9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization
4-6 September 2002, Atlanta, Georgia

AIAA 2002-5637

COORDINATION SPECIFICATION OF THE ANALYTICAL TARGET CASCADING PROCESS USING THE χ LANGUAGE

L.F.P. Etman
Eindhoven University of Technology, Eindhoven, The Netherlands
and
M. Kokkolaras and P.Y. Papalambros
University of Michigan, Ann Arbor, Michigan
and
A.T. Hofkamp and J.E. Rooda
Eindhoven University of Technology, Eindhoven, The Netherlands

ABSTRACT

Optimal design problems of large-scale and complex engineering systems are typically decomposed into a number of smaller and tractable subproblems. Analytical target cascading (ATC) is a methodology for translating overall system design targets to individual specifications for the subsystems and components that make up the system based on a hierarchical partition. We propose to use the χ language and software tools to specify and implement the coordination of the analytical target cascading process. ATC is implemented as parallel processes that exchange data via channels, which represent the links between the subproblems. The process specifications define how individual processes communicate with other coupled processes. We show the advantages of χ for coordinating the ATC process by means of an illustrative example, and demonstrate that different coordination strategies can be implemented and evaluated efficiently.

1 INTRODUCTION

A key challenge in early product development stages of complex products is to propagate desirable product characteristics, defined by product's specifications (or targets), to the various subsystems and components in a consistent and efficient manner. Consistency means that all parts of the designed system should end up working well together, while efficiency means that iterations at later product development stages, which are costly in time and resources, should be avoided.

Analytical target cascading is a methodology for the design of large engineering systems at the early product development stages (Kim, 2001). First, the design problem is partitioned into a hierarchical set of subproblems associated with systems, subsystems, and components. Design specifications (or targets) are defined at the top level of the multilevel design formulation and "cascaded down" to lower levels. Design subproblems are formulated at each level so that components, subsystems, and systems are designed to match the cascaded targets consistent with the overall system targets. The main benefits of target cascading are reduction in design-cycle time, avoidance of design iterations late in the development process, and increased likelihood that physical prototypes will be closer to production quality. Target cascading also facilitates concurrency in system design, allowing the outsourcing of subsystems and components to suppliers. Target cascading offers a robust framework for multilevel design and has been demonstrated (Michelena *et al.*, 2002) to be convergent under standard convexity and smoothness assumptions, whereas other similar problem formulations exhibit convergence difficulties.

The ATC process has been applied successfully to a wide range of vehicle design applications (Kim *et al.*, 2000, Kim *et al.*, 2001, Michelena *et al.*, 2001, Kim *et al.*, 2002). It has also been extended to the design of product families (Kokkolaras *et al.*, 2002). The key to the success of the ATC process lies in coordinating the solution process of the subproblems. Several coordination strategies are presented in (Michelena *et al.*, 2002). They recursively solve one subset of two coupled subproblems at a time. Although they do not address this, their strategies allow the concurrent solution of subproblems (e.g., at the same level of the hierarchy). For larger decompositions the implementation of the ATC coordination may become complicated and prone to errors, even if one refrains from the parallel solutions of the subproblems.

A more precise description of the ATC coordination, especially in regard to concurrency, would be advantageous. Etman *et al.* (2002) suggest to use a programming language founded on concurrency theory for this purpose. They adopt the χ specification lan-

guage (Kleijn and Rooda, 2001; Rooda, 2000) that is based on concepts of Communicating Sequential Processes (CSP) (Hoare, 1985). They show that such a language enables to model and implement the coordination of coupled subproblems in multidisciplinary optimization (MDO) including the possibly parallel behavior of the subproblems.

This paper proposes the use of χ to specify and implement the coordination process of ATC. Our intention is to demonstrate the advantages of χ by evaluating the performance of alternative coordination strategies for a multilevel target cascading problem. The paper is organized as follows. The mathematical formulation of the analytical target cascading process is given in the next section. The coordination specification of ATC is put into the χ perspective in Section 3. An illustrative example is presented in Section 4, where different coordination strategies are implemented by means of χ and evaluated. Conclusions are drawn in Section 5.

2 TARGET CASCADING FORMULATION

The analytical target cascading process is presented using a general notation, from which the design problem for each element (i.e., system, subsystem, or component) can be recovered as a special case. The formulation allows for design specifications to be introduced not only at the top level for the overall product, but also "locally" to account for individual system, subsystem, and/or component requirements. To represent the hierarchy of the partitioned design problem, the set E_i is defined at each level i, in which all the elements of the level are included. For each element j in the set E_i, the set of children C_{ij} is defined, which includes the elements of the set E_{i+1} that are children of the element. An illustrative example is presented on Figure 1: At level $i = 1$ of the partitioned problem we have $E_1 = \{\mathrm{B}, \mathrm{C}\}$, and for element "B" on that level we have $C_{1\mathrm{B}} = \{\mathrm{D}, \mathrm{E}\}$. There are two types of responses:

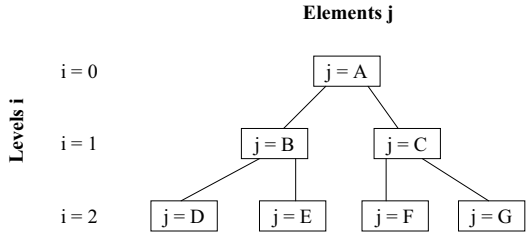

Figure 1: Example of hierarchically partitioned design problem

responses $\tilde{\mathbf{R}}$ linked to "local" targets (e.g., at the top level), and responses $\mathbf{R}$ linked to "cascaded" targets, i.e., linking two successive levels in the problem hierarchy. The design problem P_{ij} corresponding to the j-th element at the i-th level is formulated as follows:

$$
\begin{aligned}
\min_{\bar{\mathbf{x}}_{ij}} \; & \|\tilde{\mathbf{R}}_{ij} - \mathbf{T}_{ij}\| + \|\mathbf{R}_{ij} - \mathbf{R}^U_{ij}\| + \|\mathbf{y}_{ij} - \mathbf{y}^U_{ij}\| \\
& \qquad + \varepsilon^R_{ij} + \varepsilon^y_{ij} \\
\text{subject to} \quad & \sum_{k \in C_{ij}} \|\hat{\mathbf{R}}_{(i+1)k} - \hat{\mathbf{R}}^L_{(i+1)k}\| \le \varepsilon^R_{ij} \\
& \sum_{k \in C_{ij}} \|\mathbf{y}_{(i+1)k} - \mathbf{y}^L_{(i+1)k}\| \le \varepsilon^y_{ij} \qquad (1) \\
& \mathbf{g}_{ij}(\hat{\mathbf{R}}_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij}) \le 0, \\
& \mathbf{h}_{ij}(\hat{\mathbf{R}}_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij}) = 0,
\end{aligned}
$$

