. Es geht dabei nicht nur um die Größe (Anzahl Freiheitsgrade) und die Rechenzeit, sondern vor allem um qualitative Merkmale. Systemsimulation bereits in den frühen Phasen der Produktentwicklung, unter Verwendung und vor allem Wiederverwendung von Modellen, ist zur Voraussetzung geworden, um den komplexen Produktentwicklungsprozess zu beherrschen. Welchen Anforderungen

Modellierungs- und Simulationstools genügen müssen, wird im Folgenden dargestellt.

2.1.1 Multidomain Modelle

Spätestens seit dem Siegeszug der Mechatronik ist bekannt, dass man elektrische, mechanische (hydraulische, pneumatische, magnetische usw.) Effekte, Komponenten, Teilsysteme in *einem* sogenannten Multiphysics-Modell erfassen muss. In all diesen *Domänen* kann man Netzwerke von Elementen mit konzentrierten Parametern aufbauen. In ganz analoger Weise definiert man jeweils *through quantities* (Stromstärke, Kraft, Volumenstrom,…) und *across quantities* (Spannung, Geschwindigkeit, Druck,…). Man kann formale Regeln für die Verbindung von einzelnen Komponenten zu Systemen aufstellen und für die verschiedenen Domänen Bibliotheken von Elementarmodellen zur Verfügung stellen.
Damit ein Modellierungskonzept von Entwicklungsingenieuren akzeptiert wird, müssen die Begriffe, Formeln, Entwurfsmethoden der jeweiligen Disziplin unterstützt werden. Andererseits sind diese in der Regel sehr heterogenen Modelle für die Simulationsrechnung auf ein einheitliches mathematisches Modell, beschrieben in einer geeigneten Sprache, abzubilden. Abbildung 1 zeigt mit Signalflussplänen, Netzwerkmodellen, Mehrkörperstrukturen und Zustandsgraphen vier Modellierbeschreibungsmethoden auf Anwender-Niveau, die intern sämtlich auf die Sprache Modelica (siehe 2.2.1) abgebildet sind.

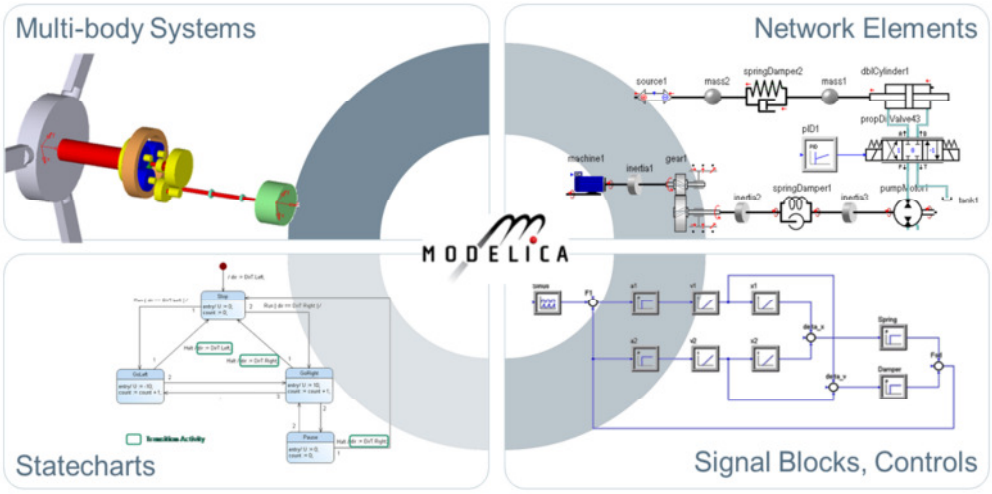

Abbildung 1: Vier Arten der Modellierung in SimulationX

Genau genommen deckt „multi-*physics*“ nicht einmal das gesamte Spektrum ab. Zum Beispiel enthalten detailliertere Modelle von Verbrennungsmotoren auch chemische Reaktionsgleichungen des Verbrennungsvorganges.

