International Journal of applied exercise physiology

ISSN: 2322-3537
Vol-11 Issue-02 Dec 2022

Optimizing Requirements Management: A Design-Driven Implementation Framework

Mr.Mohammad Sabeer[1]., Shivani Reddy[2]

1 Assistant Professor, Department of H&S, Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India

2, B.Tech CSE (19RG1A05C5),

Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India

Abstract

A customer's goals, ideal results, and trade-off preferences determine the objective functions in this paper, which explains the engineering design process as an optimization problem in the system design space. In this case, the consumer's demands are defined as constraints that determine the feasible range of system designs. In view of the aforementioned overhaul, this piece proposes carrying out a process for handling needs. With this reformulation, we hope to achieve the following: better communication between the design team and the customers; more customer involvement in the system definition process; better adherence to customer wishes when trade-offs and conflicts emerge; and integration of decision-making techniques and rationales with requirements discovery and management processes. Having laid out the proposed requirements management model in full, the paper moves on to examine the conditions under which this design process formulation is amenable to different methods of solving non-linear optimization issues. We suggest further case studies that use the aforementioned metrics to evaluate this method's effectiveness to more traditional requirements management solutions.

1. Introduction

Customers provide requirements to systems architects that specify complex systems. In a contract between a consumer and a service provider, the requirements spell out exactly what the system must be able to perform. When designing a system, it's important to take the client's preferences into account. designer's capacity to respond to changes in client priorities and explain design decisions is enhanced by direct feedback from end users.1 Models that faithfully describe the system while also being simpler to understand than the system as a whole greatly facilitate communication. Numerous models have been developed to explain the design process.2 the design organizations and the client's preferences, as well as the designer's own process model (if any); will determine which model is chosen for a given project. From very linear to cyclical, the models cover it all. Some models provide distinct phases of development, or stages through which a design passes, or processes that may run in parallel.

Two prevalent types are the "Vee Model" and the "Waterfall Model." This paper suggests recasting the design process as an optimization problem, with the goal being to identify the optimal combination of parameters that will result in a system that satisfies the needs of the client. The "best" in this context is an objective chosen by the client, and the "criteria" are the standards by which the system must comply.3 since the scope of this model is limited to requirements creation and design, additional process models may still be used to address topics like implementation and verification. There are a number of ways in which communication is enhanced by the optimization model of the design process. Requirements to goals hierarchies are typical decision making tools, and this model isolates the requirements elicitation process so that it may be communicated to consumers.7 The reference implementation illustrates this connection. This paradigm has the

International Journal of applied exercise physiology

ISSN: 2322-3537
Vol-11 Issue-02 Dec 2022

potential to enhance needs analysis by making use of other decision making tools, such as the decision tree, with which objective hierarchies are interconnected. These avenues of inquiry are ignored in the present work. By resolving conflicts and tying in the needs of the customer, this model of the design process also improves communication by including the reasoning behind decisions directly into those criteria.

2. Reference implementation

To frame the requirements management process as an optimization issue, the authors advise using this reference implementation as a background and providing terminology consistent with the optimization model. The components of this approach are not novel and may be found in a more familiar management plan for meeting needs. This implementation was created to convey the underlying paradigm as clearly as possible.4 for the most efficient execution of this strategy, it is recommended to use a model-based systems engineering tool that is database-oriented, can automatically monitor the attributes of requirements, and can automatically maintain relationships with a revision history.10

Specification classes, relationship rules, and conflict resolution, elicitation, and management of requirements are the three main pillars of this methodology. Specification in this context means an ordered description of the system. While all specifications may be thought of as requirements, not all requirements are necessarily written in the same formal style. Model-based systems engineering tools that facilitate the definition of new classes and relationships between them are ideal for employing this technique; these tools make it possible to easily traverse the relationships between constraints and attach validation and verification data to them. The term "BONC" may be used to describe this strategy, which is derived from the four types of specifications (Binding, Objective, Non-binding, and Conflict).

