Int J Adv Manuf Technol
DOI 10.1007/s00170-005-0064-3

ORIGINAL ARTICLE

George Q. Huang · T. Qu · David W. L. Cheung · L. Liang

Extensible multi-agent system for optimal design of complex systems using analytical target cascading

Received: 1 September 2004 / Accepted: 15 November 2004

Abstract Analytical target cascading (ATC) has emerged as an integrated approach and methodology for optimal design of complex hierarchical systems with good convergence. It facilitates collaborative design problem solving and enables distributed computing to extend both the capability and capacity. ATC is very powerful yet requiring comprehensive understanding and efforts in setting up specific projects. This paper presents a generic and extensible information infrastructure called atcPortal in the form of a web portal where the ATC analyst defines the ATC problem using a special-purpose XML-based language called atcXML, conducts the ATC analysis, and obtains the analytical results. atcPortal not only reduces the overheads for analysts to set up the projects for practical applications, but also allows the researchers to experiment and extend the ATC methods under different system structures and/or different ATC strategies.

Keywords Analytical target cascading · Target optimization · Web portal · Web services · XML · Hierarchical system modeling

1 Introduction

Analytical target cascading (ATC) has recently emerged as a new approach to solving target optimization problems in complex engineering systems which can be represented hierarchically in terms of constituent components and modeled mathematically in terms of decision variables [6]. The formulation addresses the problem of propagating design targets through a hierarchically partitioned product development process, so that the local targets are consistent with each other and with the overall top level targets. According to [6], ATC includes four key steps: (i) development of appropriate models, (ii) partitioning the system, (ii) formulating the target cascading problems for each element of the partition, and (iv) solving the partitioned problem through a coordination strategy to compute all stated targets. The researchers who originally proposed the ATC approach at the University of Michigan have studied and demonstrated the convergence with theoretical rigor [12] and through numerous industrial case studies (mainly in the field of vehicle design) [9, 5, 13]. Researchers in the same group have also applied ATC for building design [1]. The framework has been applied for thermal design and analysis of a fictional three-zone building. The building is decomposed into two levels. Different analysis tools are employed for computing the performance targets at each decomposed level of the building. Interestingly, most of the publications on ATC are somewhat related to the projects and the people who are working or have worked at the University of Michigan [e.g. 10, 11, 13] with a few exceptions [e.g. 4].

ATC is a powerful method with several important characteristics. Firstly, its convergence has been demonstrated, both theoretically and through case studies. It has been reported that the top-level component normally converges after about ten iterations [7, 8]. Several case studies have compared results from ATC and ATO (all at once) and reported that the results are generally consistent with each other. This indicates that ATC is at least as effective as ATO in solving the optimal system design problems. The efficiency of convergence has been improved by much reduced complexity of sub-problems which converge much faster than when they are treated as one big system. Meanwhile, the solution spaces are narrowed progressively after each cycle of cascading and backtracking (rebalancing). This also speeds up the convergence.

G. Q. Huang (✉) · T. Qu
Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong, People's Republic of China
e-mail: gqhuang@hkucc.hku.hk

D. W. L. Cheung
Department of Computer Science, The University of Hong Kong, Hong Kong, People's Republic of China

L. Liang
Business School, University of Science and Technology of China, Hong Kong, People's Republic of China

Secondly, ATC provides excellent flexibility for conducting scenario analysis. For example, we can set targets to extreme lower and upper bounds (normally 0 and ∞) for testing the boundaries of decision variables in minimization and maximization. We can also set several targets within a range to test the sensitivity and trend of different decision variables. In addition, it is easy to observe if some decision variables have taken their values at the upper or lower bounds specified in the associated constraints. Such decision variables should be highlighted for the designers to decide if it is necessary and feasible to change such extreme bounds to achieve better solutions.

Thirdly, ATC facilitates collaborative problem solving in system design. Sub-problems of different nature can be solved by using different (heterogeneous) models relatively independent of each other. The specialist users with relevant disciplines can simply focus on the issues that they are familiar with. All these are made possible by decoupling the systems and replacing their interactions by passing target values of their responses between them.

Fourthly, ATC enables distributed computing over the Internet to extend computational capacity. This is a significant advantage over the ATO approaches. For example, the dynamic programming developed for optimal supply chain configuration is limited to a certain number (typically 50–100) of sub-problems/stages. With ATC, sub-problems can be solved at separate computers. Sub-problems are much simpler and therefore can be solved by less powerful computers. More complex sub-problems can be scheduled to be solved on more powerful computers.

Fifthly, ATC supports optimization by multiple criteria. The objective function is generic in the sense that the discrepancies accumulate between all responses (criteria) concerned and their desired targets in absolute terms. In fact, responses can be freely added or removed from the objective function to conduct scenario analysis.

Sixthly, ATC is not limited to its original basic form. In fact, it has been widely extended by both of the researchers who originally proposed ATC to suit particular requirements of specific problems.

Although gradually gaining popularity in the vehicle design engineering community due to the powerful features mentioned above, ATC has not yet found wide application in optimal design of complex engineering and industrial systems as it deserves. A simple literature search on the Internet and in the university libraries has only resulted in about two dozen publications, most of which are related to projects or researchers at the University of Michigan as mentioned previously. It is not clear why this is the case. Even after understanding how ATC generally works, the overhead of setting up an ATC analysis is quite large for non-experienced analysts. It is quite straightforward to define a complex hierarchical system using the ATC convention. It is by no means a trivial task to define the communications between the different local optimization systems and the protocol regarding cycles of cascading and backtracking down and up the hierarchy. These overheads may become the possible hindrances as to why ATC has not yet been used by researchers and practitioners as it deserves to be.