2.1.2 Hierarchische Modelle

Als wesentliches Strukturelement müssen Hierarchien in Modellen aufgebaut werden können. Das heißt, Teilsysteme können für die Wechselwirkung mit anderen Systemen auf ihr Schnittstellenverhalten reduziert werden. Im Entwicklungsprozess werden sowohl einfache Modelle zu Teilsystemen verfeinert, als auch Teilfunktionalitäten in ein *Compound*-Modell verpackt, sobald dessen innere Struktur nicht mehr Gegenstand der aktuellen Untersuchung ist. In engem Zusammenhang mit der Modellhierarchie steht die Forderung, Komponenten mit alternativen Modellen abzubilden und diese Alternativen im iterativen Entwicklungsprozess einfach umzuschalten:

2.1.3 Austauschbare Modelle

Untersucht man das dynamische Verhalten etwa eines PKW-Antriebsstranges, zum Beispiel für die Auslegung des Getriebes, dann reicht ein einfaches Motormodell, etwa auf Basis einer Drehzahl-Drehmomenten-Kennlinie. Will man hingegen Verbrauchsberechnungen durchführen, dann benötigt man ein ausführlicheres Modell. Eine Modellierungsumgebung soll also auch komfortable Mittel bieten, um zwischen Modellen unterschiedlicher Tiefe, zwischen alternativen Beschreibungsmethoden, zwischen verschiedenen Baureihen umzuschalten - per Mausklick als auch über eine Programmierschnittstelle.

2.1.4 Ein Modell – verschiedene Analysen

Naturgesetze und technische Formeln findet man in Lehrbüchern in Form von Gleichungen.

$$F = m \cdot a$$

Je nachdem, was man gerade berechnen will, programmiert man dann in Simulatoren Zuweisungen:

$$F := m \cdot a \qquad \text{oder} \quad a := F/m \qquad \text{oder} \ \ldots$$

Wenn wir Modelle allerdings in Gleichungsform, also *akausal* beschreiben, dann sind diese Modelle für ganz verschiedene Anwendungsfälle, für unterschiedliche Experimente einsetzbar. In SimulationX kann ein und dasselbe Modell für transiente Simulation im Zeitbereich, für die Berechnung des statischen Gleichgewichts, (nach Linearisierung in einem Arbeitspunkt) für Eigenfrequenzanalysen, Übertragungsfunktionen und Energiebilanzen verwendet werden. Darüber hinaus nutzt SimulationX die Modelltopologie, um Fehlerfortpflanzungsinformationen im Rahmen von Zuverlässigkeitsanalysen abzuleiten. Durch die Darstellung in Gleichungsform wird die Modellierung von Komponenten einfacher, weil erst durch deren Nutzung und Zusammenschaltung mit anderen Komponenten festgelegt wird, welche Größen gegeben und gesucht ist.

2.2 Physikalische Modellierung mit Modelica

2.2.1 Modellbeschreibungssprache Modelica

Die in Kapitel 1 genannten Anforderungen an Systemsimulation waren seit ca. 15 Jahren Ausgangspunkt einer Reihe von Modellierungsansätzen, wie Modellierung mit Basis von Bondgraphen oder die Modellbeschreibungssprachen VHDL-AMS und Modelica. Dabei hat Modelica Vorteile, von Ingenieuren ganz unterschiedlicher Disziplinen akzeptiert zu werden. Modelica ist eine nicht-proprietäre, objektorientierte, gleichungsbasierte Sprache zur komfortablen Modellierung komplexer technischer/physikalischer Systeme. Modelica-Modelle sind vorwiegend *akausal* im Sinne von 2.1.4. Darüber hinaus können Variablen mit zusätzlichen Attributen wie zum Beispiel physikalischen Größen und Maßeinheiten, zulässigen Wertebereichen u. ä. versehen werden. Damit bieten Modelica und die unterstützenden Software-Tools eine adäquate Umgebung für Systemsimulation. Die Transparenz und Selbstdokumentationsfähigkeit ist besser ausgeprägt als in Tools, die nur signalflussorientierte Ansätze verwenden.
Die Entwicklung und Pflege der Sprache liegt in den Händen der Modelica Association [MODELICA ASSOCIATION]. Sie ist auch verantwortlich für die Bereitstellung der sogenannten Modelica Standard Library (MSL) und für den neuen FMI-Standard zum Modellaustausch. Das Modellierungs- und Simulationstool SimulationX von ITI nutzt Modelica als Modell-beschreibungssprache.

