

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

7 Extensible multi-agent system for optimal design of complex 8 systems using analytical target cascading

9 Received: 1 September 2004 / Accepted: 15 November 2004
10 © Springer-Verlag London Limited 2005

11 **Abstract** Analytical target cascading (ATC) has emerged
12 as an integrated approach and methodology for optimal
13 design of complex hierarchical systems with good conver-
14 gence. It facilitates collaborative design problem solving
15 and enables distributed computing to extend both the
16 capability and capacity. ATC is very powerful yet requiring
17 comprehensive understanding and efforts in setting up
18 specific projects. This paper presents a generic and ex-
19 tensible information infrastructure called atcPortal in the
20 form of a web portal where the ATC analyst defines the
21 ATC problem using a special-purpose XML-based lan-
22 guage called atcXML, conducts the ATC analysis, and
23 obtains the analytical results. atcPortal not only reduces the
24 overheads for analysts to set up the projects for practical
25 applications, but also allows the researchers to experiment
26 and extend the ATC methods under different system
27 structures and/or different ATC strategies.

28 **Keywords** Analytical target cascading ·
29 Target optimization · Web portal · Web services · XML ·
30 Hierarchical system modeling

31 1 Introduction

32 Analytical target cascading (ATC) has recently emerged as a
33 new approach to solving target optimization problems in

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

complex engineering systems which can be represented 34
hierarchically in terms of constituent components and 35
modeled mathematically in terms of decision variables [6]. The 36
formulation addresses the problem of propagating design 37
targets through a hierarchically partitioned product devel- 38
opment process, so that the local targets are consistent with 39
each other and with the overall top level targets. According to 40
[6], ATC includes four key steps: (i) development of 41
appropriate models, (ii) partitioning the system, (ii) formu- 42
lating the target cascading problems for each element of the 43
partition, and (iv) solving the partitioned problem through a 44
coordination strategy to compute all stated targets. The re- 45
searchers who originally proposed the ATC approach at the 46
University of Michigan have studied and demonstrated the 47
convergence with theoretical rigor [12] and through nume- 48
rous industrial case studies (mainly in the field of vehicle 49
design) [9, 5, 13]. Researchers in the same group have also 50
applied ATC for building design [1]. The framework has 51
been applied for thermal design and analysis of a fictional 52
three-zone building. The building is decomposed into two 53
levels. Different analysis tools are employed for computing 54
the performance targets at each decomposed level of the 55
building. Interestingly, most of the publications on ATC are 56
somewhat related to the projects and the people who are 57
working or have worked at the University of Michigan [e.g. 58
10, 11, 13] with a few exceptions [e.g. 4]. 59

ATC is a powerful method with several important 60
characteristics. Firstly, its convergence has been demon- 61
strated, both theoretically and through case studies. It has 62
been reported that the top-level component normally 63
converges after about ten iterations [7, 8]. Several case 64
studies have compared results from ATC and ATO (all at 65
once) and reported that the results are generally consistent 66
with each other. This indicates that ATC is at least as ef- 67
fective as ATO in solving the optimal system design prob- 68
lems. The efficiency of convergence has been improved by 69
much reduced complexity of sub-problems which converge 70
much faster than when they are treated as one big system. 71
Meanwhile, the solution spaces are narrowed progressively 72
after each cycle of cascading and backtracking (rebalan- 73
cing). This also speeds up the convergence. 74

75 Secondly, ATC provides excellent flexibility for con- 133
 76 ducting scenario analysis. For example, we can set targets 134
 77 to extreme lower and upper bounds (normally 0 and ∞) for 135
 78 testing the boundaries of decision variables in minimiza- 136
 79 tion and maximization. We can also set several targets 137
 80 within a range to test the sensitivity and trend of different 138
 81 decision variables. In addition, it is easy to observe if some 139
 82 decision variables have taken their values at the upper or 140
 83 lower bounds specified in the associated constraints. Such 141
 84 decision variables should be highlighted for the designers 142
 85 to decide if it is necessary and feasible to change such 143
 86 extreme bounds to achieve better solutions. 144

