EMERGING CAD PROCESSES: AN ANALYSIS AND REVIEW OF COMPUTER AIDED REQUIREMENT MANAGEMENT

IPEK OZKAYA AND OMER AKIN
Carnegie Mellon University, School of Architecture

VAN WOODS
US Army Corps of Engineers, Construction Engineering Research Laboratory (CERL)

Abstract. Many large institutional organizations and corporations actively manage their facilities and future design needs. Facility related criteria express prescriptive and descriptive requirements that are relevant to their mission-specific, institutional objectives. Creation, maintenance, distribution, and the effective use of this information are currently typically managed in an *ad hoc* manner without domain-specific computational support. In this paper, we review current approaches of managing facility requirements that provide computational support. We provide suggestions for the future role of requirement management in design and planning.

1. Introduction

Information management has been a primary focus of building design computing for over three decades. One of the key information management tasks, especially during the problem formulation phase of design, is managing the poorly or partially presented design requirements, and extracting useful design information from them. Problem formulation involves utilizing different types of information that express a desired state or a proposed solution at different stages of the design process. Requirements represent these different types of information and they can come in the form of needs and wants of the clients and users, constraints and standards of the selected technologies, information about functional and operational aspects of the desired design solution, implications of the design decisions, existing conditions, or quality determinants. The desire to achieve results within a predictable level of quality, cost, and schedule motivates organizations to spend considerable resources in the process of managing requirements. As a result, interest in computer support for requirement management tasks has been increasing.

The basic process of managing requirements refers to 1) the elicitation and expression of desired characteristics, 2) the usage, maintenance, and distribution of this information, and 3) the measurement and tracking of the degree of design compliance and other performance analyses. Requirement specification starts early in the design process and provides the context and design rationale for most subsequent design decisions. However, current approaches do not address the bottlenecks observed in inefficient use of the requirement information, unsupported change management, consistency checking and design compliance verification processes in building design. As a consequence, the use of early design information is left to be a time consuming, error prone, ad-hoc, and manual process. Our interest is in exploring how to electronically manage requirements in order to increase quality by addressing these bottlenecks, while facilitating better information management throughout the overall building life cycle.

The use of building product data models and computer supported design processes are no longer imaginary scenarios, but are necessary for design productivity and performance. Moreover, increasing computer literacy in general purpose applications, as well as in assistant tools that address performance analysis, collaborative design, and information management are promising developments for design life cycle information management. Initiatives like Leadership in Energy and Environmental Design (LEED) Green Building Rating System are also increasing the awareness of the professionals in managing the information that is created during the design delivery process. However, the concept of providing computational support for requirement management is still rather new in the building industry and has not reached a level of maturity where utilization of early design information has become more accurate, robust and persistent throughout the design life cycle.

This paper will describe computer aided requirement management (CARM) by focusing on selected studies and tools from government, industry and academic research. Section 2 will summarize the motivations behind various stakeholders’ interest in digital requirement information management. Section 3 will present an array of exemplary tools and approaches addressing building data modeling, collaborative information sharing, usability, and design and life cycle integration aspects of requirement management problems. Following this discussion, Section 4 will discuss building product data and process modeling for CARM. Section 5 will provide examples of applications of requirement management in other design-related domains, specifically from software engineering. And finally, Sections 6 and 7 will give a summary of our observations and our concluding thoughts for a roadmap, respectively.

2. Needs and Wants

Reasoning with requirements is inherently a user-mediated process. An interesting aspect with most designs is that any specification of requirements in an open domain is partial at best, given the presence of requirements that are implicit (not yet expressed) and tacit (difficult or impossible to model) (Hutchins 1995, Polanyi 1962). This is commonly referred to as the emergent nature of design, which leads to the inherent difficulty (or some would say impossibility) of automatically generating design solutions from requirement specifications, no matter how detailed they may be. This holds true even when dealing with routine cases, such as computing requirements for recurring building types (Erhan 2003).