2.2.2 Simulation von (akausalen) Modelica-Modellen

Mathematisch betrachtet stellen Modelica-Modelle Algebro-Differentialgleichungen (Differential Algebraic Equations, DAE) dar.

$$0 = f(x, \dot{x}, p, d, zf, t) \tag{1}$$

Dabei bezeichnet p die Parameter, und t ist die Zeit. Die Gleichungen enthalten außerdem diskrete Variablen d und sogenannte *zero functions* zf, mit denen kontrolliert wird, unter welchen Bedingungen sich diskrete Variablen ändern und damit das System in einen anderen (diskreten) Zustand wechseln kann. Diese Zustandswechsel (Events), deren Erkennung, präzise Lokalisierung und effiziente Behandlung, die im Allgemeinen einen Eingriff in die Schrittweitensteuerung der Differentialgleichungslöser erfordert, gehören zu den wesentlichen Herausforderungen, denen sich Systemsimulatoren stellen müssen.
Eine weitere Besonderheit ist die Bestimmung konsistenter Anfangswerte für die DAE. Da die Systembeschreibung im Allgemeinen nicht nur Differential- sondern auch rein algebraische Gleichungen enthält, sind zu Beginn der Rechnung nicht alle Anfangswerte des Systems bekannt bzw. die Anfangswerte können nur indirekt als Anfangsbedingungen (initial equations) angegeben werden. Aufgabe der Initialisierung ist es dann, aus diversen direkten und indirekten Angaben sowie unter Verwendung von optional vorzugebenden Startwerten, konsistente Anfangswerte zu berechnen bevor die Integration beginnt. Und schließlich ist ein

erheblicher analytischer Aufwand notwendig, um aus der Menge beliebiger, nichtlinearer, impliziter Gleichungen eine numerischen Lösern zugängliche mathematische Beschreibung zu erzeugen, zum Beispiel eine Darstellung von (1) in semi-expliziter Form:

$$\dot{x} = f(x, y, p, d, zf, t) \qquad (2)$$
$$0 = g(x, y, p, d, zf, t)$$

Hierbei sind die kontinuierlichen Variablen aufgeteilt in kontinuierliche Zustandsvariablen x und die rein algebraischen Variablen y. Im Allgemeinen ist außerdem eine Indexreduktion, das heißt eine Differentiation ausgewählter Gleichungen mit nachfolgenden Einsetzungen, notwendig.

Mit diesem Aufwand vor der eigentlichen Simulationsrechnung ermöglicht man dem Nutzer die bequeme Aufstellung von universell verwendbaren Modellen in Gleichungen (und verringert damit die Menge von zu verwaltenden Modellen). Außerdem können Teile des Modells bereits analytisch gelöst werden, Ableitungen und Jakobimatrizen können symbolisch bereitgestellt werden, und die numerische Lösung kann insgesamt optimiert werden.

3 Anwendungsbeispiel: Membranzylinder

3.1 Pneumatischer Zylinderantrieb

Pneumatische Antriebe werden häufig in Prüfmaschinen für statische Belastungen, aber auch für Schwingversuche verwendet.
Membranzylinder (Abbildung 2) werden eingesetzt, weil sie geringe Reibungs- bzw. Losbrechkräfte haben und weil sie wegen geringer Toleranzanforderungen preiswert zu fertigen sind. Allerdings lassen sich im Vergleich zu Kolbenzylindern nur geringe Hübe realisieren. Als Beispiel sei hier eine Anwendung zum Testen von Zahnimplantaten genannt, mit einem Hub von +- 0,5 mm bei 50 Hz.