87 Thirdly, ATC facilitates collaborative problem solving in 145
 88 system design. Sub-problems of different nature can be 146
 89 solved by using different (heterogeneous) models relatively 147
 90 independent of each other. The specialist users with relevant 148
 91 disciplines can simply focus on the issues that they are 149
 92 familiar with. All these are made possible by decoupling the 150
 93 systems and replacing their interactions by passing target 151
 94 values of their responses between them. 152

95 Fourthly, ATC enables distributed computing over the 153
 96 Internet to extend computational capacity. This is a signif- 154
 97 icant advantage over the ATO approaches. For example, the 155
 98 dynamic programming developed for optimal supply chain 156
 99 configuration is limited to a certain number (typically 50– 157
 100 100) of sub-problems/stages. With ATC, sub-problems can 158
 101 be solved at separate computers. Sub-problems are much 159
 102 simpler and therefore can be solved by less powerful com- 160
 103 puters. More complex sub-problems can be scheduled to be 161
 104 solved on more powerful computers. 162

105 Fifthly, ATC supports optimization by multiple criteria. 163
 106 The objective function is generic in the sense that the 164
 107 discrepancies accumulate between all responses (criteria) 165
 108 concerned and their desired targets in absolute terms. In 166
 109 fact, responses can be freely added or removed from the 167
 110 objective function to conduct scenario analysis. 168

111 Sixthly, ATC is not limited to its original basic form. In 169
 112 fact, it has been widely extended by both of the researchers 170
 113 who originally proposed ATC to suit particular require- 171
 114 ments of specific problems. 172

115 Although gradually gaining popularity in the vehicle 173
 116 design engineering community due to the powerful features 174
 117 mentioned above, ATC has not yet found wide application 175
 118 in optimal design of complex engineering and industrial 176
 119 systems as it deserves. A simple literature search on the 177
 120 Internet and in the university libraries has only resulted in 178
 121 about two dozen publications, most of which are related to 179
 122 projects or researchers at the University of Michigan as 180
 123 mentioned previously. It is not clear why this is the case. 181
 124 Even after understanding how ATC generally works, the 182
 125 overhead of setting up an ATC analysis is quite large for 183
 126 non-experienced analysts. It is quite straightforward to 184
 127 define a complex hierarchical system using the ATC con- 185
 128 vention. It is by no means a trivial task to define the 186
 129 communications between the different local optimization 187
 130 systems and the protocol regarding cycles of cascading and 188
 131 backtracking down and up the hierarchy. These overheads 189
 132 may become the possible hindrances as to why ATC has

not yet been used by researchers and practitioners as it 133
 deserves to be. 134

The research reported in this paper is aimed at over- 135
 coming such major hindrances by proposing a generic and 136
 extensible computational infrastructure/framework called 137
 atcPortal. When designing and developing atcPortal, we 138
 adopt the latest Internet and web technologies. Unlike Chi 139
 language [2, 3], we propose to take advantage of the popular 140
 XML to create a new but easy to use language called 141
 atcXML, for not only modeling the hierarchical systems but 142
 also defining ATC optimization problems for all system 143
 components. atcXML also acts as a blackboard mechanism 144
 for presenting output results. A distributed computational 145
 multi-agent environment is created from the atcXML defi- 146
 nition file. A Java-based program, atcEngine, has been 147
 developed to coordinate the distributed problem solving 148
 between individual components represented by TopAgents. 149

When we devise the atcPortal infrastructure, several 150
 jargons are created: Targent (design-time system compo- 151
 nent) and TopAgent (run-time system component). We 152
 added “n” in the word “Target” to reflect the extensive use 153
 of the concept of agents in developing this portal. We 154
 blended “T” and “r” with the word “agent” to reflect the 155
 fact that this portal is especially developed for supporting 156
 Target Cascading. Also, the name of TopAgents reflects the 157
 fact that the individual agents are for solving Target 158
 Optimization problems. 159