2.1. Specification classes

Identifying and conveying consumer desires, as well as deriving a system solution, are all facilitated by this approach's four specification classes. Here we distinguish between "binding constraints," "non-binding constraints," "conflicts," and "objectives." When combined, binding and non-binding constraints represent the same concept as conventional requirements management's requirements. The difference between the two types, which is not usually highlighted, is that binding constraints originate from the customer and stakeholders and should be enough to define the whole set of workable system solutions. The developer may face binding limitations from a variety of sources, such as legal or physical requirements that cannot be negotiated.

If constraints are to be nested hierarchically, non-binding constraints must be derived from either binding constraints or a higher-level non-binding constraint. The systems architect will need to do some design work if the limitations are not mandatory. When added to the customer-generated limitations, these additional constraints progressively narrow the solution space until only one viable option remains. However, non-binding constraints are handled differently from binding constraints, and the customer's agreement of them is not always required. This approach is distinguished by its non-binding limitations. The systems design group may decide how they should be handled. The specification model may also include documentation of the design process by making use of the non-binding restrictions. Using binding requirements to symbolize all of the conventional criteria at each level helps establish a clear line between design effort and what is typically viewed as requirements. Keep in mind that this definition of "binding" and "nonbinding" constraints is distinct from how other optimization textbooks define these words. Here, the solution is said to be "binding" if it perfectly satisfies the constraint and “nonbinding if it goes beyond the constraint's requirements.11

Whenever a validation check identifies a discrepancy between two requirements, a conflict is generated to detail the discrepancy and indicate the resolved constraint. The first step in resolving a dispute is to identify the two constraints at issue, which may or may not be the identical constraints that first triggered the disagreement. The resolution process proceeds in accordance with the established order of priority, and in the end, feedback from the affected customers should be included into the solution. In section 2.3, we go into further depth about how to handle conflicts between constraints.

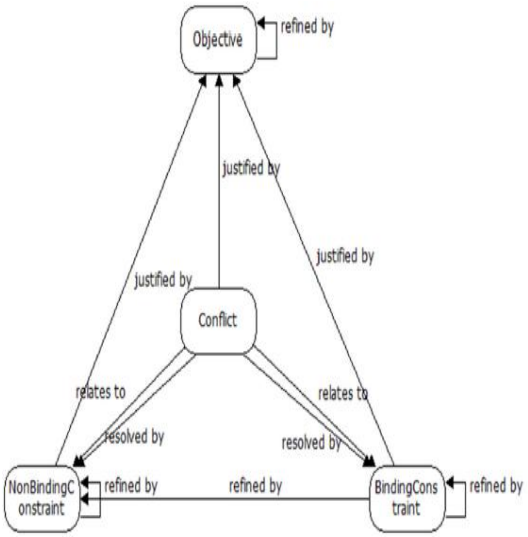
.2. Relationships

Each class in the specification may be related to another. Refines/refined by and justifies/justified by are the two most common types of class connections. Figure 1 is a graphic depicting these important connections. Both binding and non-binding restrictions may have a "refines" connection with one another. Pairings of goals, or between a binding and non-binding restriction where the latter refines the former. The objects involved in the relationships are arranged in a hierarchy, with the item doing the refining being

International Journal of applied exercise physiology

ISSN: 2322-3537
Vol-11 Issue-02 Dec 2022

lower in the hierarchy than the object being refined. This is consistent with the standard practice of categorizing needs according to their granularity. When goals are connected in this manner, a hierarchy of objectives is created, much like the decision-making hierarchies we're accustomed to seeing.



Class connections are shown in Figure 1.

Objectives and other classes have a "justifies/justified by" connection. A "justifies/justified by" link between an aim and a constraint implies that the objective has contributed to the justification for the limitation. Even if there is no verifiable non-binding constraint relating to cost, in the case where the objective function is to minimize cost, the objective function would justify the addition of a non-binding constraint by the system architect that would restrict the feasible design space to certain architectures on the basis that those architectures are lower cost. A direct line of sight between the target and the customer's wants and the system architect's design decisions is established.