where $\hat{\mathbf{R}}_{ij} = \left[\tilde{\mathbf{R}}^t_{ij}, \mathbf{R}^t_{ij}\right]^t = \mathbf{r}_{ij}(\hat{\mathbf{R}}_{(i+1)k_1}, \ldots, \hat{\mathbf{R}}_{(i+1)k_{c_{ij}}}, \mathbf{x}_{ij}, \mathbf{y}_{ij})$, $C_{ij} = \{k_1, \ldots k_{c_{ij}}\}$, and c_{ij} is the number of child elements. Note that an element's response depends on the element's design variables as well as on its children's responses. In the above problem formulation,

- $\bar{\mathbf{x}}_{ij} = \left[\mathbf{x}^t_{ij}, \mathbf{y}^t_{ij}, \mathbf{y}^t_{(i+1)k_1}, \ldots, \mathbf{y}^t_{(i+1)k_{c_{ij}}}, \hat{\mathbf{R}}^t_{(i+1)k_1}, \ldots, \hat{\mathbf{R}}^t_{(i+1)k_{c_{ij}}}, \varepsilon^R_{ij}, \varepsilon^y_{ij}\right]^t$ is the vector of all optimization variables,
- $\mathbf{x}_{ij}$ is the vector of local design variables, that is, variables exclusively associated with the element,
- $\mathbf{y}_{ij}$ is the vector of linking design variables, that is, variables associated with two or more elements that share the same parent,
- $\hat{\mathbf{R}}_{ij}$ is the vector of all responses,
- ε^R_{ij} is the tolerance optimization variable for coordinating the responses of the children of the element,
- ε^y_{ij} is the tolerance optimization variable for coordinating the linking design variables of the children of the element,
- $\mathbf{T}_{ij}$ is the vector of local target values,
- $\mathbf{R}^U_{ij}$ is the vector of response values cascaded down to the element from its parent,
- $\mathbf{y}^U_{ij}$ is the vector of linking design variable values cascaded down to the element from its parent,
- $\hat{\mathbf{R}}^L_{(i+1)k}$ is the vector of response values cascaded up to the element from its k-th child,
- $\mathbf{y}^L_{(i+1)k}$ is the vector of linking design variable values cascaded up to the element from its k-th child,
- $\mathbf{g}_{ij}$ and $\mathbf{h}_{ij}$ are vector functions representing inequality and equality design constraints, respectively.

3 ATC COORDINATION SPECIFICATION

Analytical target cascading requires the iterative solution of the optimization subproblems according to the hierarchical structure of the decomposed problem. In this iterative coordination process the subproblem targets and design variables need to be found such that they are consistent with respect to the top level objectives and the coupling constraints of the subproblems.

Several different iterative coordination strategies may lead a consistent system design. Michelena *et al.* (2002) proved that a class of ATC coordination strategies are guaranteed to converge under standard smoothness and convexity assumptions. However, it is not known a priori which strategy is the most efficient. This may even be problem dependent. In this respect one may also want to distinguish between the number of subproblems that can be solved in parallel. It would be advantageous to have a coordination language with a clear concept of concurrency that enables rapid implementation and investigation of alternative ATC coordination strategies. The purpose of the present paper is to demonstrate this by means of the specification language χ (see Etman *et al.* (2002)).

This section explains how the ATC coordination can be specified as a coupled system of χ processes. A process will be instantiated for each design problem P_{ij}. This process specifies the sequence of statements to exchange data with its parents and children, and to carry out a numerical optimization of subproblem P_{ij}. In the general form of Section 2, such an ATC subproblem process receives target and constraint values from its parent and children, respectively, and returns updates after the associated optimization problem has been solved. The exact process specification for problem P_{ij} depends on the coordination scheme that is employed to solve the overall ATC problem.

3.1 Downwards-only sequential coordination

Consider first a coordination scheme that solves optimization subproblems iteratively in a downwards level-by-level sequence. That is, optimization subproblems are solved subsequently in increasing level order: $i = 0,1,\ldots,N,0,\ldots,N$ etc., where $i = 0$ and $i = N$ represent top and bottom level, respectively.

In the downwards-only coordination, the top-level process C_{top} for problem P_0 (index j is dropped at the top level since there is only one element) starts by carrying out an optimization to determine response and linking variable targets that will be cascaded to its children. At the first iteration, C_{top} uses some user-provided initial guess of the response and linking variable values regularly passed up from its children. After C_{top} has cascaded its targets, it waits until it receives updated values of children response and linking variables. It then carries out a new optimization, compares the new vector of optimization variables to the previous one, examines convergence of the children processes, and either converges or reiterates.

A second type of process is needed for the intermediate-level subproblems P_{ij} that have both a parent and one or more children, i.e., $i \in 1,..,N-1$. For the downwards-only scheme, such an intermediate-level process C_{mid} carries out the following sequence of steps: Receive targets from the parent; carry out an optimization (at the first iteration use an initial guess for the children response and linking values); cascade determined targets to the children; wait until all the children have sent updated response and linking values; and finally pass updated values to the parent.

Thirdly, we need bottom-level processes C_{bot} for the bottom-level subproblems P_{Nj}. A bottom-level process waits until it receives targets from its parent, carries out an optimization, and returns updated values.

The downwards-only coordination scheme for a decomposed problem as depicted in Figure 1 is specified simply by coupling instances of top-level, intermediate-level, and bottom-level processes. No additional process is needed to control the overall coordination. Initially, all processes are waiting to receive target data from their parents, except for the top-level process. The top-level process carries out an optimization and cascades targets down to its children. The intermediate-level processes may carry out their optimizations in parallel; all the other processes are waiting to receive new data. The computed targets are cascaded down. This proceeds until the bottom-level processes have been reached and have carried out their optimizations. The bottom-level processes pass up updated response and linking target values to their parents, which update their parents without carrying out any new optimizations. In this manner, updated target values are rebalanced up level-by-level until the top-level process is finally reached. If convergence has not occurred, a new ATC iteration starts.