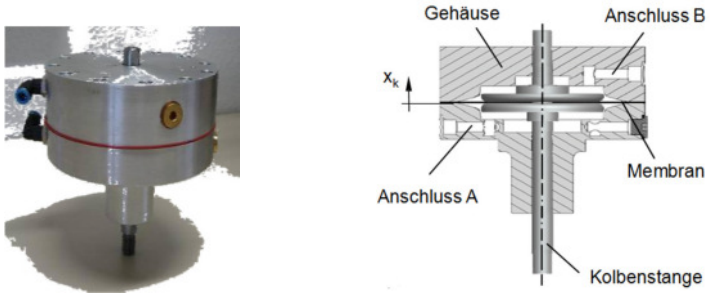

Abbildung 2: Membranzylinder.

Ziel der hier beschriebenen Entwicklung war es, einen bestehenden Antrieb so zu verändern, dass für eine andere Prüfaufgabe ein größerer Hub bei gleicher Frequenz realisiert werden konnte. Ventil und Belastung sollten beibehalten

werden. Um das zu erreichen, sollte der Membranzylinder modifiziert werden, speziell die inneren geometrischen Abmessungen. Damit die Funktion des Membranzylinders gewährleistet ist, unterliegen einige Abmessungen bestimmten Einschränkungen. (Abbildung 3a). Gleichzeitig war eine Verringerung des notwendigen Bauraumes angestrebt.

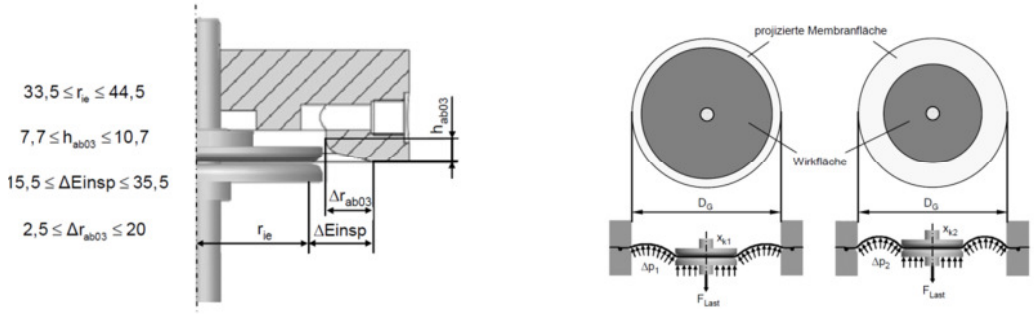

Abbildung 3: Membranzylinder, aus [FIEDLER]:
a) Auswahl von Abmessungen und Begrenzungen, b) Entstehung der Wirkfläche

Charakteristisch für einen Membranzylinder sind die komplexen Zusammenhänge zwischen Kraft und Druck bzw. Hub. Es ist zu erwarten, dass sich die Wirkfläche und das Membranvolumen aufgrund der Veränderung der Zylinderinnengeometrie während des Optimierungsprozesses ändern werden (Abbildung 3b). Um diese Veränderungen zu ermitteln, werden FEM-Berechnungen, sogenannte Stützstellenberechnungen, für die zu optimierenden Parameter durchgeführt. Dabei wurden die Wirkfläche über dem Kolbenhub und das Membranvolumen bei unterschiedlichen Druckdifferenzen bestimmt. Die berechneten Kurvenzüge werden auf die Ausgangslösung normiert und prozentual mit den Kennfeldern der Wirkfläche und des Membranvolumens verrechnet und dann auf andere Lasten und Kolbenpositionen extrapoliert. Daraus wird die entstehende Höhe der Tellerform der Membran als Amplitude x_k berechnet.