The research reported in this paper is aimed at overcoming such major hindrances by proposing a generic and extensible computational infrastructure/framework called atcPortal. When designing and developing atcPortal, we adopt the latest Internet and web technologies. Unlike Chi language [2, 3], we propose to take advantage of the popular XML to create a new but easy to use language called atcXML, for not only modeling the hierarchical systems but also defining ATC optimization problems for all system components. atcXML also acts as a blackboard mechanism for presenting output results. A distributed computational multi-agent environment is created from the atcXML definition file. A Java-based program, atcEngine, has been developed to coordinate the distributed problem solving between individual components represented by TopAgents.

When we devise the atcPortal infrastructure, several jargons are created: Targent (design-time system component) and TopAgent (run-time system component). We added "n" in the word "Target" to reflect the extensive use of the concept of agents in developing this portal. We blended "T" and "r" with the word "agent" to reflect the fact that this portal is especially developed for supporting Target Cascading. Also, the name of TopAgents reflects the fact that the individual agents are for solving Target Optimization problems.

The rest of this paper is organized as follows. Section 2 outlines the overview of the proposed atcPortal. Sections 3, 4 and 5 present some details about the three main atcPortal components, namely atcXML, atcEngine and smoUDDI, respectively. Because of limited space, it is not possible to include a complete case study in the paper. A prototype atcPortal website is under construction for presenting further information about the atcPortal project and case studies (http://www.digiprise.org/atcportal).

2 atcPortal: an extensible and generic agent-based ATC portal

atcPortal is a web portal designed and developed specifically for supporting and facilitating analytical target cascading (ATC) for optimal design of complex hierarchical systems. At present, rudimentary ATC facilities and mechanisms are supported. Figure 1 presents a brief overview.

atcPortal involves two main phases: Definition and Execution. The Definition phase is heavily dependent on the ATC analysts. The output from this Definition phase is an XML-based definition file in the atcXML standard. atcXML is a special XML-based language devised for completely defining an ATC project/problem including hierarchical system modelling and optimization problems associated with individual system components. The Definition phase includes several tasks. Firstly, the system analyst defines and models the hierarchical system. This involves defining the system components, their key parameters and hierarchical relationships. Secondly, the