Note that the tree of processes does not need to be symmetrical. The same coordination arises if one branch has more levels than another branch. In case the ATC partition consists of just two levels, the coordination specification contains only top-level and bottom-level process instances.

Finally, a repository process R is introduced to facilitate the monitoring of all processes during the ATC iterations. The subproblem processes send updated values of their optimization variables to this process R. These optimization variable values will be stored by R. After completion of the ATC, the complete iteration history is available from R and may be used for further analysis.

3.2 Definition of χ processes

The χ specifications of the processes in the downwards-only sequential coordination scheme are presented below. The following variable types are introduced:

```
type vx      = real^mx                                   (χ-1)
   , vr      = real^mr
   , vy      = real^my
   , vys     = vy^ms
   , vrs     = vr^ms
   , vxbtop  = x.vx
               × ys.vys × rs.vrs
               × epsr.real × epsy.real
   , vxbmid  = x.vx × y.vy
               × ys.vys × rs.vrs
               × epsr.real × epsy.real
   , vxbbot  = x.vx × y.vy
   , par     = wr.real × wy.real
               × ns.nat
               × nx.nat × ny.nat
               × nys.nat × nrs.nat
```

Herein, types vx, vr, and vy denote vector arrays of reals of fixed (maximum) length *mx*, *my*, and *mr*, respectively; χ is a strongly typed language requiring that the dimensions of arrays are known at compilation time (Etman *et al.*, 2002). Types vys and vrs are matrix arrays of sizes $ms \times mr$ and $ms \times my$, respectively. Types vxbtop, vxbmid, and vxbbot are tuples containing two or more data-elements of type vx, vy, vys, vrs, or real. These three tuple types match the optimization variables of the top-level, intermediate-level, and bottom-level design problems, respectively. The elements in the tuples can be accessed through their identifiers defined before the dot operator. Finally, type par is defined as a tuple of reals and nats to store the scaling values used in the optimization problem of the process, the number of children of the process, and the actual array dimensions used in the process.

Using this type definitions, the χ specification of the top-level process C_{top} is:

```
proc Ctop( f : (vxbtop, vr, vrs, vys, par)
                 → vxbtop × vr
         , b : (!vr × vy)^ms
         , c : (?vr × vy × bool)^ms
         , e :!vxbtop, s :!void
         , T : vr, xb0 : vxbtop, p : par
         , tol : real                          ) =
|[ r : vr, rLs : vrs, yLs : vys
 , xb : vxbtop, cvrg : bool, cvrgs : bool^ms
 , i, n : nat
 | cvrg := false; rLs := xb0.rs; yLs := xb0.ys
 ; n := 0
 ; ⟨xb, r⟩ := f(xb0, T, rLs, yLs, p)
 ; *[ ¬cvrg ∧ n < maxiter
    ⟶ n := n + 1
      ; i := 0
      ; *[ i < p.ns                                                    (χ-2)
         ⟶ b.i! ⟨(xb.rs).i, (xb.ys).i⟩
           ; i := i + 1
       ]
      ; i := 0
      ; *[ i < p.ns
         ⟶ c.i? ⟨rLs.i, yLs.i, cvrgs.i⟩
           ; i := i + 1
       ]
      ; ⟨xb, r⟩ := f(xb0, T, rLs, yLs, p)
      ; e!xb
      ; cvrg := topnorm(xb0, xb, p) < tol
      ; cvrg := checkcvrg(cvrg, cvrgs, p.ns)
      ; xb0 := xb
  ]
 ; s!
]|
```

The first element f of the parameter list of C_{top} defines the Matlab function that will carry out the optimization. Parameters *b* and *c* represent the send and receive port arrays, respectively, through which data is sent to and received from the children of C_{top}. The fourth parameter *e* is a send port to repository *R* to update the coordination history; the fifth parameter *s* is a synchronization port to *R* to notify when the ATC coordination has been finished. The last four parameters are the top-level response targets, the initial values of the optimization variables, the subproblem parameter values, and the convergence tolerance, respectively.

The following local variables are introduced in C_{top}: a response vector *r*, matrices *rLs* and *yLs* of response target values and linking variable target values, respectively, that have been passed up from the children, a tuple of optimization variables *xb*, a boolean variable *cvrg*, a boolean array *cvrgs*, and two natural variables *i* and *n*. C_{top} starts by carrying out an initial, one-time only optimization. The following sequence of tasks is then repeated: cascading targets, receiving updated

target values, carrying out optimization, checking convergence, and updating optimization variables.

C_{top} is declared locally converged if the square of the norm of the difference between previous and current iterates is smaller than some predefined value tol, i.e., if $||\overline{\mathbf{x}}^{(n)} - \overline{\mathbf{x}}^{(n-1)}||_2^2 \leq tol$. Note that the vector of optimization variables $\overline{\mathbf{x}}$ has different instantiations for the top, intermediate, and bottom levels. C_{top} is declared globally converged if convergence has occurred for the "local" optimization problem as well as for all the children optimization problems. The function *checkcvrg* returns true if this is the case. To this end, the children pass up their convergence status in addition to the updated response and linking variable values. C_{top} stores the convergence status of its children in the boolean array *cvrgs*. The functions *topnorm* and *checkcvrg* are specified as χ functions. The ATC process is terminated when C_{top} has converged or the maximum number of iterations has been reached. A synchronization is sent to repository process R to acknowledge this.

The intermediate-level process C_{mid} is specified in a similar way:

```
proc Cmid( f : (vxbmid, vr, vy, vrs, vys, par)
                 → vxbmid × vr
         , a :?vr × vy
         , b : (!vr × vy)^ms
         , c : (?vr × vy × bool)^ms
         , d :!vr × vy × bool
         , e :!vxbmid, xb0 : vxbmid, p : par
         , tol : real                          ) =
|[ r, rUS : vr, rLs : vrs, yUS : vy, yLs : vys
 , xb : vxbmid, cvrg : bool, cvrgs : bool^ms, i : nat
 | rLs := xb0.rs; yLs := xb0.ys
 ; *[ true
    ⟶ a? ⟨rUS, yUS⟩
      ; ⟨xb, r⟩ := f(xb0, rUS, yUS, rLs, yLs, p)
      ; e!xb
      ; i := 0
      ; *[ i < p.ns
         ⟶ b.i! ⟨(xb.rs).i, (xb.ys).i⟩
           ; i := i + 1
       ]
      ; i := 0
      ; *[ i < p.ns
         ⟶ c.i? ⟨rLs.i, yLs.i, cvrgs.i⟩
           ; i := i + 1
       ]
      ; cvrg := midnorm(xb0, xb, p) < tol
      ; cvrg := checkcvrg(cvrg, cvrgs, p.ns)
      ; xb0 := xb
      ; d! ⟨r, xb.y, cvrg⟩
 ]
]|
```