The rest of this paper is organized as follows. Section 2 160
 outlines the overview of the proposed atcPortal. Sections 3, 161
 4 and 5 present some details about the three main atcPortal 162
 components, namely atcXML, atcEngine and smoUDDI, 163
 respectively. Because of limited space, it is not possible to 164
 include a complete case study in the paper. A prototype 165
 atcPortal website is under construction for presenting 166
 further information about the atcPortal project and case 167
 studies (<http://www.digiprise.org/atcportal>). 168

2 atcPortal: an extensible and generic agent-based ATC 169 portal 170

atcPortal is a web portal designed and developed specifi- 171
 cally for supporting and facilitating analytical target cas- 172
 cading (ATC) for optimal design of complex hierarchical 173
 systems. At present, rudimentary ATC facilities and mecha- 174
 nisms are supported. Figure 1 presents a brief overview. 175

atcPortal involves two main phases: Definition and 176
 Execution. The Definition phase is heavily dependent on 177
 the ATC analysts. The output from this Definition phase is 178
 an XML-based definition file in the atcXML standard. 179
 atcXML is a special XML-based language devised for 180
 completely defining an ATC project/problem including 181
 hierarchical system modelling and optimization problems 182
 associated with individual system components. The Def- 183
 inition phase includes several tasks. Firstly, the system 184
 analyst defines and models the hierarchical system. This 185
 involves defining the system components, their key 186
 parameters and hierarchical relationships. Secondly, the 187

