MULTI-LEVEL DECOMPOSED SYSTEMS DESIGN: CONVERTING A REQUIREMENT SPECIFICATION INTO AN OPTIMIZATION PROBLEM

Beernaert, T. F.; Etman, L. F. P.

Eindhoven University of Technology

ABSTRACT

Complex technological artefacts are often decomposed into smaller components to keep their design manageable. The resulting challenge is to coordinate decisions that involve multiple components and to design components such that high-level targets are met. Analytical Target Cascading (ATC) is an analytical coordination method for the optimization of decomposed systems, which we aim to incorporate in systems engineering design process. To this extent, we relate the domain of engineering optimization to the domain of requirements engineering, and propose a method that constructs an ATC problem from functional specifications and requirements written in the newly developed Elephant Specification Language. The proposed method is demonstrated in the two-level design of an automotive powertrain. This contribution is a step towards design automation and is expected to increase the usability of decomposed optimization techniques.

Keywords: Systems Engineering (SE), Integrated product development, Design Automation, Requirements, Optimisation

Contact:
Beernaert, Torben Frans
Eindhoven University of Technology
Mechanical Engineering
The Netherlands
torbenbeernaert@gmail.com

Cite this article: Beernaert, T.F., Etman, L.F.P. (2019) 'Multi-level Decomposed Systems Design: Converting a Requirement Specification into an Optimization Problem', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.376

1 INTRODUCTION

The rising complexity in technological systems poses many challenges for their management and design. The field of systems engineering is concerned with the life cycle of complex engineered systems and provides various techniques to manage complexity. A popular principle is decomposition, the process of splitting a whole system into smaller components. However, the challenge of designing a decomposed system is that the relation between the complete system and its components can be obscure or even unknown. This makes it difficult to make decisions that affect multiple components and to predict how changes in one component relate to other components and to the system as a whole. There exist multiple subdomains of systems engineering that aim to support the development of decomposed systems.

The requirements engineering subdomain supports the establishment and management of requirements. A recently proposed language from this domain, the Elephant Specification Language (ESL), enables the specification of functions, requirements and dependencies for multi-level decomposed systems (Wilschut, 2018). ESL can visualize the relations between system components and can provide mappings between requirements and design variables.

Analytical models are used in every design stage to predict the behaviour and properties of the system. As the design progresses, separate highly detailed models will be developed that describe different parts of the system. Considering separate models increases practicality, but it becomes difficult to overview quantitative trade-offs that involve multiple system components. The engineering optimization subdomain supports the analytical design phase with Analytical Target Cascading (ATC), a quantitative coordination method for the optimal design of decomposed systems. Literature has mainly focused on numerical properties of ATC, but has never opted how a decomposed optimization problem should be used in the context of systems engineering design. Considering ATC from a design perspective could be a step towards its integration in real-world applications.

ESL and ATC show similar perspectives and descriptions, and are expected to be suitable for a joint integration in systems engineering design. Combining these techniques will relate a multi-level functional system specification to a decomposed optimization problem, and is expected to support the transition from requirements to analytical design in both early and late design phases. The hypothesis of this contribution is that an ATC optimization problem can be constructed from an ESL specification.

We propose a method to convert an ESL specification into an ATC problem, and conclude that a functional system description in ESL can provide most information that is necessary for the definition of an ATC optimization problem. The multi-level ESL decomposition tree can be converted into an equivalent ATC tree of optimization subproblems, and ESL requirements can be interpreted as engineering optimization constraints. We expect that the conversion method can derive an optimization problem from a requirements specification in consecutive stages of multi-level systems design.

In Section 2, we elaborate decomposed systems engineering and the paradigms that are considered in this work. Sections 3 and 4 elaborate ESL and ATC, respectively, and highlight their similarities. In Section 5, we propose how to convert an ESL specification into an ATC problem, which is demonstrated in Section 6 by the two-level design of a vehicle powertrain. We conclude this contribution with final remarks in Section 7.

2 MULTI-LEVEL SYSTEMS DESIGN

Perhaps the most popular method for the design of complex systems is the systems engineering V-model, proposed by Forsberg and Mooz (1991). The V-model advocates that the analysis and design of the system increase in detail as time progresses. However, considering too much detail leads to complex and unmanageable design tasks. Therefore, the V-model opts to decompose a system into its subsystems when one system becomes too complex. This principle can be applied on consecutive levels, thereby generating a hierarchical decomposition tree. This schematic tree is a view of the structure of a system, involving multiple levels of detail. The system is broken down into its components that are distributed over decomposition levels, such that a parent component on level i consists of its children on level $i + 1$. Generally, systems engineers aim to decompose the system in such a way that the resulting components can be analyzed and designed separately. However, there are always trade-offs and decisions that affect multiple components, which require coordination among them. Design Structure Matrices (DSMs) are visualizations of the interactions between the components of a decomposed system, and can support the management of its development (Eppinger and Browning, 2012).