(χ-3)

C_{mid} has four communication ports: a and d are receive and sent ports coupled to the parent; b and c are receive and sent port arrays (bundles), the elements of which are coupled to the children through channels. C_{mid} receives response and linking variable targets through port a; cascades targets to its children through ports $b.i$; receives updated response and linking variable values as well as the status of convergence of children-problems through ports $c.i$; and finally passes updated values and convergence status to its parent through port d. This sequence is repeated indefinitely. After C_{mid} has received the target values cascaded down from its parent, it carries out the local optimization defined by function f, using initial optimization variable values $xb0$ and subsystem parameters p. After the convergence status of the children has been received, the local and overall convergence of C_{mid} is determined as described for C_{top}; however, a modified norm function (*midnorm*) is used. T

The bottom-level process C_{bot} is specified next. C_{bot} receives targets through port a, carries out the optimization defined by f, and passes up updated values and convergence status through port d:

```
proc Cbot( f : (vxbbot, vr, vy, par)
                 → vxbbot × vr
         , a :?vr × vy, d :!vr × vy × bool
         , e :!vxbbot
         , xb0 : vxbbot, p : par, tol : real ) =
|[ r, rUS : vr, yUS : vy, xb : vxbbot, cvrg : bool
 | *[ true
    ⟶ a? ⟨rUS, yUS⟩
      ; ⟨xb, r⟩ := f(xb0, rUS, yUS, p)
      ; e!xb
      ; cvrg := botnorm(xb0, xb, p) < tol
      ; xb0 := xb
      ; d! ⟨r, xb.y, cvrg⟩
 ]
]|
```

(χ-4)

Finally, there is a repository process R that collects optimization results every time a problem is solved, and stores them in lists:

```
proc R(t :?vxbtop
       ,m : (?vxbmid)^nm, b : (?vxbbot)^nb
       ,s :?void, tol, sca : real            ) =
|[ vt : vxbtop, td : vxbtop*
 , vm : vxbmid, md : (vxbmid*)^nm
 , vb : vxbbot, bd : (vxbbot*)^nb, b : bool
 | td := [] ; md := ini_mid(); bd := ini_bot()
 ; *[ true; t?vt
      --> td := td ++ [vt]                    (χ-5)
    [] j : nat <- 0..nm : true; m.j?vm
      --> md.j := md.j ++ [vm]
    [] j : nat <- 0..nb : true; b.j?vb
      --> bd.j := bd.j ++ [vb]
    [] true; s?
      --> b := pp(td, md, bd, tol, sca)
  ]
]|
```

R has ports to all subproblem processes in the coordination. R has three different port parameters since the optimization variables tuple differs for the top-level, intermediate-level, and bottom-level processes, respectively. Additionally we have a synchronization port to the top-level proces for the acknowledgement of the ATC finish. The key statement of R is a repetitive selective waiting statement. The repetitive selective waiting statement waits until a new iteration update is received from one of the subproblem processes or until a synchronization is received from the top-level process. R stores the iteration updates of the subproblems in seperate lists designated to each of the processes. If the synchronization communication is carried out, the lists of the subproblems are post-processed by function pp for inspection of the ATC iteration history.

3.3 Alternative coordination schemes

Additional coordination schemes can be obtained by modifying slightly the specifications of the subproblem processes. For example, a downwards-upwards sequential coordination scheme ($i = 0,1,\ldots,N-1,N,N-1,\ldots,1,0$, etc.) is obtained by simply inserting the line

$$;\langle xb, r\rangle := f(xb0, rUS, yUS, rLs, yLs, p)$$

after the statement

```
;*[ i < p.ns
    --> c.i? ⟨rLs.i, yLs.i, cvrgs.i⟩
        ; i := i + 1
 ]
```

in the C_{mid} process specification (χ-3). By doing this, intermediate-level processes carry out an additional optimization every time updated values are rebalanced up.

Michelena *et al.* (2002) consider nested coordination schemes. For example, the nested coordination that corresponds to the scheme depicted in Figure 2, is

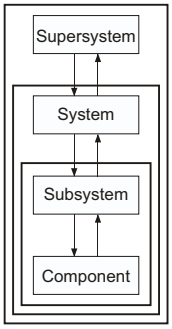

Figure 2: Top-down nested coordination.

obtained by simply inserting an iteration loop between the receive and send statements with respect to the parent-related communication. In this manner, communications with the children are nested with respect to communications with the parent. The repetition statement is inserted into (χ-3) as follows.

```
;*[ true
   --> a? ⟨rUS, yUS⟩
       ;*[ ¬cvrg ∧ n < maxiter
           --> n := n + 1
               ; ⟨xb, r⟩ :=
                     f(xb0, rUS, yUS, rLs, yLs, p)
               ...
               ; xb0 := xb
         ]
       ; d! ⟨r, xb.y, cvrg⟩
 ]
```

4 EXAMPLE

We will now demonstrate the implementation of different coordination strategies using the χ language on a simple but illustrative analytical optimization problem.

4.1 Mathematical formulation of example problem

Our example is based on the geometric programming problem presented in Kim (2001):

$$\begin{aligned}
&\min_{\mathbf{z}\geq\mathbf{0}} \quad z_1^2+z_2^2\\
\text{s.t.}\quad &\frac{z_3^{-2}+z_4^2}{z_5^2}-1\leq 0\ ;\ \frac{z_5^2+z_6^{-2}}{z_7^2}-1\leq 0\\
&\frac{z_8^2+z_9^2}{z_{11}^2}-1\leq 0\ ;\ \frac{z_8^{-2}+z_{10}^2}{z_{11}^2}-1\leq 0\\
&\frac{z_{11}^2+z_{12}^{-2}}{z_{13}^2}-1\leq 0\ ;\ \frac{z_{11}^2+z_{12}^2}{z_{14}^2}-1\leq 0\\
&z_1^2-z_3^2-z_4^{-2}-z_5^2=0\ ;\ z_2^2-z_5^2-z_6^2-z_7^2=0\\
&z_3^2-z_8^2-z_9^{-2}-z_{10}^{-2}-z_{11}^2=0\\
&z_6^2-z_{11}^2-z_{12}^2-z_{13}^2-z_{14}^2=0
\end{aligned}\qquad (2)$$