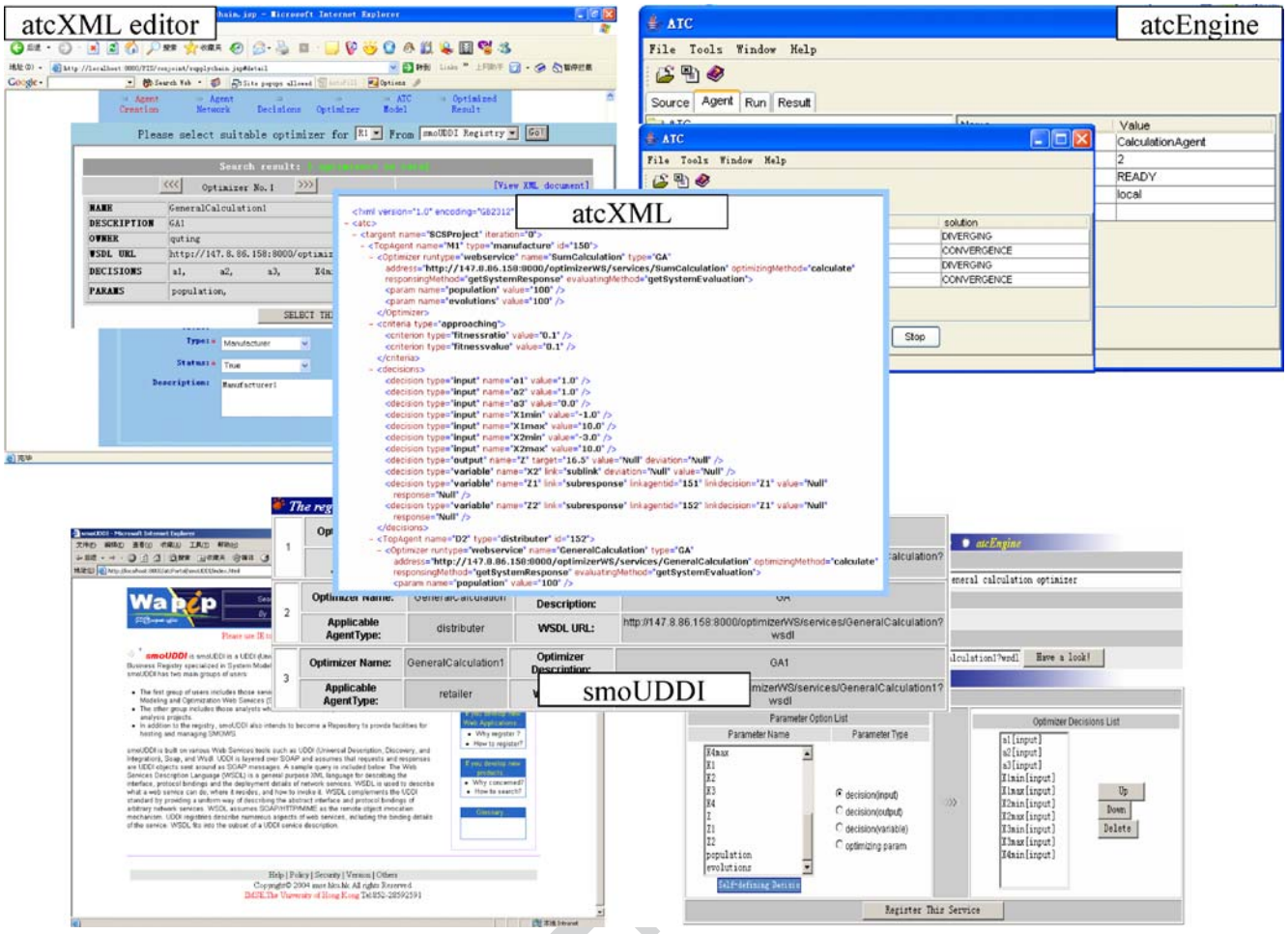

Fig. 1 atcPortal overview

system analyst defines target optimization problems for all the constituent system components. This normally involves setting up the objective function, and establishing equality and inequality constraints over the decision variables and their domains. Thirdly, the system analyst chooses appropriate optimization solution methods and convergence criteria for all the system components to solve their target optimization problems. This stage usually involves specifying the locations of the optimization problem solvers and their associated parameters. To avoid manual programming of the atcXML definition file, an atcXML editor is developed which provides a web-based visual editing environment for the atcPortal users, i.e. the system analyst.

The second Execution phase is mainly managed by the atcPortal itself through an atcEngine. atcEngine is a multi-agent system where individual agents solve their local problems and are coordinated by a specific ATC strategy to achieve the overall optimum. atcEngine does not allow the ATC analyst to change ATC parameters anymore. Instead, the ATC analyst is able to inquire about the progress and status of the ATC application with the help of its user interface. atcEngine accomplishes several tasks. It first generates and sets up the multi-agent community based on the atcXML definition. Then, atcEngine starts its process with the top-level component (agent), cascades down and backtracks up the system hierarchy recursively until the top-level agent converges (assume that all sub-level agents converge). During this process, the analyst is able to interrogate the status of the ATC process without being able to change anything except for stopping the process.

atcPortal provides three key components equipped with appropriate user interfaces to support the three ATC stages. They are atcXML, smoUDDI, and atcEngine. atcXML is a special XML-based language devised for defining an ATC project/problem including hierarchical system modelling and optimization problems associated with individual system components. atcXML and its editor facilities fully support the first stage of hierarchical system modeling and partially supports the second stage of optimization problem formulation.

Both definition and execution phases rely on the computational resources that must be made available for system modeling and optimization over the Internet. atcPortal provides a directory called smoUDDI where such resources are available. smoUDDI is a UDDI (universal description, discovery and integration) business

registry specialized in system modeling and optimization web services (SMOWS). The three atcPortal components are constructed with a few important concepts which will be discussed in detail in the following three sections.

3 atcXML: ATC definition language

atcXML is the special-purpose language devised in atcPortal for defining an ATC project. The ATC definition encompasses the following aspects:

- General project description information
- Defining the system hierarchical structure
- Defining system characteristics
- Defining ATC optimization models

In order to facilitate the discussion and to save space, a simple system shown in Fig. 2 is used throughout this paper. This system is abstracted and modified from an example presented by [6]. The example was slightly modified to create three levels in the hierarchy. Figure 2 shows the system hierarchy and Table 1 lists the 105 lines of the atcXML definition file. An ATC project is defined in the atcXML file between the pair of brackets<ATC …> … </ATC>. General project information is defined by the attributes of the "ATC" tag.

3.1 Hierarchical system modelling

System modeling includes four general aspects: composition—what components constitute a system, configuration—how system components are related to each other, characteristics—by what system components are described, and constraints—how characteristics of system components are related to each other.

ATC applies predominantly to hierarchical systems. The composition of a system is relatively simple. That is, the system consists of a set of components which in turn consist of lower-level components. This recursive composition repeats until bottom-level elementary components which do not have further child components.

In atcXML, each system component is defined by the "Targent" tag—the pair of brackets "<Targent …> …</Targent>". Each Targent is defined with several key attributes, including "id" and "name". The "id" attribute is unique to the Targent and must not be the same between any two Targents. There are five pairs of the brackets, defining the five Targents corresponding to the five components of the hierarchical system shown in Fig. 2.

The hierarchical relationships between system components are implied by the nesting relationships between the Targents.

3.2 Defining system characteristics

Each system component (i.e. Targent) is described by a set of characteristics or decisions. atcXML distinguishes between three types of characteristics: input parameters whose values must be specified, independent decision variables, and output dependent decision variables. All three types of decisions are defined in atcXML using one "decision" tag with a "type" attribute for differentiating them. The "data" attribute is for defining the data type of the decision, e.g. integer, real, discrete, etc. If it is of the continuous types (type="real" or type="integer"), then the domain may be specified by the lower and upper limits. If