Print will be in black and white

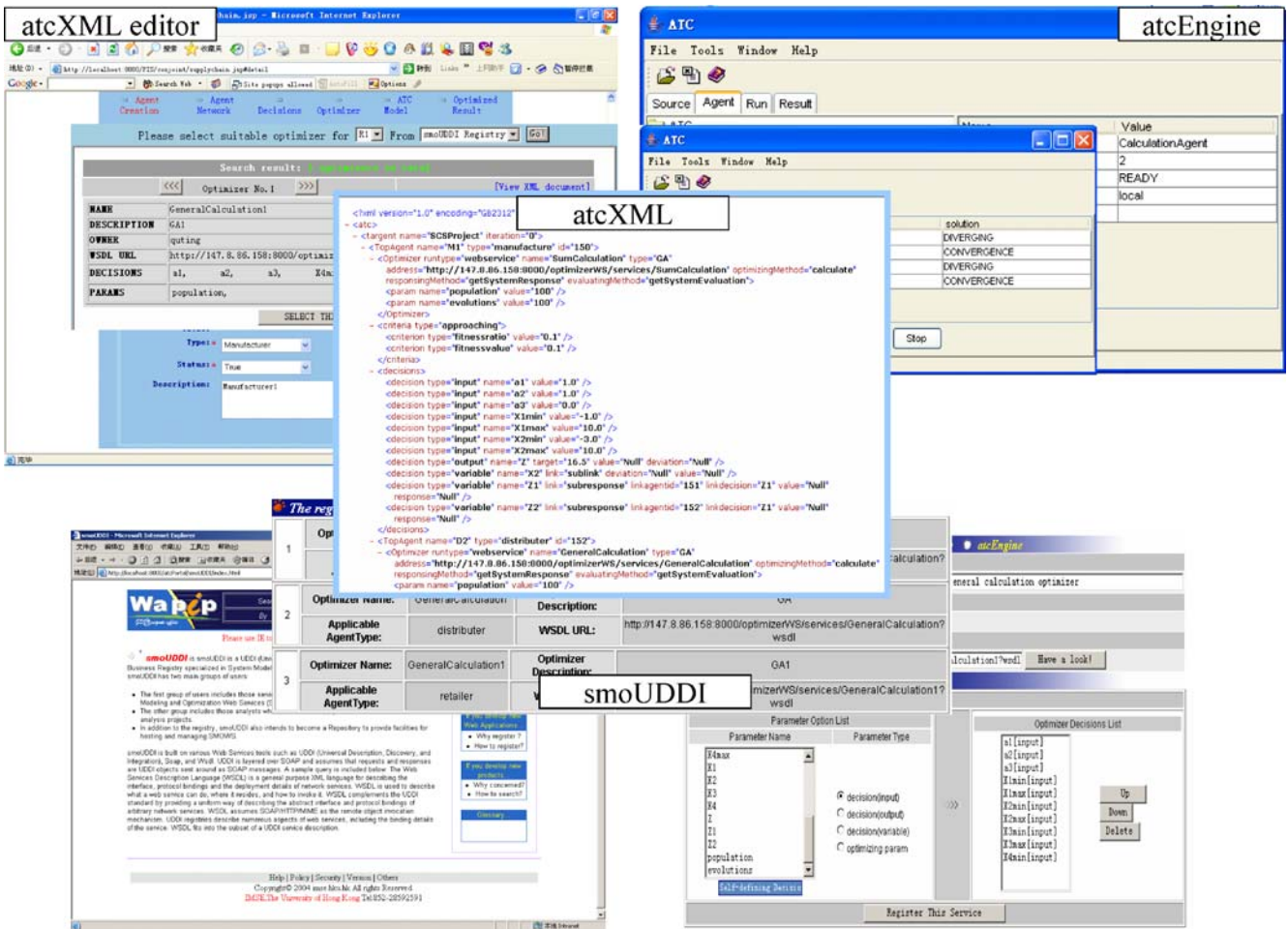


Fig. 1 atcPortal overview

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

201 The second Execution phase is mainly managed by the
 202 atcPortal itself through an atcEngine. atcEngine is a multi-
 203 agent system where individual agents solve their local
 204 problems and are coordinated by a specific ATC strategy to
 205 achieve the overall optimum. atcEngine does not allow the
 206 ATC analyst to change ATC parameters anymore. Instead,
 207 the ATC analyst is able to inquire about the progress and
 208 status of the ATC application with the help of its user
 209 interface. atcEngine accomplishes several tasks. It first
 210 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

211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233

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

238 **3 atcXML: ATC definition language**

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

- 242 – General project description information
- 243 – Defining the system hierarchical structure
- 244 – Defining system characteristics
- 245 – Defining ATC optimization models

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

256 **3.1 Hierarchical system modelling**

257 System modeling includes four general aspects: composi-
 258 tion—what components constitute a system, configuration
 259 —how system components are related to each other,
 260 characteristics—by what system components are de-

scribed, 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 compo-
 sition repeats until bottom-level elementary components
 which do not have further child components.