The solution of problem (2) is known to be equal to $\mathbf{z}$ = [2.84 3.09 2.36 0.76 0.87 2.81 0.94 0.97 0.87 0.80 1.30 0.84 1.76 1.55]. Kim (2001) decomposed this problem using the equality constraints as responses within a bi-level hierarchical structure and demonstrated the application of the ATC process. Here, we decompose the original problem into three levels as shown in Figure 3; z_5 is the linking variable that couples the subproblems of the intermediate level. Note that z_{11} is a linking variable coupling the two problems of the bottom level. The ATC formulation does not allow subproblems to share variables unless they are children of the same parent. Since the purpose of this example is to illustrate the χ implementation of alternative coordination strategies, we treat z_{11} as a parameter using its known optimal value.

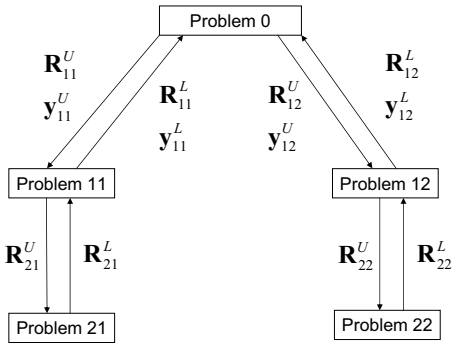

Figure 3: Hierarchical structure of decomposed problem

The subproblems are formulated in the next subsections following the notation presented in Section 2; the index j is dropped for the top-level problem since there is only one element.

Top-level problem Problem P_0 is formulated as

$$\begin{aligned}
&\min_{\mathbf{R}_{11},\mathbf{R}_{12},\mathbf{y}_0,\varepsilon_0^y,\varepsilon_0^R} \quad \|\mathbf{R}_0-\mathbf{T}_0\|+\varepsilon_0^y+\varepsilon_0^R\\
&\text{subject to}\quad \|\mathbf{y}_0-\mathbf{y}_{11}^L\|+\|\mathbf{y}_0-\mathbf{y}_{12}^L\|\leq\varepsilon_0^y\\
&\qquad\qquad \|\mathbf{R}_{11}-\mathbf{R}_{11}^L\|+\|\mathbf{R}_{12}-\mathbf{R}_{12}^L\|\leq\varepsilon_0^R,
\end{aligned}\qquad (3)$$

where $\mathbf{R}_{11}:=z_1$, $\mathbf{R}_{12}:=z_2$, $\mathbf{y}_0:=z_5$, $\mathbf{R}_0=r_0(\mathbf{R}_{11},\mathbf{R}_{12})=z_1^2+z_2^2$, and $\mathbf{T}_0=0$. Note that z_1, z_2, and z_5, correspond to the formulation of the original problem, and that z_5 is a linking variable computed at the problems of the intermediate level and coordinated at the top level.

Intermediate-level problems There are two problems at the intermediate level. Problem P_{11} is formulated as

$$\begin{aligned}
&\min_{\mathbf{R}_{21},\mathbf{y}_{11},\mathbf{x}_{11},\varepsilon_{11}^R} \quad \|\mathbf{R}_{11}-\mathbf{R}_{11}^U\|+\|\mathbf{y}_{11}-\mathbf{y}_0^U\|+\varepsilon_{11}^R\\
&\text{subject to}\quad \|\mathbf{R}_{21}-\mathbf{R}_{21}^L\|\leq\varepsilon_{11}^R,\\
&\qquad\qquad \mathbf{g}_{11}(\mathbf{R}_{21},\mathbf{x}_{11},\mathbf{y}_{11})\leq\mathbf{0}
\end{aligned}\qquad (4)$$

where $\mathbf{R}_{21}:=z_3$, $\mathbf{x}_{11}:=z_4$, $\mathbf{y}_{11}:=z_5$, $\mathbf{R}_{11}=r_{11}(\mathbf{R}_{21},\mathbf{x}_{11},\mathbf{y}_{11})=\sqrt{z_3^2+z_4^{-2}+z_5^2}$, and $\mathbf{g}_{11}(\mathbf{R}_{21},\mathbf{x}_{11},\mathbf{y}_{11})=(z_3^{-2}+z_4^2)z_5^{-2}-1$. Problem P_{12} is stated as

$$\begin{aligned}
&\min_{\mathbf{R}_{22},\mathbf{y}_{12},\mathbf{x}_{12},\varepsilon_{12}^R} \quad \|\mathbf{R}_{12}-\mathbf{R}_{12}^U\|+\|\mathbf{y}_{12}-\mathbf{y}_0^U\|+\varepsilon_{12}^R\\
&\text{subject to}\quad \|\mathbf{R}_{22}-\mathbf{R}_{22}^L\|\leq\varepsilon_{11}^R,\\
&\qquad\qquad \mathbf{g}_{12}(\mathbf{R}_{22},\mathbf{x}_{12},\mathbf{y}_{12})\leq\mathbf{0}
\end{aligned}\qquad (5)$$

where $\mathbf{R}_{22}:=z_6$, $\mathbf{x}_{12}:=z_7$, $\mathbf{y}_{12}:=z_5$, $\mathbf{R}_{12}=r_{12}(\mathbf{R}_{22},\mathbf{x}_{12},\mathbf{y}_{12})=\sqrt{z_5^2+z_6^2+z_7^2}$, and $\mathbf{g}_{12}(\mathbf{R}_{22},\mathbf{x}_{12},\mathbf{y}_{12})=(z_5^2+z_6^{-2})z_7^{-2}-1$.