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

The requirements engineering domain is concerned with the management of requirements (Hull, Jackson and Dick, 2005; Young, 2004). Wilschut (2018) proposed ESL as a language for the specification of functional requirements. ESL aids the coordination of multi-level systems by generating DSMs at various levels of decomposition. It specifies the relations between components, functions and variables, but cannot incorporate analytical models.

The engineering optimization domain does consider analytical system models and aims to find a design that minimizes a mathematical objective function. ATC is a technique for the optimal design of multi-level decomposed systems, developed by Kim *et al.* (2003), and has been demonstrated by several examples in literature (see Papalambros and Wilde (2017) for references). It is the only optimization technique that can consider a multi-level problem architecture, and is therefore expected to be suitable for the analytical design of systems that are decomposed into multiple hierarchical layers. The numerical properties of ATC have been thoroughly investigated, but we have never seen how an ATC problem relates to other phases of systems design or how it can be constructed.

Establishing requirements and conducting analyses are steps that should be repeated on every level of decomposition, but are subject to different domains. Combining these domains is a step towards design automation and is expected to increase the traceability, efficiency and quality of the development. Therefore, we aim to convert elements from multi-level requirements engineering into multi-level engineering optimization. Our proposition is that a combination of ESL and ATC can provide a mathematical framework for the integrated design of decomposed systems. This development will benefit the domains of requirements engineering and engineering optimization, as well as systems engineering in general.

3 ELEPHANT SPECIFICATION LANGUAGE

ESL is a recently developed language for the specification of the structure, functionality and requirements of multi-level decomposed systems (Wilschut, 2018, Chapter 6). This language enables the definition of components in a hierarchical decomposition tree via explicit parent-child relations. Besides this vertical coupling, ESL is capable of specifying the interactions between components at the same level of decomposition. Analysis of an ESL specification can automatically yield several DSMs (Eppinger and Browning, 2012) that can support the management of the system at hand. Listing 3 shows an example of an ESL specification.

Listing 1 An ESL specification of a powertrain and its child components.

```
# Level 1 ------------------------------------------------------------
define component powertrain
    parameters
        mechanical-power is a DesignVariable
    variables
        energy, mass, time, heat, electrical-power is a DesignVariable
        power-source-mass, cooling-mass, drive-mass is a DesignVariable
    components
        Powersource is a battery with arguments
            electrical-power, energy, time, power-source-mass
        Coolingsystem is a cooling-system with arguments
            heat, cooling-mass
        Drivetrain is a DC-torque-converter with arguments
            electrical-power, mechanical-power, heat, drive-mass
    design-requirement
        dsr-0: mechanical-power must be at least 40e3 W
        dsr-1: time must be equal to 1200 s
    goal-requirement
        gfr-0: Powersource must provide electrical-power to Drivetrain
        gfr-1: Drivetrain must provide heat to Coolingsystem
    transformation-requirement
        tfr-0: must convert energy into mechanical-power
    relation
        rlt-0: Composition with arguments
            mass, power-source-mass, cooling-mass, drive-mass

```


https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

```
27 # Level 2 ------------------------------------------------------------
28 define component battery
29     parameters
30         electrical-power, energy, time, mass is a DesignVariable
31     transformation-requirement
32         tfr-0: must convert energy into electrical-power with subclause
33             c-0: electrical-power should be at most 80e3 W
34     relation
35         rlt-0: BatteryModel with arguments
36             electrical-power, energy, time, mass
37
38 define component cooling-system
39     parameters
40         heat, mass is a DesignVariable
41     variables
42         dissipation is a DesignVariable
43     transformation-requirement
44         tfr-0: must convert heat into dissipation with subclause
45             c-0: dissipation must be at least heat
46     relation
47         rlt-0: CoolingModel with arguments
48             mass, dissipation
49
50 define component DC-torque-converter
51     parameters
52         electrical-power, mechanical-power, heat, mass is a DesignVariable
53     variables
54         efficiency is a DesignVariable
55     transformation-requirement
56         tfr-0: must convert electrical-power into mechanical-power, heat
57     relation
58         rlt-0: DrivetrainModel with arguments
59             electrical-power, mechanical-power, heat, mass, efficiency
```

A specification comprises component definitions that contain variables, child component instances, requirements and relations. Variables that are defined in a parent component may be passed to a child component via arguments (lines 10, 12 & 14); the same variables have to be defined as parameters in the corresponding child component (lines 30, 40 & 52). As such, ESL specifies a decomposition tree where components are linked strictly hierarchically via variables. This multi-level hierarchy is shared by ATC.