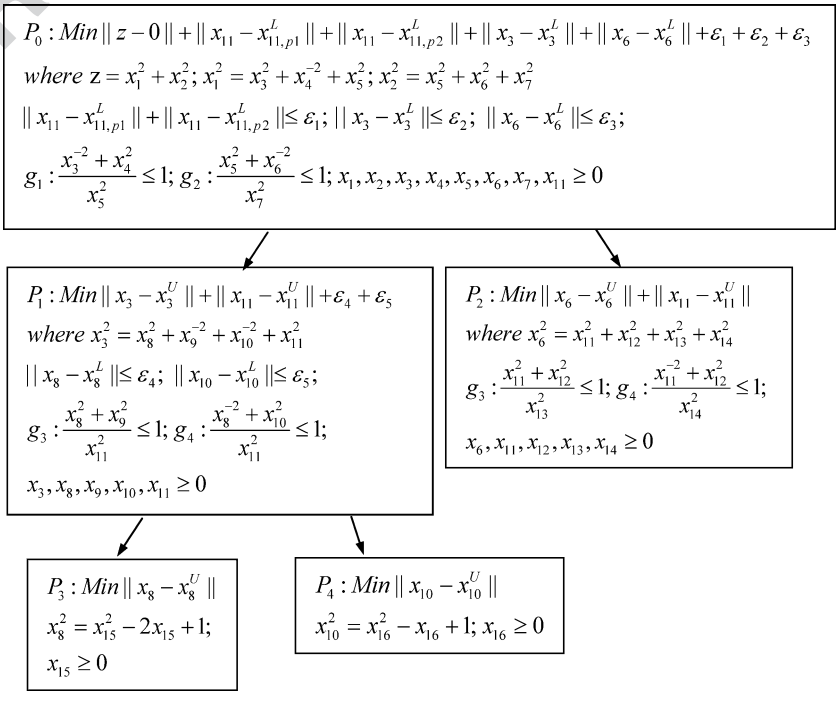

Fig. 2 Example of system hierarchy and ATC sub-problems

t1.1 **Table 1** atcXML definition file for the case study shown in Fig. 2

```
<?xml version="1.0" encoding="UTF-8"?>
<ATC project="Geometric Test" team="GQ"
    date="14/2/2004">
  <Targent id="p0" name="p0 " runat="local">
   <Optimizer id="p0" name="simpleGA0" runtype="WS"
 address="http://147.8.86.45/optimizerWS/services/SGA0"
 optimizingMethod="calculate">
    <param name="population" value="1000"/>
    <param name="evolution" value="500"/>
    </Optimizer>
     <Decisions>
  <decision type="dependent" data="real" name="x1" lower="0"/>
  <decision type="dependent" data="real" name="x2" lower="0"/>
  <decision type="independent" data="real" name="x3" lower="0""/>
  <decision type="independent" data="real" name="x4" lower="0"/>
  <decision type="independent" data="real" name="x5" lower="0"/>
  <decision type="independent" data="real" name="x6" lower="0"/>
  <decision type="independent" data="real" name="x7" lower ="0"/>
  <decision type="independent" name="x11"/>
  <decision type="dependent" data="real" name="z" target="0"
    lower="0"/>
    </Decisions>
    <Links>
      <link decision="x11">
        <with decision="x11" of="p1"/>
        <with decision="x11" of="p2"/>
      </link>
      <link decision="x3" with="x3" of="p1"/>
      <link decision="x6" with="x3" of="p2"/>
    </Links>
    <Converge>
      <criterion on="or" decision="objective" type="ratio"
    value="0.1"/>
      <criterion on="or" decision="objective" type="deviation"
    value="0.01"/>
    </Converge>
    <Targent id="p1" name="p1" runat="local">
   <Optimizer id="p0" name="simpleGA1" runtype="WS"
address="http://147.8.86.46/optimizerWS/services/SGA1"
optimizingMethod="calculate" >
        <param name="population" value="1000"/>
        <param name="evolution" value="500"/>
    </Optimizer>
      <Decisions>
        <decision type="dependent" name="x8" lower="0"/>
        <decision type="independent" name="x9" lower="0"/>
        <decision type="dependent" name="x10" lower="0"/>
        <decision type="independent" name="x11"
    lower="0"/>
        <decision type="dependent" name="x3" lower="0"/>
      </Decisions>
      <Links>
        <link decision="x8" with="x8" of="p3"/>
        <link decision="x10" with="x10" of="p4"/>
      </Links>
      <Converge>
        <criterion on="or" decision="objective" type="ratio"
    value="0.1"/>
        <criterion on="or" decision="objective"
    type="deviation" value="0.01"/>
      </Converge>
      <Targent id="p3" name="p3" runat="local">
   <Optimizer id="p3" name=" simpleGA3" runtype="WS"
address="http://147.8.86.47/optimizerWS/services/SGA2"
 optimizingMethod="calculate" >
        <param name="population" value="1000"/>
        <param name="evolution" value="500"/>
      </Optimizer>
      <Decisions>
        <decision type="dependent" name="x8" lower="0"/>
        <decision type="independent" data="discrete"
  name="x15"/>
          <member value="1.0"/>
          <member value="2.0"/>
          <member value="2.8"/>
        </decision>
      </Decisions>
      <Converge>
<criterion on="or" decision="objective" type="ratio" value="0.1"/>
<criterion on="or" decision="objective" type="deviation" value="0.01"/>
      </Converge>
    </Targent>
    <Targent id="p4" name="p4" runat="localhost:4446">
    <Optimizer id="p4" name=" simpleGA4" runtype="WS"
address="http://147.8.86.48/optimizerWS/services/SGA4"
 optimizingMethod="calculate" >
        <param name="population" value="1000"/>
        <param name="evolution" value="500"/>
      </Optimizer>
      <Decisions>
        <decision type="dependent" name="x10" lower="0"/>
        <decision type="independent" name="x16" lower="0"/>
      </Decisions>
      <Converge>
        <criterion on="or" decision="objective" type="ratio"
  value="0.1"/>
        <criterion on="or" decision="objective" type="deviation"
  value="0.01"/>
      </Converge>
    </Targent>
  </Targent>
  <Targent id="p2" name="p2" runat="local"/>
   <Optimizer id="p2" name=" simpleGA2" runtype="WS"
  address="http://147.8.86.49/optimizerWS/services/SGA3"
  optimizingMethod="calculate" >
      <param name="population" value="1000"/>
      <param name="evolution" value="500"/>
    </Optimizer>
    <Decisions>
      <decision type="independent" name="x11" lower="0"/>
      <decision type="independent" name="x12" lower="0"/>
      <decision type="independent" name="x13" lower="0"/>
      <decision type="independent" name="x14l" lower="0"/>
      <decision type="dependent" name="x6" lower="0"/>
    </Decisions>
    <Converge>
      <criterion on="and" decision="objective" type="ratio"
  value="0.1"/>
      <criterion on="or" decision="objective" type="deviation"
  value="0.01"/>
    </Converge>
  </Targent>
</Targent>
<ATC>
```