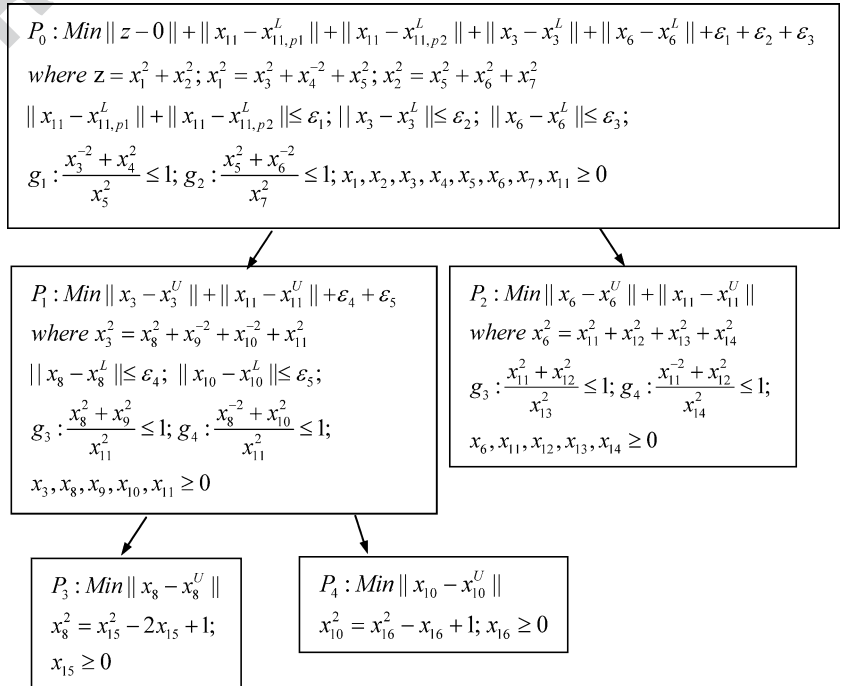
In atcXML, each system component is defined by the
 “Targent” tag—the pair of brackets “<Targent ...> ...</
 Targent>”. Each Targent is defined with several key
 attributes, including “id” and “name”. The “id” attribute
 is unique to the Targent and must not be the same between
 any two Targents. There 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 compo-
 nents 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">
  <Target id="p0" name="p0" runat="local">
    <Optimizer id="p0" name="simpleGA0" runtime="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>
      <criteria on="or" decision="objective" type="ratio"
        value="0.1"/>
      <criteria on="or" decision="objective" type="deviation"
        value="0.01"/>
    </Converge>
    <Target id="p1" name="p1" runat="local">
      <Optimizer id="p0" name="simpleGA1" runtime="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>
        <criteria on="or" decision="objective" type="ratio"
          value="0.1"/>
        <criteria on="or" decision="objective"
          type="deviation" value="0.01"/>
      </Converge>
    </Target id="p3" name="p3" runat="local">
    <Optimizer id="p3" name=" simpleGA3" runtime="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>
      <criteria on="or" decision="objective" type="ratio" value="0.1"/>
      <criteria on="or" decision="objective" type="deviation" value="0.01"/>
    </Converge>
  </Target>
  <Target id="p4" name="p4" runat="localhost:4446">
    <Optimizer id="p4" name=" simpleGA4" runtime="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>
      <criteria on="or" decision="objective" type="ratio"
        value="0.1"/>
      <criteria on="or" decision="objective" type="deviation"
        value="0.01"/>
    </Converge>
  </Target>
  <Target id="p2" name="p2" runat="local">
    <Optimizer id="p2" name=" simpleGA2" runtime="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="x14" lower="0"/>
      <decision type="dependent" name="x6" lower="0"/>
    </Decisions>
    <Converge>
      <criteria on="and" decision="objective" type="ratio"
        value="0.1"/>
      <criteria on="or" decision="objective" type="deviation"
        value="0.01"/>
    </Converge>
  </Target>
</ATC>

```

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

294 3.3 Defining common and cascading decision 295 variables

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

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

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

317 3.4 Defining response targets

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

325 3.5 Defining targent optimizers

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

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

337 This assumption has simplified the atcXML definition to
338 a large extent. For example, Targents (TopAgents as their
339 run-time components) in the case study use the Optimizers
340 that are based on genetic algorithm (GA). Five of such
341 Optimizers have been developed and deployed as web

services. It is assumed that the provider of an Optimizer
342 register necessary information in order for this Optimizer to
343 be called on the Internet. Such information may include the
344 Internet location, input and output data templates, and
345 parameters for setting up the optimization. 346

The “Optimizer” tag is used for defining the Targent’s
347 Optimizer, as in lines 4–7. The Optimizer “simpleGA” in
348 the above definition is an optimization web service built
349 with simple/basic GA and registered in smoUDDI. 350
Parameters defined are specifically associated with the
351 optimization method. They are passed to the Optimizer
352 when it is called upon. In this case, “Population”,
353 “Generation” and “Penalty” (on inequality constraints)
354 are the three parameters for simpleGA. The convergence