ESL specifies the functionality of a component via goal and transformation functions. Goal functions relate two components via a variable (lines 19 & 20). This variable is considered an interface, and is always passed to both connected child components via arguments. Transformation functions describe a component's internal working principle by the transition of variables (lines 22, 32, 44 & 56). ESL uses goal and transformation functions to construct a component DSM.

Design requirements specify conditions by comparing a variable to a numeric constant or to another variable (lines 16 & 17). The comparison is written in natural language, but can be interpreted as a mathematical operator ($\leq, <, =, >$ or $\geq$). Goal and transformation functions can be refined by subclauses (line 45) in the same textual format as design requirements. Design requirements and subclauses closely resemble engineering optimization constraints because of the mathematical comparison operator.

Finally, ESL enables the specification of relations between variables (lines 24-25, 35-36, 47-48 & 58-59). A relation implies that arguments depend on each other via a named mathematical function. For instance, the relation in lines 24-25 specifies that there is a function, called `Composition`, that depends on the mass of the powertrain and its children. ESL does not define the actual mathematical function, but only specifies its existence.

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

4 ANALYTICAL TARGET CASCADING

ATC is a coordination method for the optimization of decomposed systems, and was initially developed as a way to propagate high-level targets to low-level components (Kim *et al.*, 2003). ATC considers decomposed system design as an optimization problem that is split into subproblems and represented in a multi-level architecture. Figure 1 displays the architecture of a three-level ATC problem. Subproblem P_{ij} is located on decomposition level i, where j is an index for subproblems at level i.

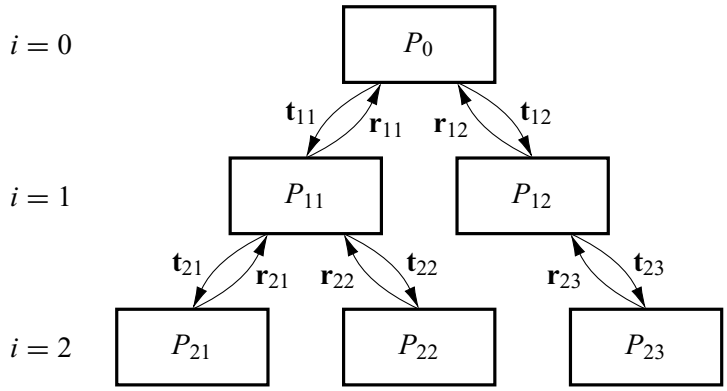

Figure 1. An ATC optimization problem comprises subproblems in a multi-level hierarchy. Subproblems are linked via target and response variables $\mathbf{t}_{ij}$ and $\mathbf{r}_{ij}$, respectively.

Constraints are assigned to single subproblems, but variables may be shared by multiple subproblems and thereby link them in the hierarchy. Parent subproblems iteratively send target values to their children, who send response values back up in an effort to reach their acquired targets. Every subproblem has a local objective function, f_{ij}. The sum of these local objective functions is equal to the objective function of the original optimization problem. The general formulation of subproblem P_{ij} is given by:

$$\begin{aligned} \min_{\bar{\mathbf{x}}_{ij}} \quad & f_{ij}(\bar{\mathbf{x}}_{ij}) + \phi(\mathbf{c}_{ij}) \\ \text{subject to} \quad & \mathbf{g}_{ij}(\bar{\mathbf{x}}_{ij}) \leq \mathbf{0}, \quad \mathbf{h}_{ij}(\bar{\mathbf{x}}_{ij}) = \mathbf{0} \\ \text{where} \quad & \bar{\mathbf{x}}_{ij} = \left[\mathbf{x}_{lij}^T, \mathbf{t}_{(i+1)k_1}^T, \ldots, \mathbf{t}_{(i+1)k_{nc_{ij}}}^T, \mathbf{r}_{ij}^T\right]^T \\ & \mathbf{c}_{ij} = \left[(\mathbf{t}_{ij} - \mathbf{r}_{ij})^T, (\mathbf{t}_{(i+1)k_1} - \mathbf{r}_{(i+1)k_1})^T, \ldots, (\mathbf{t}_{(i+1)k_{nc_{ij}}} - \mathbf{r}_{(i+1)k_{nc_{ij}}})^T\right]^T \end{aligned} \tag{1}$$