an output is discrete, its domain needs to be defined through a membership (lines 62–64).

3.3 Defining common and cascading decision variables

Because of hierarchical decomposition of the system used in ATC, the relationships between the parent and children components between the two adjacent levels and the relationships between the sibling components of the same parent are described by linking variables shared between them.

Let us first consider the case of defining linking variables between sibling Targents at the same level. Individual sibling Targents first define their own variables independent of each other. If they share a particular common variable, then a dummy decision variable is created in their parent Targent. This dummy variable is linked to the common variables, as defined in lines 23–26 for variable x11. Common variables may have different names. atcEngine will treat these three variables as the single one and build up the objective functions for the Targents (TopAgents) involved in due course.

The case of linking variables between the parent and child Targents at two adjacent levels is simpler to handle because such a relationship is one–one. A few attributes are added in the "decision" tag, as in lines 27–28 in Table 1.

3.4 Defining response targets

Target values may be defined for one or more output variables (responses) for Targents. As the name ATC implies, this is mandatory for the top-level Targent while optional for other Targents. This is easily done by explicitly defining the "target" attribute as in line 20 for decision "z". atcEngine will adjust the objective function to include the term |z-Target| when creating the TopAgents.

3.5 Defining targent optimizers

In ATC, individual system components formulate their own target optimization problems and solve these problems with certain optimization methods. The target optimization problems are said to be formulated in the preceding section when system characteristics are fully defined. This section is only concerned with the solution methods.

At present, optimization solution methods are assumed to be reflected in the computational programs for individual system components. Such computational programs are in turn treated as blackbox web services—Optimizers—that are interoperable on the Internet.

This assumption has simplified the atcXML definition to a large extent. For example, Targents (TopAgents as their run-time components) in the case study use the Optimizers that are based on genetic algorithm (GA). Five of such Optimizers have been developed and deployed as web services. It is assumed that the provider of an Optimizer register necessary information in order for this Optimizer to be called on the Internet. Such information may include the Internet location, input and output data templates, and parameters for setting up the optimization.

The "Optimizer" tag is used for defining the Targent's Optimizer, as in lines 4–7. The Optimizer "simpleGA" in the above definition is an optimization web service built with simple/basic GA and registered in smoUDDI. Parameters defined are specifically associated with the optimization method. They are passed to the Optimizer when it is called upon. In this case, "Population", "Generation" and "Penalty" (on inequality constraints) are the three parameters for simpleGA. The convergence and/or stop criteria may be explicitly defined as parameters or derived from the parameters defined. For example, simpleGA will stop after a maximum generation of 5,000 if other convergence criteria do not become valid.

3.6 Defining targent convergence criteria

A Targent is said to converge if and only if its local solution is consistent with the solution obtained from the results backtracked from its children. The meaning of "consistent" refers to the satisfaction of the convergence criteria defined explicitly by the pair of<Converge> ...</Converge> brackets as in lines 30–33. The Targent converges when all compulsory criteria with attribute "and" must be met while any of the criteria with "or" attribute is satisfied.

The criterion defined in line 31 can be interpreted as where value=0.1 and criterion defined in line 32 as where value=0.01. These two criteria are defined for the overall objective function. Similar criteria may be defined for linking variables, in addition to the overall objective function of the Targent.

It is important to note the complete difference between the convergence of the Targent and its Optimizer. Assume the Optimizer is based on a simple GA that is asked to converge into the best solution after a given number of generations. The convergence criteria of the Optimizer are only implicitly defined in atcXML through the Optimizer's parameters. The Optimizer must interpret them as designed.

4 atcEngine: atcPortal engine

atcEngine in the atcPortal is the central mechanism that creates, starts, monitors and coordinates the participating TopAgents to accomplish the computational ATC process as defined in the atcXML file. atcEngine is basically a multi-agents system (MAS) consisting of a community of TopAgents. TopAgents are run-time implementation of Targents as defined in atcXML. The TopAgent community Organizer is responsible for creating TopAgents from Targents as instances of the TopAgent Java class. TopAgents are presently implemented in Java and each TopAgent will run in a separate thread. TopAgents are

supposed to be "thin" in the sense that they provide containers for the corresponding Optimizers which are supposed to be "fat" and consuming substantial computational resources. Therefore, TopAgents are not designed to run in a distributed computational environment but Optimizers are.

In order to reflect the distributed and collaborative nature of ATC problem solving, atcEngine has been designed to maximize the autonomy of individual TopAgents in managing their own internal affairs and minimizing the central control over them. After their creation, TopAgents reason and maintain their own states, and they reason and take their own actions. They interact with each other by exchanging messages.