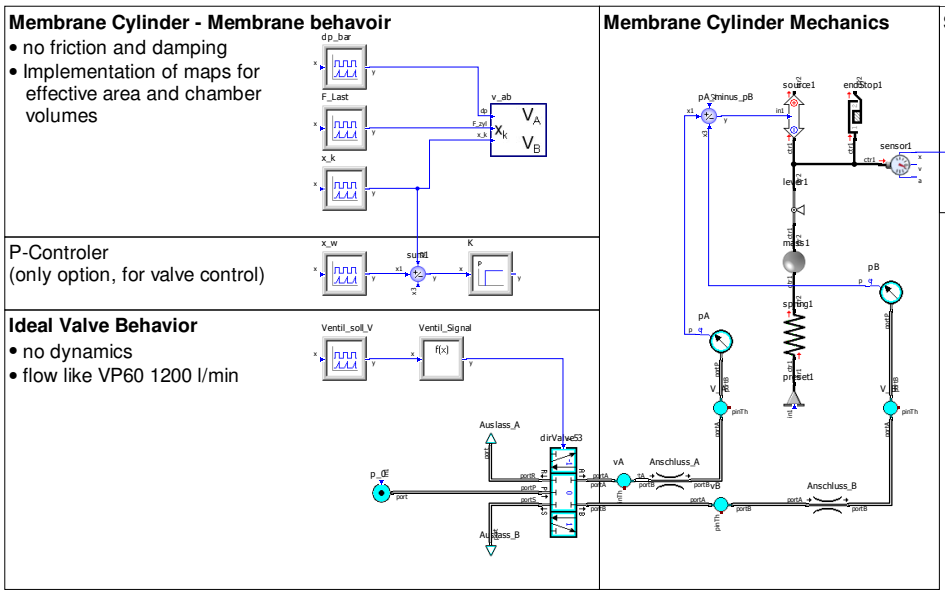

Abbildung 4: Modell des Membranzylinderantriebs in SimulationX

Damit wird sichergestellt, dass während der Zylindergeometrieoptimierung das Verhalten der Membran ausreichend berücksichtigt wird. Die aufbereiteten Daten aus der FEM-Berechnung wurden im Block *Membrane Behavior* des Simulationsmodells (Abbildung 4) abgebildet.

3.2 Sensitivitätsanalyse

Als Teil der Systemoptimierung wurde zunächst eine Sensitivitätsanalyse durchgeführt, die Informationen darüber liefern sollte, welche der insgesamt ca. 10 Designparameter den Hub x_k wesentlich beeinflussen. Dazu wurde ein Modell des Antriebs verwendet, dessen Struktur in Abbildung 4 ersichtlich ist. Links unten sorgt ein Ventil für die benötigte Hubfrequenz von 50 Hz. Es steuert die rechts abgebildete Zylindermechanik, wo auch die Zielgröße Amplitude x_k abgegriffen wird.

Der prinzipielle Test der Toolkette mit neuen SimulationX-Schnittstelle von optiSLang 4 sollte mit nur wenigen Parametern durchgeführt werden. optiSLang kann in der GUI für den SimulationX als Solver alle Variablen strukturgetreu abbilden (Abbildung 5).