Every subproblem has a local objective function f_{ij} and local inequality and equality constraints $\mathbf{g}_{ij}$ and $\mathbf{h}_{ij}$, respectively. Subproblem P_{ij} has access to local design variables $\mathbf{x}_{lij}$, targets $\mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{nc_{ij}}}$ to set for its $n_{c_{ij}}$ children, and responses $\mathbf{r}_{ij}$ to send to its parent. Consistency constraints $\mathbf{c}_{ij}$ require that all corresponding targets and responses are equal to each other, but are relaxed via a penalty function ϕ. The state-of-the-art ATC algorithms utilize an Augmented Lagrangian Coordination (ALC) penalty function (Tosserams *et al.*, 2006; Xu, Fadel and Wiecek, 2017). The linear and quadratic weights of the ALC penalty function are updated via schemes that drive $\mathbf{c}_{ij}$ to zero.

ATC is a local optimization technique; for convex problems, the solution converges to a globally optimal system design (Michelena, Park and Papalambros, 2003). If the posed optimization problem is non-convex, one cannot guarantee that ATC converges. If it does converge, only local optimality is ensured. We expect that ATC can provide a framework to couple separate analytical models. We envision that ATC can be used in subsequent stages of design, where an expanding decomposition tree leads to a larger and more complex ATC problem. Algorithms such as ALC could then coordinate trade-offs among these models to arrive at an optimal system design.

5 PROPOSED CONVERSION METHOD

Similarities between ATC and ESL have led to the hypothesis that an ESL specification can be converted into an ATC problem. We aim to exploit these similarities (i.e., the multi-level architecture, variable coupling and definition of variables and functions) in the proposed conversion method. Our

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

most fundamental presumption is that every component defined in ESL can be converted into an ATC subproblem, so the hierarchical structure of the ESL component decomposition tree will be identical to the ATC problem structure. In this section, we analyse the elements that comprise the ESL specification of component concept c_{ij} and translate them to optimization problem P_{ij}. Figure 2 summarizes the proposed mapping between ESL and ATC, and demonstrates a simple conversion of an ESL specification into an ATC subproblem. We will refer to this example in the following sections to elicit the concept.