355 and/or stop criteria may be explicitly defined as parameters
356 or derived from the parameters defined. For example,
357 simpleGA will stop after a maximum generation of 5,000 if
358 other convergence criteria do not become valid. 359

360 3.6 Defining targent convergence criteria

A Targent is said to converge if and only if its local solution
361 is consistent with the solution obtained from the results
362 backtracked from its children. The meaning of “consistent”
363 refers to the satisfaction of the convergence criteria defined
364 explicitly by the pair of <Converge> ...</Converge>
365 brackets as in lines 30–33. The Targent converges when
366 all compulsory criteria with attribute “and” must be met
367 while any of the criteria with “or” attribute is satisfied. 368

The criterion defined in line 31 can be interpreted as
369 where value=0.1 and criterion defined in line 32 as where
370 value=0.01. These two criteria are defined for the overall
371 objective function. Similar criteria may be defined for
372 linking variables, in addition to the overall objective
373 function of the Targent. 374

It is important to note the complete difference between
375 the convergence of the Targent and its Optimizer. Assume
376 the Optimizer is based on a simple GA that is asked to
377 converge into the best solution after a given number of
378 generations. The convergence criteria of the Optimizer are
379 only implicitly defined in atcXML through the Optimizer’s
380 parameters. The Optimizer must interpret them as de-
381 signed. 382

4 atcEngine: atcPortal engine 383

atcEngine in the atcPortal is the central mechanism that
384 creates, starts, monitors and coordinates the participating
385 TopAgents to accomplish the computational ATC process
386 as defined in the atcXML file. atcEngine is basically a
387 multi-agents system (MAS) consisting of a community of
388 TopAgents. TopAgents are run-time implementation of
389 Targents as defined in atcXML. The TopAgent community
390 Organizer is responsible for creating TopAgents from
391 Targents as instances of the TopAgent Java class. Top-
392 Agents are presently implemented in Java and each
393 TopAgent will run in a separate thread. TopAgents are
394

395 supposed to be “thin” in the sense that they provide
 396 containers for the corresponding Optimizers which are
 397 supposed to be “fat” and consuming substantial computa-
 398 tional resources. Therefore, TopAgents are not designed to
 399 run in a distributed computational environment but Opti-
 400 mizers are.

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

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

411 4.1 TopAgent properties

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

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

424 4.2 TopAgent status

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

- 427 – Idle: A TopAgent is not asked to do anything yet.
- 428 – Optimizing: A TopAgent has called its Optimizer to
 429 solve the local ATC optimization problem and is
 430 waiting for the result.
- 431 – Cascading: A TopAgent has cascaded downwards its
 432 children TopAgents and is waiting for their feedback.
- 433 – Backtracking: A TopAgent has converged and back-
 434 tracked upwards its parent TopAgent.
- 435 – Succeeding: All TopAgents reach the “Succeed” status
 436 when the top-level TopAgent converges.
- 437 – Failing: A TopAgent fails when it does not have a
 438 solution to its local target optimization problem or any
 439 of its children backtracks with “failure”.

440 4.3 Messages and message box

441 Each TopAgent maintains a message box. When a
 442 TopAgent needs to interact with another TopAgent, it
 443 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"> 453
  <Decision name="x3">2.00</Decision> 454
  <Decision name="x11">1.00</Decision> 455
  ... 456
</Cascade> 457
```