2.3. Conflicts

A conflict between two constraints may be uncovered during validation.5 when requirements are at odds with one another; it means that there is no workable solution inside the design area that has been defined by the constraints. While a discussion of tensions is not required for the theoretical application of the optimization model, it is vital when real-world circumstances that may develop. Although conflicts are unusual, they should be resolved since they indicate gaps in our knowledge of the system. The client should direct the resolution of any disputes between two binding restrictions. The following principles are provided as guides, and a conflict may be handled by the client or by modifying non-binding restrictions as the system architect sees appropriate. When a binding requirement conflicts with a non-binding constraint, the latter should take precedence since it was mandated by the customer. Constraints at a higher level should often be given more weight than those at a lower level. If a binding constraint at a lower level conflicts with a non-binding constraint at a higher level, or if two non-binding constraints at the same level conflict, the conflict should be traced to the higher level constraints until one of the other cases is encountered, as the constraints may be refining higher level constraints that only slightly conflict. Although unresolved constraints that were engaged in a dispute should be retained for traceability, they should not be published with the body of current constraints.

To resolve the conflict, either the conflicting constraint must be deleted or another constraint must be refined. For audit ability purposes, conflicts should be recorded at the same level as the conflicting restrictions.

2.4. Elicitation process

To make the most of this approach, it's important to gather trade off relationships with the customer's broad objectives and narrow preferences during requirement elicitation. This technique depends on this kind of elicitation, which is already a part of many requirements management procedures, to generate a suitable objective function for the parameters. Detailed expressions of aspirations and motivations that cannot be reduced to a single constraint statement are also given more weight.

2.5. Management process

The amount of specificity of the restrictions informs the hierarchical organization of the components. All projects should be able to compile with at least level 1 restriction. Most projects just need a single document at Level 1 that details the major client goals and restrictions. Formatting suggestions include: explaining the goal utility first, then customer priorities and compromises. The next step is to classify the binding constraints as either functional or process. Last but not least, the requirements for client acceptability testing must be specified. There may be a need for many papers if there are many different types of constraints or if there are many different stakeholders who will submit binding constraints. If this is the case, then everyone should know exactly where to find the list of goals.

International Journal of applied exercise physiology

ISSN: 2322-3537
Vol-11 Issue-02 Dec 2022

3. Implications of the optimization model of the design process

The optimization model of the design process will be defined as

$$\min f(x) \ s.t. \ B(x) \tag{1}$$

$$x \in X$$

The objective function is denoted by f(x), while the restrictions are denoted by B(x). All possible choices for a design may be found in the set X. Many of the qualities that make solving the issues easier in optimization theory are absent from the binding constraints, viable designs, and even the target. In most cases, the functions are not linear, possible inconsistency; not defined over a convex set; neither convex nor concave. This often renders the theorems associated with extended optimization, such as the Karush-Kuhn-Tucker (KKT) optimality conditions, inaccessible. Several approaches exist that attempt to deal with these concerns while still allowing for inferences to be made about the system design process.9 I will elaborate on a select handful of them below.

The first approach is to break the system down into more manageable pieces that yet satisfy the requirements. The ideal reduced system would satisfy the condition for a convex programming problem, allowing for easier solution with less work.8 In some cases, this is straightforward to achieve; for instance, a software system may have a clear goal and a set of functional requirements that can be mapped onto convex functions, but also include requirements specifying documentation or quality assurance that are not tied to the design variables specified in the functional requirements. In reality, this approach may be quite subjective when deciding which compact systems are most suited. In addition, only "well behaved" systems will naturally reduce to a convex programming problem without further iteration on the constraints. The process of breaking down a complex issue into manageable pieces is standard practice in design; hence it does not advance the state of the art in system design. Altering requirements elicitation such that only convex-function-isomorphic constraints are generated is another option. Requirements analysis utilizing KKT conditions or other ways of addressing convex programming issues might alter the conventional elicitation procedure and provide a set of system constraints. If the design process were represented as a convex programming issue, viable solutions may be arrived at more quickly since the theorems pertaining to convex programming give sets of both necessary and sufficient criteria for discovering optimum solutions. However, this approach may be quite challenging, and there is no assurance that the system the client wishes to develop can be formulated using this technique. In addition, this method of obtaining restrictions may seem exotic to system architects and completely alien to end users. This undermines the design process optimization model's communication benefits. System architects would benefit greatly from the discovery of a tangible, intelligible approach for extracting actual system constraints that can be transferred to convex functions.