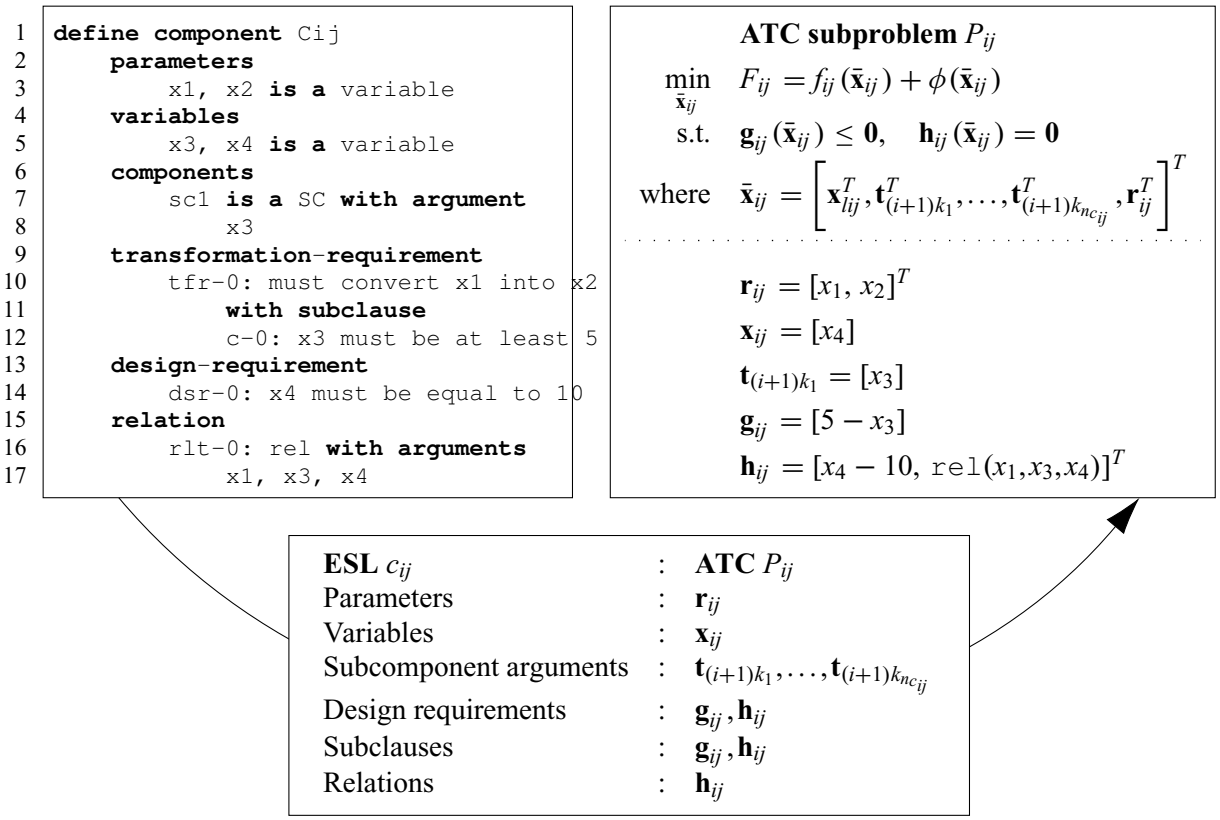

Figure 2. Conversion from the ESL specification of component c_{ij} into ATC subproblem P_{ij}.

5.1 Variables

Components contain a pool of variables, some of which link them to their children and/or to their parents, as explained in the previous section. Our interpretation is that all parameters and variables are actually design variables in the context of ATC. We read the 'parameters' section of a component definition as a 'variables that are linked to my parent' section. This structure enables a perfect mapping to the ATC paradigm; The pool of variables is the vector $\bar{\mathbf{x}}_{ij}$, which consists of local variables $\mathbf{x}_{lij}$, target variables $\mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{nc_{ij}}}$ and response variables $\mathbf{r}_{ij}$. In the given example, child component `sc1` will declare one parameter in its specification, corresponding to `x3`. Vice versa, component c_{ij} has been instantiated in its parent component, with two arguments corresponding to `x1` and `x2`. From the perspective of c_{ij}, x_3 is a target to its child `sc1` ($\mathbf{t}_{(i+1)k_1}$) and x_1 and x_2 are responses to its parent ($\mathbf{r}_{ij}$).

5.2 Design requirements and subclauses

Design requirements and subclauses use a comparison operator in written language to compare a variable to a constant or to another variable. Depending on the operator, we translate subclauses and design requirements into inequality $\mathbf{g}_{ij}$ (line 12) and equality $\mathbf{h}_{ij}$ (line 14) constraints according to Table 1.
ESL allows designers to use specific verbs, e.g., must, should or could, in the specification of design requirements and subclauses to distinguish subtle priority difference. The implied priorities are currently not recognized by the conversion; All design requirements and subclauses are treated as constraints. The interpretation of priorities in an engineering optimization problem could be a future development of the ESL to ATC conversion.

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

Table 1. Interpretation of subclauses and design requirements as ATC constraints.

ESL specification	Interpreted equation	ATC constraint
A ... equal to B	$A - B = 0$	Equality
A ... at most B	$A - B \leq 0$	Inequality
A ... at least B	$B - A \leq 0$	Inequality

5.3 Relations

Each ESL relations specifies that there exists a mathematical function that must be considered in the analysis of the component. We interpret relations as mathematical functions that must be true, which can be expressed by equality constraints. Considering that any mathematical relation can be rearranged such that its right-hand side is equal to zero, relations can be formulated as equality constraints. For instance, Newton's second law, that relates the arguments force F, mass m and acceleration a, would be interpreted as the equality constraint $h = F - ma = 0$. ESL does not define the mathematical function associated to a relation. Therefore, line 16 only tells us that there is a function `rel` with three arguments, x_1, x_3 and x_4, that should satisfy: $h = \texttt{rel}(x_1, x_3, x_4) = 0$. It is expected that other interpretations can lead to a reduced optimization problem, i.e. with less variables and constraints, that will be more efficient to solve. For such principles of optimal design, we refer to Papalambros and Wilde (2017).

5.4 Goal and transformation functions

Goal and transformation functions are not directly interpreted. A goal function describes a relation between two components, and always has to be accompanied by passing the associated interface variable to both components. Transformation functions specify the internal working of a component by linking several variables, but have to be further elaborated by a relation. Therefore, analysing component arguments and relations is sufficient to capture the effects of goal and transformation functions from an analytical design perspective. Goal and transformation functions can form a basis to generate a component DSM (Wilschut, 2018, Chapter 4). According to our conversion concept, the elements in this component DSM translate to linking variables in ATC.