Likewise, the purpose of a Backtrack message is to
 submit the values achieved upon the TopAgent conver-
 gence 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"> 463
  <Decision name="x3">2.01</Decision> 464
  <Decision name="x11">1.01</Decision> 465
  ... 466
</Backtrack> 467
```

The Optimize messages are internal in the sense that
 they are exchanged between the TopAgent and its Op-
 timizer 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"> 484
  <Decision name="x3:uvalue">1.12</Decision> 485
  <Decision name="x11:uvalue">2.21</Decision> 486
  <Decision name="x8:lvalue">0.98</Decision> 487
  <Decision name="x8:old">0.85</Decision> 488
  ... 489
</Request> 490
```

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/ser- 494
vices/SGA1"> 495
```

```

496 <Decision name="x3"></Decision>
497 <Decision name="x11"></Decision>
498 <Decision name="x8"></Decision>
499 <Decision name="x9"></Decision>
500 <Decision name="x10"></Decision>
501 <Decision name="objective"></Decision>
502 <Decision name="et"></Decision>
503 <Decision name="ey"></Decision>
504 <Decision name="er"></Decision>
505 ...
506 </Response>
    
```

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

510 4.4 TopAgent Methods

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

517 The first method called by the messenger is the property
 518 manager method. It reads a message and updates the pro-
 519 perties according to the message type and body information.

520 The Cascade and Backtrack methods are triggered by the
 521 messenger method if the incoming message type is Cas-
 522 cascade and Backtrack, respectively. How Cascade and Back-
 523 track methods perform the tasks are governed by the

524 chosen ATC control strategy. There are four ATC strategies
 525 as discussed by Michelena et al. [12]. At present, only
 526 Strategy IV is used because it has been used in most of the
 527 reported case studies. In Strategy IV of ATC, the Cascade
 528 method and the Backtrack method follow the procedure
 529 outlined in Fig. 3.

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

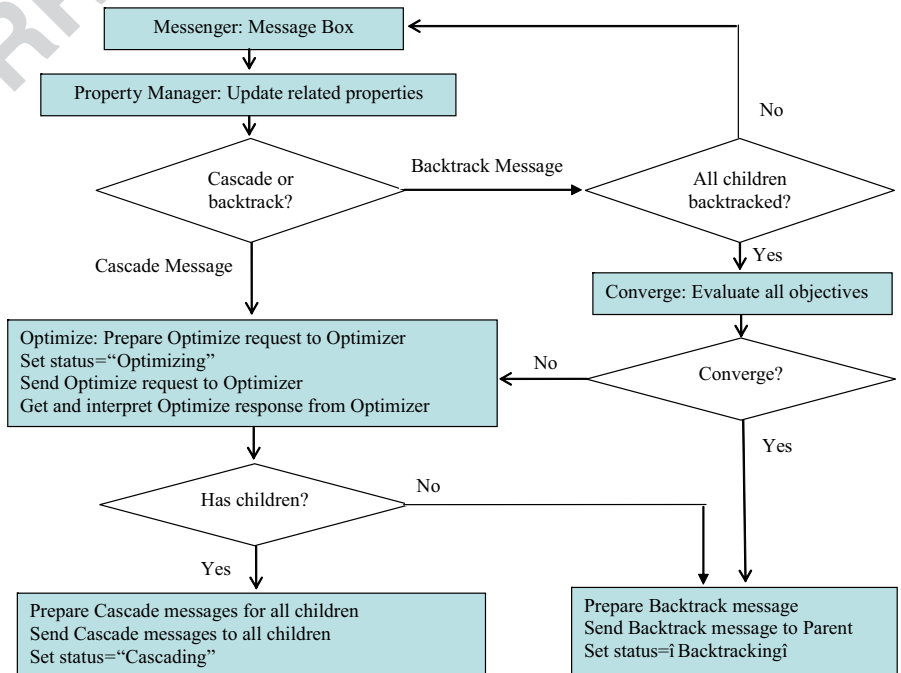
534 If all the children have backtracked, the Backtrack meth-
 535 od triggers the Converge method. The Converge method is
 536 responsible (1) for assessing the values of the different
 537 terms included in the TopAgent's objective function from
 538 the children's feedback data, and (2) for checking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

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

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

Print will be in black and white

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



577 5 smoUDDI: atcPortal's UDDI directory of SMOWS

558 When formulating the ATC problem mathematically, [6]
 559 distinguishes between optimal design models and analysis
 560 models. Optimal design models call analysis models to
 561 evaluate responses from corresponding system compo-
 562 nents. Analysis models take lower level decision variables
 563 and parameters, as well as responses from lower level
 564 components in the system hierarchy, and return responses
 565 for optimization problems of upper level system compo-
 566 nents. Technically, analysis models are equivalent to
 567 equality constraints defined for an optimization problem.

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

583 smoUDDI is built on various web services tools such as
 584 UDDI, SOAP, and WSDL. UDDI is layered over SOAP and
 585 assumes that requests and responses are UDDI objects sent
 586 around as SOAP messages. The web services description
 587 language (WSDL) is a general purpose XML language for
 588 describing the interface, protocol bindings and the deploy-
 589 ment details of network services. WSDL is used to describe
 590 what a web service can do, where it resides, and how to
 591 invoke it. WSDL complements the UDDI standard by pro-
 592 viding a uniform way of describing the abstract interface
 593 and protocol bindings of arbitrary network services. WSDL