Following the object-oriented convention, we discuss the TopAgent model in terms of properties and methods.

4.1 TopAgent properties

A TopAgent maintains a set of properties. During the process, all attributes defined for Targents in atcXML are converted into TopAgent properties. In addition, the hierarchical relationships between system components (i.e. Targents in the atcXML file) are also captured as TopAgent properties. Connections to the remote Optimizers defined as Web services in atcXML are verified with smoUDDI and necessary information is deployed within the corresponding TopAgents.

In addition to the above design-time properties, each TopAgent has run-time properties, e.g. status property, inbox and outbox of messages, etc.

4.2 TopAgent status

The status of a TopAgent is defined by the following possible values:

- Idle: A TopAgent is not asked to do anything yet.
- Optimizing: A TopAgent has called its Optimizer to solve the local ATC optimization problem and is waiting for the result.
- Cascading: A TopAgent has cascaded downwards its children TopAgents and is waiting for their feedback.
- Backtracking: A TopAgent has converged and backtracked upwards its parent TopAgent.
- Succeeding: All TopAgents reach the "Succeed" status when the top-level TopAgent converges.
- Failing: A TopAgent fails when it does not have a solution to its local target optimization problem or any of its children backtracks with "failure".

4.3 Messages and message box

Each TopAgent maintains a message box. When a TopAgent needs to interact with another TopAgent, it first prepares a message and then puts it in the message box of the receiving TopAgent. The TopAgent's Messenger continuously processes the incoming messages.

There are three types of messages: Cascade, Backtrack, and Optimize. The Cascade and Backtrack messages are exchanged between adjacent TopAgents in the system hierarchy. The purpose of a Cascade message is to cascade the values generated by the parental TopAgent for the hierarchical linking decision variables to its children. Therefore, it will take the following form:

```
<Cascade ID=Unique From="p0" To="p1">
  <Decision name="x3">2.00</Decision>
  <Decision name="x11">1.00</Decision>
...
</Cascade>
```

Likewise, the purpose of a Backtrack message is to submit the values achieved upon the TopAgent convergence to its parent for the hierarchical linking decision variables to its parent. Therefore, it will take the following form:

```
<Backtrack ID=Unique From="p1" To="p0">
  <Decision name="x3">2.01</Decision>
  <Decision name="x11">1.01</Decision>
...
</Backtrack>
```

The Optimize messages are internal in the sense that they are exchanged between the TopAgent and its Optimizer as web service. The Optimize message is in fact a pair of Request and Response messages sent and received by the TopAgent to and from its Optimizer. The purpose of the Request message is to start up the remote web service (Optimizer) and pass the necessary decision parameters and variables to the Optimizer web service. The purpose of the Response message is to return all the resulting data from the Optimizer web service to the TopAgent. Both request and response messages basically follow the SOAP (simple object access protocol) standard. The main body of a typical request message contains the location of the web service and parameters to be passed to this web service. For example a Request message for TopAgent "p1" looks like:

```
<Request to=" http://147.8.86.5/wsdl/demoGeo/p1">
  <Decision name="x3:uvalue">1.12</Decision>
  <Decision name="x11:uvalue">2.21</Decision>
  <Decision name="x8:lvalue">0.98</Decision>
  <Decision name="x8:old">0.85</Decision>
...
</Request>
```

The response message from the Optimizer of the "p1" TopAgent contains the values obtained from the Optimizer upon the given parameters:

<Response from=" http://147.8.86.46/optimizerWS/services/SGA1">

```
<Decision name="x3"></Decision>
<Decision name="x11"></Decision>
<Decision name="x8"></Decision>
<Decision name="x9"></Decision>
<Decision name="x10"></Decision>
<Decision name="objective"></Decision>
<Decision name="et"></Decision>
<Decision name="ey"></Decision>
<Decision name="er"></Decision>
...
</Response>
```

TopAgents process the incoming messages, update their properties and determine what should be done next. These tasks are accomplished by their built-in methods.

4.4 TopAgent Methods

The TopAgent Java class includes several main methods that accomplish certain special tasks. Messenger is the method mainly responsible for processing the incoming messages in the message box. It identifies and calls the appropriate method (action) depending on the type of the message and current status of the TopAgent.

The first method called by the messenger is the property manager method. It reads a message and updates the properties according to the message type and body information.

The Cascade and Backtrack methods are triggered by the messenger method if the incoming message type is Cascade and Backtrack, respectively. How Cascade and Backtrack methods perform the tasks are governed by the chosen ATC control strategy. There are four ATC strategies as discussed by Michelena et al. [12]. At present, only Strategy IV is used because it has been used in most of the reported case studies. In Strategy IV of ATC, the Cascade method and the Backtrack method follow the procedure outlined in Fig. 3.

The Cascade method starts with triggering the Optimize method and ends with sending Cascade messages to all its children, if any, or otherwise a Backtrack message to its parent.

If all the children have backtracked, the Backtrack method triggers the Converge method. The Converge method is responsible (1) for assessing the values of the different terms included in the TopAgent's objective function from the children's feedback data, and (2) for checking if TopAgent meets the specified convergence criteria. The TopAgent ends its Backtrack method by preparing and sending a Backtrack message up to its parent if it converges, or otherwise triggering the Optimize method again.

The Converge method is responsible for testing if the specified convergence criteria have been satisfied. If the TopAgent does not have any children, then it converges when its Optimizer converges. If it does have children, then all the "and" criteria are first checked and then each "or" criterion is checked. This method changes the status of the TopAgent.