5.5 Additional input

Considering subproblem P_{ij} in Figure 2, the specification of a local objective function f_{ij} is necessary to form a well-defined optimization problem. The ESL paradigm currently does not provide a way to specify this function. From an engineering optimization point of view, the incorporation of an objective element in ESL is an essential recommendation for its development. Furthermore, ESL only specifies that there exists a relation between variables, but does not define the actual mathematical relation. That is, the function `rel` in line 16 has to be defined in order to properly formalize an ATC problem.
An ESL specification can be converted into an ATC subproblem, under the conditions that an objective function and mathematical definitions of ESL relations are provided. Similarities between the two enabled a structured overlap of the specification and analysis phases in systems engineering design, which are essential in the design of complex decomposed systems.

6 DEMONSTRATION

In this section, we demonstrate the proposed conversion method in the two-level design of an automotive powertrain, specified in Listing 1.

6.1 Analytical target cascading problem

A powertrain subproblem is generated at the top level, while the bottom level consists of the child components' subproblems. The ESL variables are directly translated to ATC variables. Linking variables are established from child arguments and parameters. We separately add a local objective to the overarching powertrain subproblem, namely to minimize its total mass m. Design requirements and subclauses translate to explicit constraints (two in the powertrain subproblem and one in the power source subproblem), while relations lead to implicit equality constraints (one in each subproblem). A single relation may involve multiple calculations and can therefore impose multiple equality constraints. Figure 3 displays the resulting ATC problem.

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

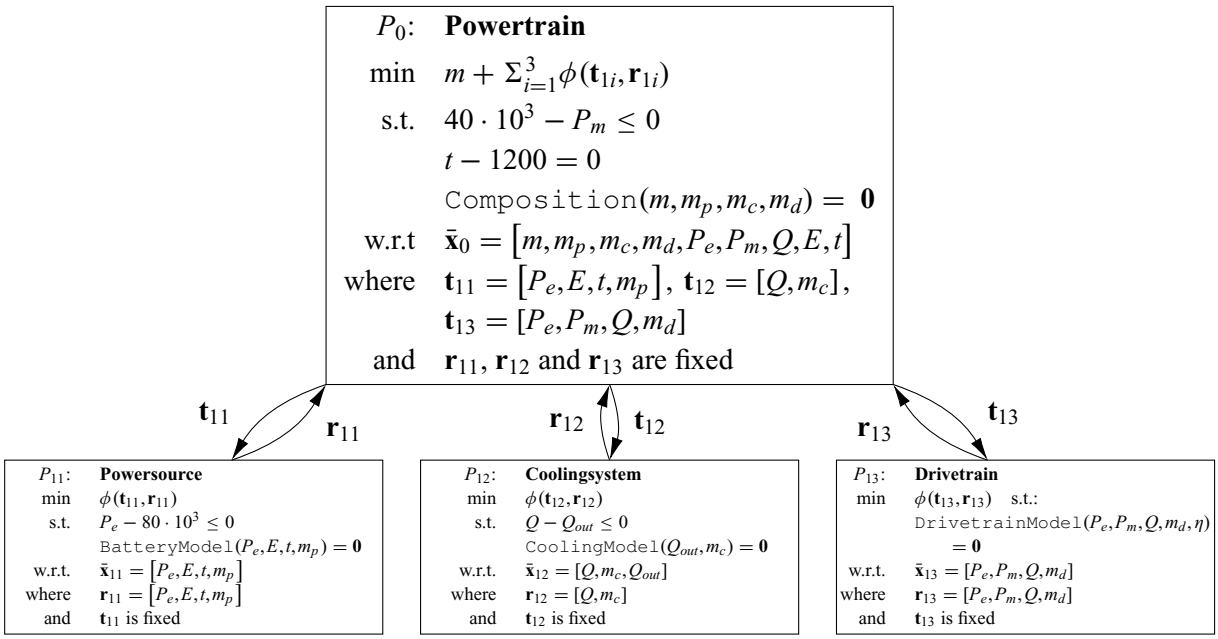

Figure 3. Two-level ATC problem, automatically generated from the ESL specification in Listing 1.

6.2 Dependency structure matrix

The goal function specifications in ESL can be analyzed to establish a DSM, as shown by (Wilschut, 2018, Chapter 4). This DSM contains the decomposition that has been specified in ESL, but can be analyzed and clustered to identify an improved decomposition. Usually, clustering methods rearrange components into groups with few interactions, and presume that the coordination effort that is necessary for their integrated design reduces. Clustering a DSM leads to a different hierarchical system structure and therefore, following the proposed method, to a different ATC optimization problem. We expect that the effects of a DSM clustering action can be observed in the coordination effort of the equivalent ATC algorithm, which can be quantified as the number of iterations to achieve an optimal design.