Bottom-level problems There are two problems at the bottom level. Problem P_{21} is given by

$$\begin{aligned}
&\min_{\mathbf{x}_{21}} \quad \|\mathbf{R}_{21}-\mathbf{R}_{21}^U\|\\
&\text{subject to}\quad \mathbf{g}_{21}(\mathbf{x}_{21})\leq\mathbf{0}
\end{aligned}\qquad (6)$$

where $\mathbf{x}_{21}:=[z_8,z_9,z_{10}]$, $p=1.3(=z_{11})$, $\mathbf{R}_{21}=r_{21}(\mathbf{x}_{21})=\sqrt{z_8^2+z_9^{-2}+z_{10}^{-2}+p^2}$, and

$$\mathbf{g}_{21}(\mathbf{x}_{21})=\begin{bmatrix}(z_8^2+z_9^2)p^{-2}-1\\(z_8^{-2}+z_{10}^2)p^{-2}-1\end{bmatrix}.$$

The formulation of problem P_{22} is

$$\begin{aligned}
&\min_{\mathbf{x}_{22}} \quad \|\mathbf{R}_{22}-\mathbf{R}_{22}^U\|\\
&\text{subject to}\quad \mathbf{g}_{22}(\mathbf{x}_{22})\leq\mathbf{0}
\end{aligned}\qquad (7)$$

where $\mathbf{x}_{22} := [z_{12}, z_{13}, z_{14}]$, $p = 1.3 (= z_{11})$, $\mathbf{R}_{22} = r_{22}(\mathbf{x}_{22}) = \sqrt{z_{12}^2 + z_{13}^2 + z_{14}^2 + p^2}$, and

$$\mathbf{g}_{22}(\mathbf{x}_{22}) = \begin{bmatrix} (p^2 + z_{12}^{-2}) z_{13}^{-2} - 1 \\ (p^2 + z_{12}^2) z_{14}^{-2} - 1 \end{bmatrix}.$$

4.2 Implementation of coordination using χ

The χ implementation of the three-level ATC example requires a number of processes as specified in Section 3. We assign the following constant values first:

```
const mx     : nat  = 3                                   (χ-6)
    , mr     : nat  = 1
    , my     : nat  = 1
    , ms     : nat  = 2
    , nm     : nat  = 2
    , nb     : nat  = 2
    , maxiter: nat  = 1000
    , nan    : real = −9.9e99
```

where *mx* is the maximum length of design variable arrays (there are maximum three local design variables in any of the subproblems), *mr* is the maximum length of response variable arrays, *my* is the maximum length of linking variable arrays, *ms* is the maximum number of children in any subproblem (2 in the top-level problem), *nm* is the number of intermediate-level processes (P_{11} and P_{12}), and *nb* is the number of bottom-level processes (P_{21} and P_{22}). Moreover, we define the maximum number of iterations *maxiter* and a "not-a-number" value *nan* that is used for empty entries in the fixed-length arrays.

The system that couples the processes is instantiated according to Figure 4. The top-level problem

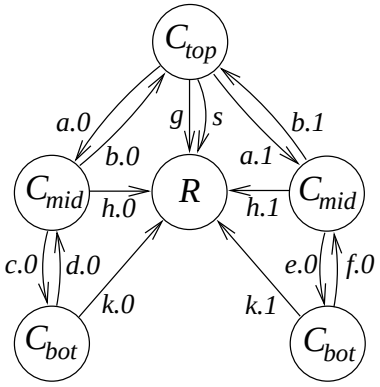

Figure 4: Processes and channels in the three-level ATC example.

P_0 requires a process instantiation of C_{top}. Each of the two children of P_0 requires an instantiation of the intermediate-level process C_{mid}, representing problems P_{11} and P_{12}. The bottom-level subproblems P_{21} and P_{22} are represented by two process instantiations of C_{bot}. Finally, repository process R needs to be instantiated.

These six processes have to be coupled to each other by channels of appropriate data type. The channel names are defined in Figure 4. For example, C_{top} sends data of type vr × vy to each of its children via channel array *a*, and receives data of type vr × vy × bool from its children via channel array *b*. C_{top} shares two additional synchronization send channels with repository *R*: one to send iteration updates, and one to acknowledge the completion of the ATC coordination. The intermediate-level process C_{mid} representing design problem P_{11} receives data of type vr × vy from C_{top} via channel $a.0$ and sends data of type vr × vy × bool via channel $b.0$. The C_{mid} process representing P_{11} also communicates through channels $c.0$ and $d.0$ with the C_{bot} process that represents its child P_{21}. Problems P_{12} and P_{22} are represented in a similar fashion. The two C_{mid} processes and the two C_{bot} processes send updates to repository *R* through channel arrays *h* and *k*, respectively.

Note that channels *a* to *f* are channel arrays of size $ns = 2$. However, P_{11} has only one child, and therefore only the first elements of the channel arrays *c* and *d* are used, i.e. $c.0$ and $d.0$. The same holds for P_{12} and channel arrays *e* and *f*. The constant *ns* cannot be defined as a separate parameter in the process definition; χ is a strongly typed language and does not allow (channel) array type specifications of variable size. At present, χ also does not allow to specify the actual array sizes used in a process upon instantiation of the process in a system.

The subproblem optimizations are carried out using the Matlab optimization toolbox function `fmincon`, which is an implementation of the sequential quadratic programming (SQP) algorithm. We use the square of the l_2-norm as the metric of the deviation terms in the objective and constraints of Problem (1). The function calls are realized by means of a χ-python interface (Hofkamp, 2001), developed such that χ can call Python functions like native χ functions. Python is a scripting language designed to glue pieces of software to each other. This interface can be used to link χ with any other desired software package. In our case, we have used pymat to couple χ through Python with Matlab (see Etman *et al.*, 2002).

The system specification is given below:

$$
\begin{aligned}
&\mathsf{syst}\ S(tol, sca : \text{real}) = \\
&|[a, c, e : (-\text{vr} \times \text{vy})^{ms} \\
&, b, d, f : (-\text{vr} \times \text{vy} \times \text{bool})^{ms} \\
&, g : -\text{vxbtop}, h : (-\text{vxbmid})^{nm} \\
&, k : (-\text{vxbbot})^{nb}, s : -\text{void} \\
&| \, Ctop(\text{P0}, a, b, g, s, \langle 0.0 \rangle \\
&\qquad , \langle \langle nan, nan, nan \rangle \\
&\qquad\ , \langle \langle 1.0 \rangle , \langle 1.0 \rangle \rangle \\
&\qquad\ , \langle \langle 1.0 \rangle , \langle 1.0 \rangle \rangle , 0.0, 0.0 \rangle \\
&\qquad , \langle sca, sca, 2, 0, 0, 1, 1 \rangle , tol) \\
&|| \, Cmid(\text{P11}, a.0, c, d, b.0, h.0 \\
&\qquad , \langle \langle 1.0, nan, nan \rangle , \langle 1.0 \rangle \\
&\qquad\ , \langle \langle nan \rangle , \langle nan \rangle \rangle \\
&\qquad\ , \langle \langle 1.0 \rangle , \langle nan \rangle \rangle , 0.0, 0.0 \rangle \\
&\qquad , \langle sca, sca, 1, 1, 1, 0, 1 \rangle , tol) \qquad (\chi\text{-}7) \\
&|| \, Cmid(\text{P12}, a.1, e, f, b.1, h.1 \\
&\qquad , \langle \langle 1.0, nan, nan \rangle , \langle 1.0 \rangle \\
&\qquad\ , \langle \langle nan \rangle , \langle nan \rangle \rangle \\
&\qquad\ , \langle \langle 1.0 \rangle , \langle nan \rangle \rangle , 0.0, 0.0 \rangle \\
&\qquad , \langle sca, sca, 1, 1, 1, 0, 1 \rangle , tol) \\
&|| \, Cbot(\text{P21}, c.0, d.0, k.0 \\
&\qquad , \langle \langle 1.0, 1.0, 1.0 \rangle , \langle nan \rangle \rangle \\
&\qquad , \langle nan, nan, 0, 3, 0, 0, 0 \rangle , tol) \\
&|| \, Cbot(\text{P22}, e.0, f.0, k.1 \\
&\qquad , \langle \langle 1.0, 1.0, 1.0 \rangle , \langle nan \rangle \rangle \\
&\qquad , \langle nan, nan, 0, 3, 0, 0, 0 \rangle , tol) \\
&|| \, R(g, h, k, s, tol, sca) \\
&]|
\end{aligned}
$$

All processes in system S are instantiated with the appropriate channels and parameters. All optimization variables in the tuple $xb0$ of processes C_{top}, C_{mid}, and C_{bot} are given an initial value of one, except for the ε_{ij}^{R} and ε_{ij}^{y} variables which are initialized at zero. The parameter array p contains the scaling parameter value w for the ε terms in the objective function of problems P_0, P_{11}, and P_{12}, and the number of children, local design variables, linking variables, and children response and linking variable target values (assumed equal for all children) for each process C_{top}, C_{mid}, and C_{bot} . The χ specification ends with:

$$
\mathsf{xper}\ (tol, sca : \text{real}) = |[S(tol, sca)]| \qquad (\chi\text{-}8)
$$

denoting that after compilation a system execution of S can be carried with tolerance tol and scaling w as input.

4.3 Results

We have implemented three coordination schemes for our three-level target cascading problem. Denoting top, intermediate, and bottom levels with 0, 1, and 2, respectively, the sequences are

Scheme I (downwards only): $0 \to 1 \to 2 \to 0$

Scheme II (downwards/upwards): $0 \to 1 \to 2 \to 1 \to 0$

Scheme III (nested): $0 \to 1 \to 2 \to 1 \to 2 \ldots 0$

It has been observed that scaling the ε terms in the objective of problems 3 - 5 has an effect on the convergence behavior of the ATC process. Therefore, numerical experiments have been performed with different values for the termination criterion tolerance and scaling parameter *sca* (same for all ε terms). The results are summarized in Tables 1 - 3. There are two rows for each scaling parameter value; in the top row we report accuracy of obtained results, represented by the square of the l_2-norm of the difference between obtained and exact solution. In the bottom row, and for schemes I and II, we report the number of required target cascading iterations. The computational cost can then be determined for each scheme according to the following considerations: For scheme I, the top-level problem is solved $n+1$ times, where n is the number of target cascading iterations, and each intermediate-level and bottom-level problem is solved n times. Since intermediate-level and bottom-level problems are solved in parallel, the total computational cost in terms of solved optimization problems is therefore $n+1+n+n = 3n+1$. For scheme II, the top level problem is solved $n+1$ times, each intermediate-level problem is solved $2n$ times, and each bottom-level problem is solved n times; total cost equals to $4n+1$. For scheme III, the bottom row includes three numbers: starting from the left, these are the number of times n_t that the top-level problem has been solved, and the numbers of times n_l and n_r that the left and right intermediate-level and bottom-level problems have been solved, respectively. Note that for each side of the tree (left and right), the number of intermediate-level problems is equal to the number of solved bottom-level problems. The total cost is then given by $n_t + 2max(n_l, n_r)$.

The variation of accuracy and computational cost (as defined above) for each coordination scheme is depicted in Figures 5 and 6, respectively, for the termination tolerance value of 10^{-6}. These plots look quite similar for $tol = 10^{-8}$. However, for $tol = 10^{-10}$, the behavior is quite different, as can be seen in Figures 7 and 8.

Although this example is quite simple and serves , we can draw some interesting conclusions: For low tolerance values, the scaling parameter yielding the highest accuracy differs between scheme I (1e4) and schemes II and III (1e6). For high tolerance values, the scaling parameter yielding the highest accuracy is the same for all schemes (1e4). The accuracy of the obtained results does not improve dramatically as the

Table 1: Results for coordination scheme I

sca	Termination tolerance		
	1e-6	1e-8	1e-10
1e2	0.87526 9	0.87532 10	0.87565 17
1e3	0.03840 20	0.03773 31	0.03720 98
1e4	0.00597 45	0.00196 129	0.00151 295
1e5	0.02541 8	0.02486 16	0.02442 76
1e6	0.02602 8	0.02597 12	0.02596 15
1e7	0.02567 9	0.02564 11	0.02563 14
1e8	0.02595 9	0.02595 10	0.02592 16

Table 2: Results for coordination scheme II

sca	Termination tolerance		
	1e-6	1e-8	1e-10
1e2	0.87513 6	0.87543 9	0.87541 15
1e3	0.03825 21	0.03772 32	0.03725 149
1e4	0.03204 6	0.03204 6	0.00150 313
1e5	0.03371 9	0.03326 17	0.03246 70
1e6	0.00863 11	0.00839 16	0.00838 20
1e7	0.01105 7	0.00849 18	0.00849 20
1e8	0.04686 3	0.04686 3	0.00870 18

Table 3: Results for coordination scheme III

sca	Termination tolerance		
	1e-6	1e-8	1e-10
1e2	0.87507 7 11 12	0.87559 11 20 23	0.87552 12 45 46
1e3	0.03825 22 41 42	0.03798 26 51 49	0.03780 27 105 56
1e4	0.03205 7 12 12	0.03205 49 191 97	0.00712 6 11 12
1e5	0.01770 6 11 12	0.00932 10 12 22	0.03418 14 92 39
1e6	0.00998 11 13 20	0.00839 18 20 36	0.03131 6 185 17
1e7	0.03014 9 14 17	0.03102 13 33 25	0.03394 14 121 46
1e8	0.04686 4 6 6	0.04686 4 6 6	0.02650 6 46 17