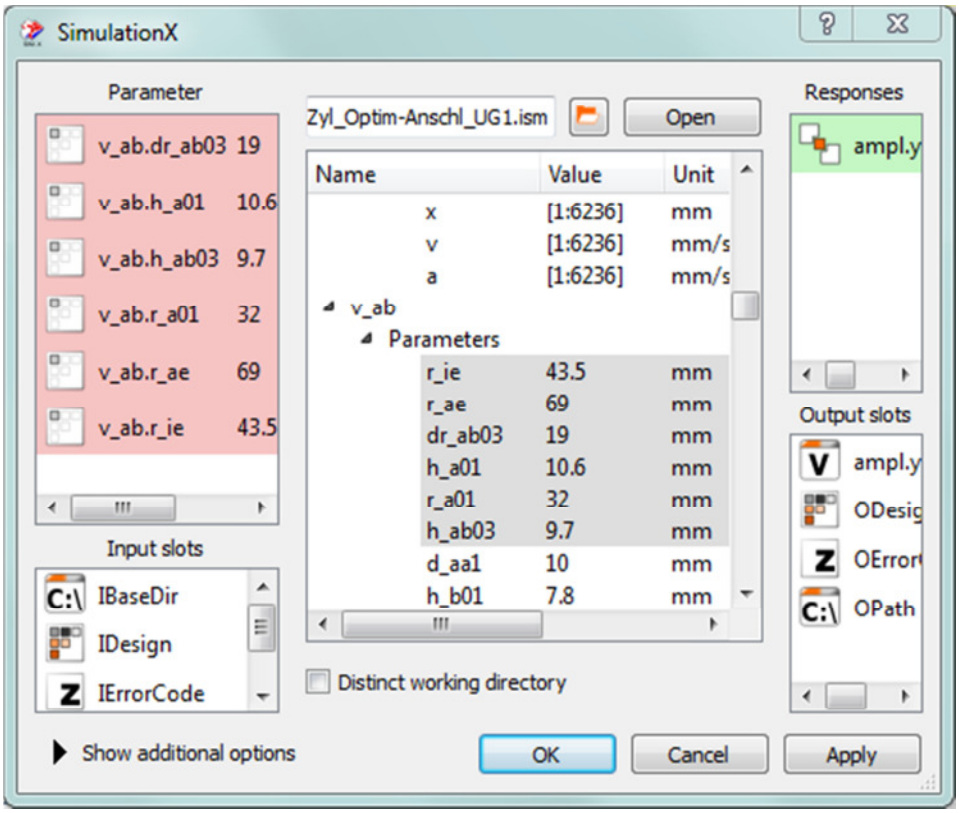

Abbildung 5: optiSLang Solver-Dialog für SimulationX

Zunächst wurden die Variablen dr_ab03, h_ab03, r_a01 und r_ie ausgewählt. Als response-Variable wird die Hubamplitude verwendet. Bei der Festlegung des Wertebereiches kann man die automatischen Voreinstellungen von optiSLang mit den konstruktiv bedingten Restriktionen überschreiben. Die Sensitivitätsanalyse ergab mit einem Coefficient of Prognosis (CoP) von 1% keine Hinweise auf optimierungsrelevante Parameter. Offensichtlich fehlten hier wesentliche Designparameter. Schon eine Erweiterung um zwei Variablen (r_ae, r_a01) (Abbildung 5) bringt hier wertvolle Hinweise für die eigentliche Optimierungsrechnung (auf die im Rahmen dieses Demonstrationsbeispiels verzichtet wurde). Abbildung 6 weist auf die für die Optimierung als sinnvoll erkannten Parameter r_ae und r_ie (mit akteptablen CoP) hin.