Figure 4 shows a component DSM that is generated from Listing 1. It shows the interactions between the child components of the powertrain, specified by goal functions. Goal function `gfr-0` links the `Powersource` to the `Drivetrain`, which is represented in the DSM by elements 2-3 and 3-2. Similarly, `gfr-1` links `Drivetrain` to the `Coolingsystem` via elements 1-2 and 2-1.

From an analytical perspective, the interface variable in a goal function specification needs to be considered in the design of both connected components. To this extent, the interface variable is passed to the connected components via ESL arguments and, according to our conversion concept, converted into ATC linking variables. This means that all elements in a DSM are interpreted as linking variables and coordinated by the common parent of both connected components.

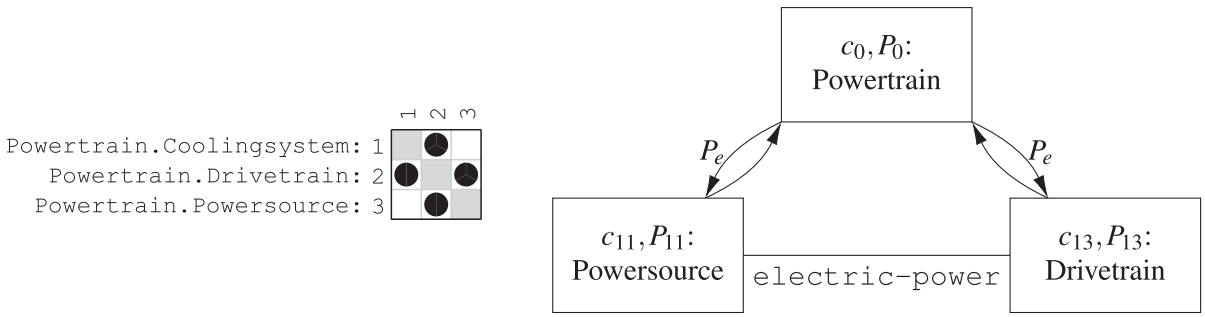

Figure 4. A level two component DSM, automatically generated from Listing 1 (left). DSM interfaces (e.g. `electric-power`) are coordinated via linking variables (e.g. P_e) to their common parent (right).

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

The detail of the system increases as the design progresses. The strength of ESL lies in its flexibility towards lower levels of decomposition, as it enables the user to add more layers of components to the decomposition tree while maintaining high-level specifications. Figure 6.2 shows a DSM in a subsequent design stage, where two child components are defined for every component in the level two DSM in Figure 4. The interface `electric-power` between the `Powersource` and the `Drivetrain` automatically migrated to their children, `Inverter` and `ElectricMotor` respectively.

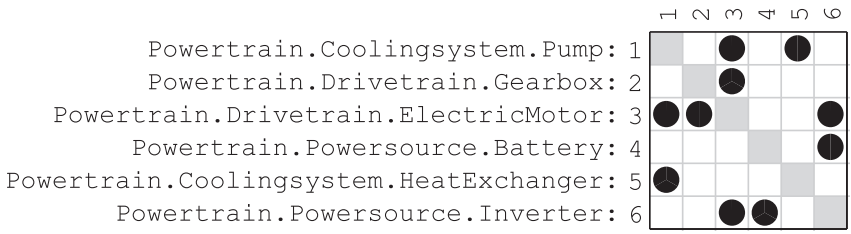

Figure 5. A level three component DSM that is generated from Listing 1 after adding the specification of a new decomposition layer.

6.3 Optimization results

The ATC problem in Figure 3 is solved using the alternating direction method of multipliers, as presented by Tosserams *et al.* (2008). The termination criterion is set to $\epsilon = 1 \cdot 10^{-2}$ and the weight update parameters are set to $\beta = 1$ and $\gamma = 0.5$. The solution converges in 225 iterations. Table 2 shows the mapping between ESL and ATC variables, and their initial and optimized values.

Table 2. Analytical results of a two-level ATC optimization.

ESL	ATC	Initial	Optimized	Unit
`mass`	m	60	140.44	kg
`power-source-mass`	m_p	30	27.68	kg
`cooling-mass`	m_c	10	2.19	kg
`drive-mass`	m_d	20	110.56	kg
`mechanical-power`	P_m	40	40	kW
`electrical-power`	P_e	40	41.43	kW
`heat`	Q	2	1.379	kW
`energy`	E	5	13.84	kWh
`time`	t	1000	1200	s
`dissipation`	Q_{out}	2	1.481	kW
`efficiency`	η	0.90	0.97	-

The ATC algorithm successfully coordinates the value of the specified linking variables among the related components. For example, the `electrical-power` is involved in the analysis of both the `Powersource` and the `Drivetrain`, but converges to a value that is optimal for the overall system.