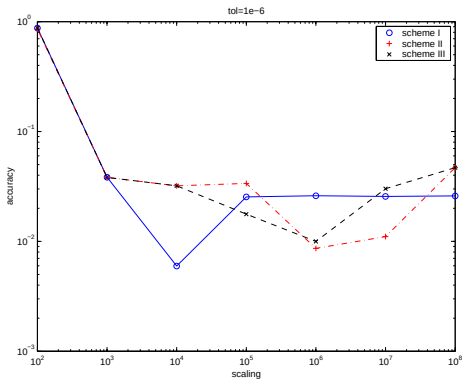

Figure 5: Accuracy of results obtained for $tol = 10^{-6}$

tolerance value increases for all three schemes, especially considering the substantial increase in computational cost. Overall, although it seems that scheme I provides the most accurate results, scheme III is much more efficient.

5 Conclusions

The key to the success of the ATC process lies in the coordination. It is advantageous to have a language that enables precise and compact specification of the ATC coordination. The χ language has been used to specify and implement alternative coordination strategies of the ATC process. Top-level, intermediate-level, and bottom-level processes have been identified for this purpose. Once a specific coordination strategy was implemented, only limited modifications (confined to the intermediate-level process) were required to define additional coordinations. Moreover, larger problems can be treated readily by simply adding necessary instantiations of the defined processes. The example demonstrated that the χ language is well-suited

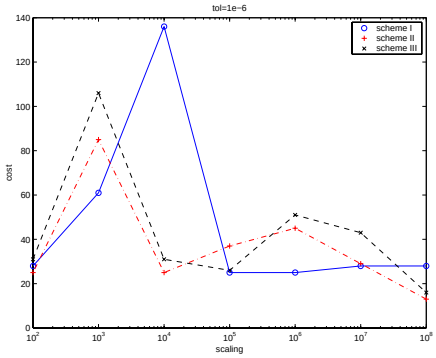

Figure 6: Computational cost for $tol = 10^{-6}$

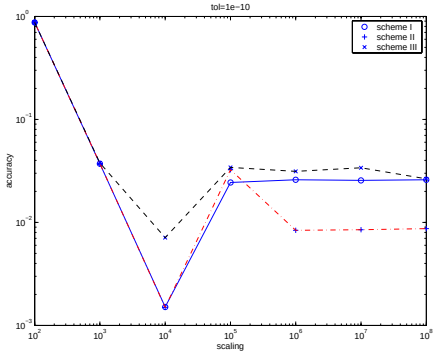

Figure 7: Accuracy of results obtained for $tol = 10^{-10}$

to specify and implement the ATC coordination, and enables rapid investigation of coordination strategies. The reader should have in mind that the usefulness of χ has been utilized in a small degree only due to the simplicity of the example. The specification of ATC as parallel processes may lead to new ways of thinking in regard to ATC coordination schemes.

REFERENCES

Kim, H.M., Michelena, N.F., Papalambros, P.Y., and Jiang, T., 2000, "Target cascading in optimal system design". *Proceedings of the 26th Design Automation Conference*, Baltimore, MD, paper no. DAC-14265.

Kim, H.M., 2001, *Target Cascading in Optimal System Design*, PhD thesis, The University of Michigan, Ann Arbor, Michigan.

Kim, H.M., Michelena, N.F., Papalambros, P.Y., and Jiang, T., 2001, "Analytical target cascading in automotive vehicle design". *Proceedings of the 27th Design Automation Conference*, Pittsburgh, PA, paper no. DAC-21079.

Kim, H.M., Kokkolaras, M., Louca, L.S., Delagrammatikas, G.J., Michelena, N.F., Filipi, Z.S., Papalambros, P.Y., Stein, J.L., and Assanis, D.N., 2002, "Target cascading in vehicle redesign: A class VI truck study". *International Journal of Vehicle Design*, 29(3):1-27.

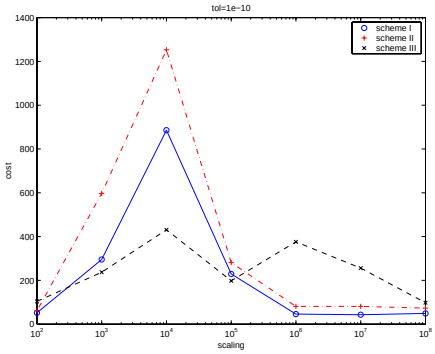

Figure 8: Computational cost for $tol = 10^{-10}$

Kokkolaras, M., Fellini, R., Kim, H.M., Michelena, N.F., and Papalambros, P.Y., 2002, "Extension of the target cascading formulation to the design of product families", *Journal of Structural and Multidisciplinary Optimization*, to appear.

Michelena, N.F., Kokkolaras, M., Louca, L.S., Lin C.C., Jung D., Filipi, Z.S., Assanis, D.N., Papalambros, P.Y., Peng, H, Stein, J.L., and Feury, M., 2001, "Design of an advanced heavy tactical truck: a target cascading case study ", *SAE International Truck & Bus Meeting and Exhibition*, Chicago, IL, paper no. 2001-01-2793.

Michelena, N., and Park, H., and Papalambros, P., 2002, "Convergence properties of analytical target cascading", *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, paper no. AIAA-2002-5506.

Etman, L.F.P., Hofkamp, A.T., Rooda, J.E., Kokkolaras, M., and Papalambros, P.Y. 2002, "Coordination specification for distributed optimal system design", *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, paper no. AIAA-2002-5410.

Hoare, C.A.R., 1985: *Communicating Sequential Processes*, Prentice-Hall.

Kleijn, J.J.T., and Rooda. J.E. 2001, "χ Manual", Systems Engineering Group, Eindhoven University of Technology, http://se.wtb.tue.nl.

Rooda. J.E. 2000, "Modelling Industrial Systems", lecture notes, Systems Engineering Group, Eindhoven University of Technology, http://se.wtb.tue.nl.

Hofkamp, A.T. 2001: "Python from χ", note, Systems Engineering Group, Eindhoven University of Technology, http://se.wtb.tue.nl.