The Optimize method also changes the status of the TopAgent by temporarily setting it to "Optimizing". This status will be switched to "Cascading" after the Optimizer returns the response message. The Optimize method is triggered on two occasions: (1) within the Cascade method, and (2) within the Backtrack method after the TopAgent fails to converge upon rebalancing.

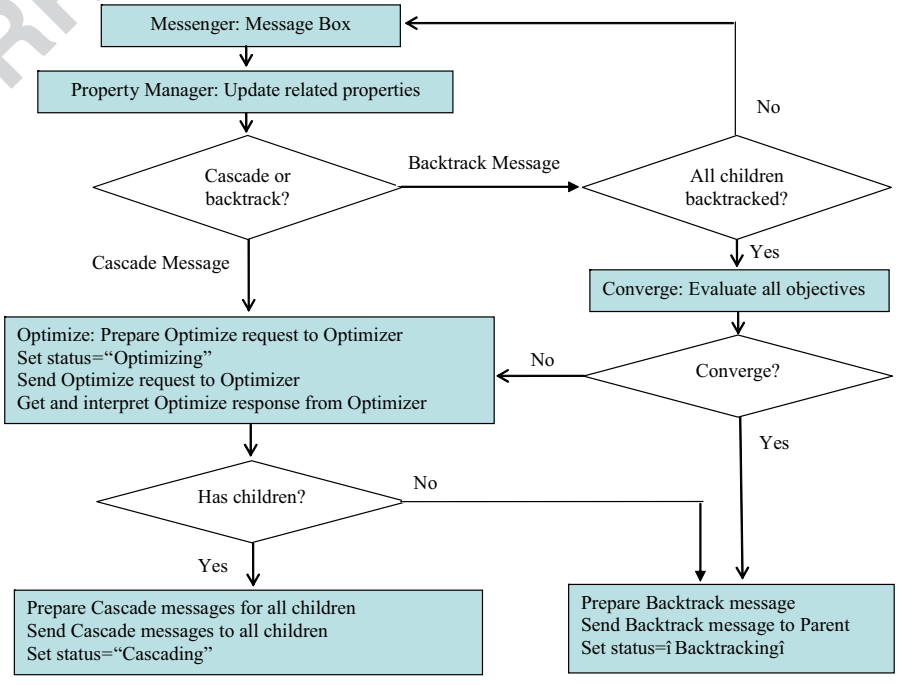

Fig. 3 Information/workflow of a single TopAgent's Cascade and Backtrack methods

5 smoUDDI: atcPortal's UDDI directory of SMOWS

When formulating the ATC problem mathematically, [6] distinguishes between optimal design models and analysis models. Optimal design models call analysis models to evaluate responses from corresponding system components. Analysis models take lower level decision variables and parameters, as well as responses from lower level components in the system hierarchy, and return responses for optimization problems of upper level system components. Technically, analysis models are equivalent to equality constraints defined for an optimization problem.

smoUDDI, as mentioned previously, is a UDDI business registry specialized in system modeling and optimization web services (SMOWS). smoUDDI has two main groups of users. The first group of users includes those service providers who are responsible for providing SMOWS. Although SMOWS are usually devised for the optimization of common modules of certain practical systems, such as the chassis of a vehicle, to the illustrative mathematical system presented in Fig. 2, these SMOWS refer only to several optimizing programs of particular quadratic equations. The other group includes those analysts who consume and apply these SMOWS in their ATC analysis projects. In atcPortal, this consumption and application refer to the invoking of optimizer by TopAgents running in the solution process of atcEngine.

smoUDDI is built on various web services tools such as UDDI, SOAP, and WSDL. UDDI is layered over SOAP and assumes that requests and responses are UDDI objects sent around as SOAP messages. The web services description language (WSDL) is a general purpose XML language for describing the interface, protocol bindings and the deployment details of network services. WSDL is used to describe what a web service can do, where it resides, and how to invoke it. WSDL complements the UDDI standard by providing a uniform way of describing the abstract interface and protocol bindings of arbitrary network services. WSDL assumes SOAP/HTTP/MIME as the remote object invocation mechanism. UDDI registries describe numerous aspects of web services, including the binding details of the service. WSDL fits into the subset of a UDDI service description.

According to the work logic of web services, there are three main stages required before the web service can truly be integrated into the main system, namely web service registering (on UDDI), web service finding (from UDDI) and web service binding (through WSDL). These stages are embodied as optimizer registering (on smoUDDI), optimizer finding (from smoUDDI) and optimizer binding (through optimizer's WSDL) in the atcPortal. Following will be the detailed explanation of these three stages in the atcPortal with regard to the problem in Fig. 2.

5.1 Optimizer registration with smoUDDI

In this stage, two types of information are published on smoUDDI by the optimizer provider. The first is the optimizer's functional information, which includes its applying scope, enabling algorithm, optimizing speed and even its charge. This type of information provides the basis for the Optimizer Finding stage. The second is the optimizer's interfacial description, i.e. the WSDL, which comprises its Internet location, input and output data templates. This WSDL enables the correct binding of optimizer during the Optimizer Binding stage.