6.4 Discussion

We have shown how an ESL specification can be converted into a DSM and an ATC problem. The problem is solved and we have acquired quantitative component designs that are system-optimal. As the decomposition tree expands in later design stages, so will the equivalent ATC problem.

With the presented example, we aim to demonstrate that there is a possibility to integrate mathematical optimization techniques in the design of complex engineered systems. Because of the preliminary nature of this method, a rather simple example has been used for the demonstration. The demonstration features a convex optimization problem and may not be representative for a real-world design problem. In general, it is difficult to test new systems engineering ideas and methods because, by definition, the cases should be extensive and complex. Requirements, conditions and goals for systems engineering methods, based on large-scale design problems, may be extremely valuable for the assessment and development of new techniques in this domain.

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press

7 CONCLUSIONS

A key challenge in the design of complex engineered systems is the coordination of trade-offs and changes that involve multiple components of a system. Distributed optimization techniques can use analytical models of system components to compute design candidates that are mathematically optimal for the system as a whole. Because it has never been shown how such a distributed optimization problem can be constructed, we have presented an approach how to extract an ATC problem formulation based on an ESL specification from the requirements engineering domain.

Every instantiated ESL component is converted into an ATC subproblem, leading to an identical structural decomposition hierarchy. Component arguments and parameters in ESL translate to target and response variables in ATC, ESL design requirements and subclauses generate ATC constraints, and ESL relations are interpreted as ATC equality constraints. However, it seemed that ESL does not facilitate the specification of an objective function and does not provide analytical models. Both are necessary to establish a complete optimization problem. After providing these ingredients separately, we have demonstrated the proposed method by converting a two-level specification of a powertrain and its child components into a corresponding ATC problem. The resulting ATC problem could then be solved to find the system optimal design.

We expect that the proposed perspectives on multi-level systems design can lead to an integrated approach towards the distributed optimization of such systems. In future work, we will investigate the integration of the conversion method in the V-model design process. This involves considering the development of ESL specifications and ATC optimization problems through subsequent stages of design, while the system decomposition tree expands. Finally, we have to take into account that system design inadvertently involves qualitative concepts like creativity and inventiveness. Given that the optimization paradigm is limited to a quantitative system view, the consequences of the proposed method for qualitative design aspects have yet to be explored.

REFERENCES

Eppinger, S. D., and Browning, T. R. (2012), "Design Structure Matrix Methods and Applications", Cambridge, MA, USA: MIT Press.

Forsberg, K., and Mooz, H. (1991), "The Relationship of System Engineering to the Project Cycle", *Joint Conference of NCOSE and the American Society for Engineering Management*, pp. 57–65.

Hull, E., Jackson, K., and Dick, J. (2005), "Requirements Engineering", 2nd edn., Springer: London, Berlin Heidelberg.

Kim, H.-M., Michelena, N. F., Papalambros P. Y., and Jiang, T. (2003), "Target Cascading in Optimal System Design", *Journal of Mechanical Design*, Vol. 125 No. 3, pp. 474–480.

Michelena, N. F., Park, H., and Papalambros, P. Y. (2002), "Convergence properties of Analytical Target Cascading", *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia.

Papalambros, P. Y., and Wilde, D. (2017), "Principles of Optimal Design: Modeling and Computation", 3rd edn., Cambridge: Cambridge University Press.

Tosserams, S., Etman, L. F. P., Papalambros, P. Y., and Rooda, J. E. (2006), "An augmented Lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers", *Structural and Multidisciplinary Optimization*, Vol. 31 No. 3, pp. 176–189.

Tosserams, S., Etman, L. F. P., and Rooda, J. E. (2008), "Augmented Lagrangian coordination for distributed optimal design in MDO", *International Journal for Numerical Methods in Engineering*, Vol. 73, pp. 1885–1910.

Wilschut, T. (2018), "System specification and design structuring methods for a lock product platform", *Dissertation*, Eindhoven University of Technology.

Xu, M., Fadel, G., and Wiecek, M. M. (2017), "Improving the Performance of Augmented Lagrangian Coordination: Decomposition Variants and Dual Residuals", *Journal of Mechanical Design*, Vol. 139 No. 3, pp. 031401.

Young, R. R. (2004), "The Requirements Engineering Handbook", Boston, MA, USA: Artech House.

ACKNOWLEDGEMENTS

We thank Tim Wilschut and Albert Hofkamp from Eindhoven University of Technology for their input and support during this research.

https://doi.org/10.1017/dsi.2019.376 Published online by Cambridge University Press