 594 assumes SOAP/HTTP/MIME as the remote object invoca-
 595 tion mechanism. UDDI registries describe numerous aspects
 596 of web services, including the binding details of the service.
 597 WSDL fits into the subset of a UDDI service description.

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

608 5.1 Optimizer registration with smoUDDI

609 In this stage, two types of information are published on
 610 smoUDDI by the optimizer provider. The first is the op-
 611 timizer's functional information, which includes its apply-

ing 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 com-
 pletely define a Targent XML node for the atcXML
 definition file, the optimizer, which represents the opti-
 mization solution methods of this Targent, must be simulta-
 neously 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 classifica-
 tion 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 an-
 alytical target cascading. Its practical contribution is that
 ATC analyses can be easily set up and conducted with the

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

673 Finally, although many of the limitations that this pro-
 674 totype atcPortal system suffers at present can be easily
 675 improved, it is important to point out a fundamental lim-
 676 itation on the balance between the generality and ex-
 677 tensibility as set out in the title of the paper. Optimizers as
 678 web services collected in the smoUDDI are supposed to be
 679 generally reusable regardless if they are used in ATC
 680 analyses or for other applications. At present, they contain
 681 ATC-specific elements. In order to overcome this limita-
 682 tion, it is necessary to introduce two tiers of web services:
 683 one for Optimizers which are allowed to contain ATC-
 684 specific elements to enhance their extensibility, and the
 685 other for response/analysis models without ATC-specific
 686 elements to improve their generality. This is the area
 687 requiring substantial research and development efforts.

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

692 References

693 1. Choudhary R, Malkawi A, Papalambros PY (2003) A
 694 hierarchical design optimization framework for building
 695 performance analysis. Proceedings of the 8th international
 696 building performance simulation association (IBPSA) confer-
 697 ence, Eindhoven, Netherlands, 11–14 August 2003
 698 2. Etman LFP, Kokkolaras M, Papalambros PY, Hofkamp AT,
 699 Rooda JE (2002) Coordination specification for analytical
 700 target cascading using the Chi language. Proceedings of the 9th
 701 AIAA/ISSMO symposium on multidisciplinary analysis and
 702 optimization, work in progress, AIAA–2002–5637, Atlanta,
 703 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 multi-
 disciplinary optimization, Dalian, China, 4–8 June 2001
 704 705 706 707
 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
 708 709 710
 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
 711 712 713 714
 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
 715 716 717 718
 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
 719 720 721 722 723
 8. Kim HM (2001) Target cascading in optimal system design.
 PhD Dissertation, Department of Mechanical Engineering,
 University of Michigan, Ann Arbor, Michigan, USA
 724 725 726
 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 mechan-
 ical engineering congress and exposition, New Orleans,
 Louisiana, 17–22 November 2002
 727 728 729 730 731 732
 10. Mahmoud HA, Kabamba PT, Ulsoy AG, Brusher GA (2002)
 Target reduction and balancing using system norms. Proceed-
 ings of the American control conference, Anchorage, AK, May
 2002
 733 734 735 736
 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
 737 738 739 740
 12. Michelena N, Park H, Papalambros P (2003) Convergence
 properties of analytical target cascading. AIAA J 41:897–905
 741 742
 13. Nyström M, Larsson T, Karlsson L, Kokkolaras M, Papalam-
 bros 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
 743 744 745 746 747 748 749