5.2 Optimizer search smoUDDI

Optimizer Search is conducted during the generation process of the atcXML definition file. In order to completely define a Targent XML node for the atcXML definition file, the optimizer, which represents the optimization solution methods of this Targent, must be simultaneously identified. By using the atcXML editor, the optimizer finding operation can be automatically supported by smoUDDI through the following four steps:

(1) atcXML editor prepares a SOAP message including the Targent's classification information, and send it to smoUDDI.
(2) smoUDDI searches the repository to pick out all the optimizers whose applying scope match the classification information parsed out from the received message, then feed back the functional information and the WSDL address of these qualified optimizers to atcXML editor via SOAP.
(3) atcXML editor parses out all these potentially suitable optimizers from the returned SOAP message and lists them in the editor for selection.
(4) The system analyst selected the most suitable optimizer through further evaluation.

The most suitable optimizer has been determined after the completion of step 4, but to accomplish the final definition of atcXML file so as to facilitate the later ATC solution progress, this optimizer should be bound with the Targent.

5.3 Optimizer binding through optimizer's WSDL

Based on the WSDL address of the selected optimizer, its interfacial description is easily fetched back through smoUDDI, and the atcXML editor could then create the optimizer tag, which contains the invoking detail of the optimizer, for the Targent XML node. So far, the atcXML definition file has been successfully defined, and an example can be seen from lines 4–7 of Table 1.

6 Concluding discussions

This paper has proposed and discussed a prototype web-based multi-agent system called atcPortal for specifically supporting optimal design of complex systems using analytical target cascading. Its practical contribution is that ATC analyses can be easily set up and conducted with the

help of this atcPortal, especially after it is fully developed. Its scientific contributions include at least the following two aspects. Firstly, the latest web technologies have been employed for the development and implementation of the atcPortal. The design optimization is truly distributed and collaborative on the Internet. Secondly, the atcPortal provides a basis for extending the ATC methods to incorporate and experiment with other strategies and for other types of system structures (e.g. hierarchical structures with limited number of shared components, and hierarchical structures with alternative branches). Discussions on these extensions will be reported separately in the near future.

Finally, although many of the limitations that this prototype atcPortal system suffers at present can be easily improved, it is important to point out a fundamental limitation on the balance between the generality and extensibility as set out in the title of the paper. Optimizers as web services collected in the smoUDDI are supposed to be generally reusable regardless if they are used in ATC analyses or for other applications. At present, they contain ATC-specific elements. In order to overcome this limitation, it is necessary to introduce two tiers of web services: one for Optimizers which are allowed to contain ATC-specific elements to enhance their extensibility, and the other for response/analysis models without ATC-specific elements to improve their generality. This is the area requiring substantial research and development efforts.

Acknowledgements The authors are most grateful to the Hong Kong University Committee on Research and Conference Grants, William Mong Engineering Fund and NSFC (Grant No: 70371023) for financial support that made this research possible.

References

1. Choudhary R, Malkawi A, Papalambros PY (2003) A hierarchical design optimization framework for building performance analysis. Proceedings of the 8th international building performance simulation association (IBPSA) conference, Eindhoven, Netherlands, 11–14 August 2003
2. Etman LFP, Kokkolaras M, Papalambros PY, Hofkamp AT, Rooda JE (2002) Coordination specification for analytical target cascading using the Chi language. Proceedings of the 9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, work in progress, AIAA–2002–5637, Atlanta, Georgia, USA, 4–6 September 2002
3. Fellini R, Kim HM, Kokkolaras M, Michelena N, Papalambros PY (2001) Target cascading for design of product families. Proceedings of the 4th congress on structural and multidisciplinary optimization, Dalian, China, 4–8 June 2001
4. Ge P, Lu S C-Y, Suh NP (2002) An axiomatic approach for target cascading of parametric design of engineering systems. Ann CIRP 51(1):111–114
5. Kim HM, Kokkolaras M, Louca L, Delagrammatikas G, Michelena N, Filipi Z, Papalambros P, Assanis D (2002) Target cascading in vehicle redesign: a class VI truck study. Int J Veh Des 29(3):199–225
6. Kim HM, Michelena NF, Papalambros PY, Jiang T (2000) Target cascading in optimal system design. Proceedings of the 2000 ASME design automation conference, DAC paper 14253, Baltimore, 10–13 September 2000
7. Kim HM, Rideout DG, Papalambros PY, Stein JL (2001) Analytical target cascading in automotive vehicle design. Proceedings of the 2001 ASME design automation conference, DAC–21079, Pittsburgh, Pennsylvania, USA, 9–12 September 2001
8. Kim HM (2001) Target cascading in optimal system design. PhD Dissertation, Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, USA
9. Louca LS, Kokkolaras M, Delagrammatikas GJ, Michelena NF, Filipi ZS, Papalambros PY, Assanis DN (2002) Analytical target cascading for the design of an advanced technology heavy truck. Proceedings of the ASME international mechanical engineering congress and exposition, New Orleans, Louisiana, 17–22 November 2002
10. Mahmoud HA, Kabamba PT, Ulsoy AG, Brusher GA (2002) Target reduction and balancing using system norms. Proceedings of the American control conference, Anchorage, AK, May 2002
11. Mahmoud HA, Kabamba PT, Ulsoy AG, Brusher GA (2003) Ranking subsystem targets according to their influence on system performance. Proceedings of the American control conference, Denver, CO, June 2003
12. Michelena N, Park H, Papalambros P (2003) Convergence properties of analytical target cascading. AIAA J 41:897–905
13. Nyström M, Larsson T, Karlsson L, Kokkolaras M, Papalambros PY (2003) Linking analytical target cascading to engineering information systems for simulation-based optimal vehicle design. ICED 03, 14th international conference on engineering design, research for practice-innovative products, processes and organisations, 19–21 August 2003, Stockholm, Sweden