3.1. Duality

The capacity to conceive and solve the dual issue is perhaps the most significant consequence of the suggested model of the design process. An issue's dual is a similar problem expressed in a different set of variables. Even though the original issue cannot be properly examined, its dual may be due to the dual's unique qualities. What is meant by the term "dual function"? 6

$$\theta(w) = \inf\{f(x) + w^T B(x) : x \in X\} \tag{2}$$

$$w^T = (u, v)^T$$

Lagrange multipliers u and v and the minimum function info are defined as follows. Therefore, we have a two-pronged issue:

$$\sup \theta(w) \ s.t. \ u \geq 0 \tag{3}$$

The dual function has numerous useful qualities even if the original issue is not specified over convex sets or if the constraints are not convex, including being concave, possessing sub gradients, and allowing for the determination of directions of steepest climb. The solution to the dual issue (D) is simplified as a result of all of the above. An issue of maximizing. It is also feasible that, even when using standard design methods, a fresh perspective gained from the formulation of the dual issue will allow for a deeper understanding of the system or the identification of optimal solutions.

International Journal of applied exercise physiology

ISSN: 2322-3537
Vol-11 Issue-02 Dec 2022

4. Conclusions and further work

In this study, I suggested modeling the design process as an optimization problem with a customer-provided goal and design constraints. I also provided a case study of a requirements management system installation that shows how this idea may be used to effectively convey system needs, or limits, must be met. The illustrative requirements management system not only enables the customer-provided information to be defined, but also the design choices and their justifications. One might apply the system architect-defined limitations during system design to reduce the viable area to a single design. In this publication, we attempted to provide the groundwork for future research. From a mathematical perspective, seeing the design process as an optimization issue is neither strange nor difficult. It has to be seen, however, if this reduces the amount of time needed to find a solution or results in a better outcome. It's not easy to design a new system, especially because not all optimization methods are computationally efficient. To assess this framework's efficacy and potential applications, a real example of its use must be constructed and contrasted head-on with a conventional formulation. In order to meet the needs of a nanosatellite development project at a university, a case study of this kind is now in the planning stages.

References

1. Carlshamre P, Regnell B. Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. REP2000.

2. CMMI for Development, Version 1.2. Pittsburgh: Carnegie Mellon Software Engineering Institute, 2006.

*3. Mylopoulos J, Chung L, Nixon B. Representing and Using Nonfunctional Requirements: A Process-Oreiented Approach. IEEE Transactionson Software Engineering 1992; **18**:6 .*

4. Hoffman M, Kühn N, Weber M, Bittner M. Requirements for Requirements Management Tools. IEEE International RequirementsEngineering Conference 2004.

5. Shehata M, Eberlein A, Fapojuwo A. IRIS-TS: Detecting Interactions Between Requirements in DOORS.

6. Bazaraa M, Sherali H, Shetty C. Nonlinear Programming: Theory and Algorithms. 3rd Ed. New Jersey:John Wiley & Sons; 2006.

7. Clemen R, Reilly T. Making Hard Decisions. South-Western; 2001.

8. Boyd S, Vandenberghe L. Convex Optimization. Cambridge, UK:Cambridge University Press; 2004.

9. Le Thi A, Pham T. The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems. Annals of Operations Research 2005; 133:23-46.

10. Long D, Scott Z. A Primer for Model Based Systems Engineering. 2nd Ed. Vitech Corp; 2012.

11. Hillier F, Lieberman G. Introduction to Operations Research. 9th Ed. New York:McGraw-Hill; 2010.