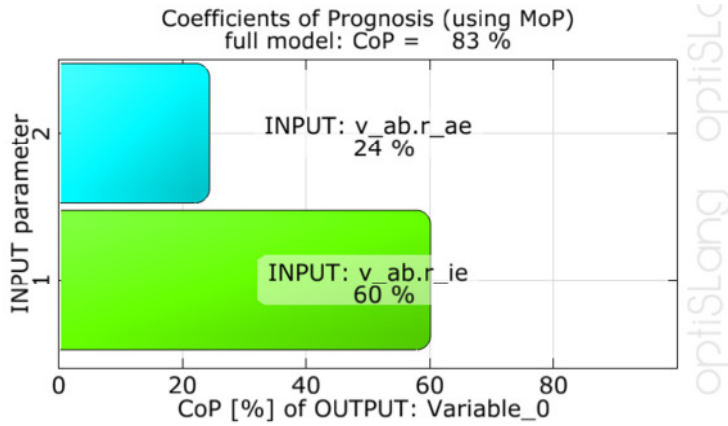

Abbildung 6: Result of Sensitivity Analysis with optiSLang

3.3 Toolintegration SimulationX – optiSLang

Wie bei allen im Produktentwicklungsprozess eingesetzten Berechnungsprogrammen ist auch bei Tools zur Systemsimulation die Möglichkeit, auf leistungsfähige Optimierungsverfahren zurückgreifen zu können, ein wesentliches Qualitätsmerkmal. Je umfassender die Computerunterstützung für die Designoptimierung ist, umso mehr kann sich der Ingenieur auf die kreativen Schritte der Modellierung konzentrieren. Die Entwickler von SimulationX haben sich entschieden, Optimierungsverfahren nicht innerhalb des Softwarepaketes zu implementieren. Stattdessen bekommt der Nutzer Zugriff auf leistungsfähige Optimierungstools von Spezialanbietern. Hier kann man erwarten, dass sowohl die Leistungsfähigkeit, Einsatzbreite und Zuverlässigkeit der angebotenen Verfahren, als auch eine komfortable Bedienung und effiziente Abläufe geboten und gepflegt werden. Mit optiSLang 4 kann ITI seinen Kunden - die nicht immer Experten in Optimierungsfragen sind - eine zuverlässige und auch verständliche Zusatzsoftware zu SimulationX empfehlen.
Die Akzeptanz einer derartigen Toolkopplung entscheidet sich an der Qualität der Schnittstelle zwischen dem Simulationsmodell und der Definition des Optimierungsproblems sowie an der Schnittstelle zwischen den

Optimierungsverfahren und den Aufrufen der Simulationsrechnung. Die Implementation der Kopplung erfolgte durch das Dynardo-Team. Die Programmierer konnten dabei auf eine umfassende API von SimulationX auf COM-Basis zurückgreifen (Abbildung 7), mit der alle parametrischen, strukturellen und Prozesseinstellungen ausgelesen bzw. gesteuert werden können. Im Optimierungsumfeld gewährleistet das insbesondere eine transparente Parameter- und Response-Festlegung, wenn SimulationX als Solver-Knoten im optiSLang-Workflow eingerichtet wird. Während der Analysen und Optimierung werden die spezifizierten Eigenschaften für jedes Design direkt im SimulationX-Modell eingestellt. Die erste Version der Toolkopplung hat sich damit als sehr gut nutzbar erwiesen. Weitere Arbeiten können sich der weiteren Verbesserung der Performance widmen.

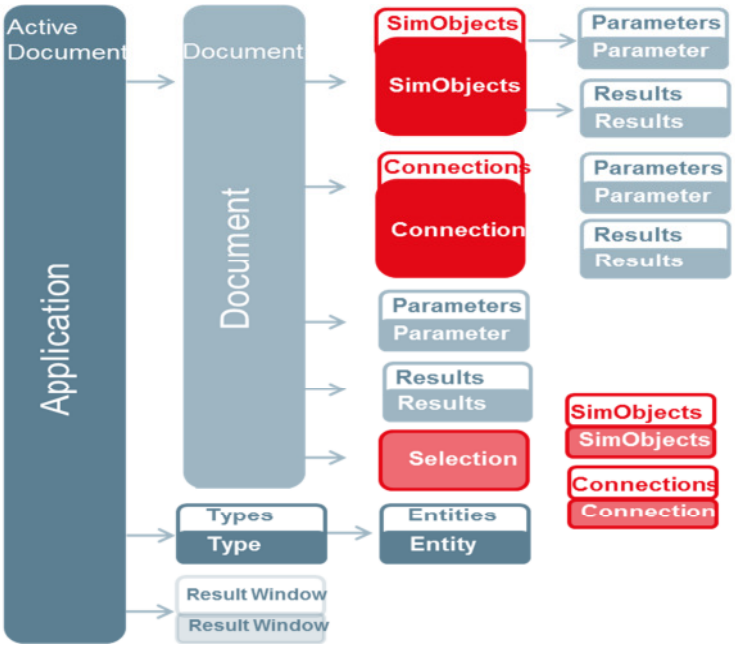

Abbildung 7: Übersicht zum SimulationX-COM-Interface

4 Referenzen

FIEDLER, M.: *Modellbildung und numerische Optimierung am Beispiel eines servopneumatischen Membranzylinderantriebs*. TU Dresden, Dissertation 2010

JANSCHEK, K.: *Systementwurf mechatronischer Systeme*, Springer, 2009, S. 59 ff.

MODELICA ASSOCIATION.: URL www.modelica.org

SIMULATIONX, ITI GMBH, URL www.simulationx.com