Currently, many requirement specifications are already stored as digital artifacts, albeit in loosely structured formats. These include items such as facility design guides and conceptual design analysis reports in textual formats in word processing documents. The use of digital spreadsheets for architectural programming spatial analysis is also common. However, these only provide digital storage media and are not accompanied by any advanced computational support specific to the process commonly referred to as briefing or architectural programming.

While the basic premise is to formalize what is expected and what is delivered, there are opportunities to meet more advanced user needs in requirement management. The user in this case includes facility owners and designers, as well as the many other architecture, engineering and construction (AEC) industry professionals participating in the design and delivery of buildings. During the requirement elicitation processes, issues such as collaborative data management for group elicitation and support for broader data gathering methods, such as surveys, are of relevance. The common participation of numerous stakeholders makes distribution of requirement information, change notification, and version control processes critical. Issues surrounding data maintenance and updating are also inevitable, given that requirement information for buildings are rarely static over time. In addition to the requirement elicitation and information distribution issues, the actual usage of the created information for decision-making brings to the fore unique needs for CARM. Decision-making processes utilize requirement information relevant to both designers and managers such as requirement dependencies, preliminary cost and schedule, and other possibly analyzable trends, such as coverage rate and creativity.

Given the collaborative nature of the requirement management process, requirement tracking in terms of compliance validation, responsible agent identification for a particular requirement item, priority, approval, and completion status all become crucial pieces of information to control. Other desirable functionalities include providing efficiencies through adaptable reuse, possibly in the form of solution templates and reusability strategies or patterns. And finally, an item that is commonly overlooked is how such approaches fit into current and legacy information documentation. Computational support for reverse engineering and data mapping to loosely structured documents are exemplary functionalities in this area.

Currently, treatment of requirement management via computer aided design tool support is dispersed. While there are tools that cover the common information management aspects of CARM, such as navigation, accessibility, exchange and maintenance of data, these do not impact the use of requirement information as a design aid. In section 3 we will present example studies from industry, academic and government research platforms to demonstrate the focus areas in CARM.

3. Approaches to CARM

Research, both in academia and in government agencies, has been addressing the CARM problem for more than a decade now. Recently, these developments have resulted in a significant increase in the number of prototype tools that address requirement management both in research and in commercial initiatives. Exploration of appropriate computational techniques to aid designers in CARM, generally speaking, calls for an emphasis on information processing and design integration. More specifically, the literature on design requirement management in architecture reveals multiple tracks: architectural programming, performance modeling, design integration, and requirement management in other design disciplines whose problems are similar to architectural problems in complexity and structure, such as software engineering. These suggest a variety of approaches for managing requirement information in AEC.

Requirement discovery and representation problems based on designers' need to translate customer and product needs into technical constructs are generally referred to as requirement elicitation. Techniques used in this area, such as QFD for client requirement processing (Kamara et al. 2000), value added functional analysis (Elbibany et al. 1997), and axiomatic design (Suh 2001), are commendable approaches to structuring the process, but their manual approach and lack of overall integration with building models make them expensive and time consuming to use. Moreover, such techniques address requirement management problem during early phases of design which creates potential bottlenecks in the use, maintenance, distribution, design compliance verification, and tracking of the requirement information in subsequent design phases. In this paper, we review examples which address one or more of the following: managing design life cycle through requirement information, integrating requirements to building product data modeling, and extracting potential generalized approaches for requirement management support.

3.1 GOVERNMENT INITIATIVES

The United States government is arguably the largest owner of facilities worldwide, as well as the owner of the largest amount of documented requirements and criteria. The amount of requirements are due to 1) the shear magnitude of quantity of facilities that are necessary, 2) the vendor neutral nature in which the government operates which requires performance specification rather than specific manufacturer and product preferences, 3) the strict performance requirements necessary to uphold the unique security and technical functions, 4) the presence of commonly recurring facility types which leads to an accumulation of corporate knowledge over time, and 5) the need to collaborate between highly dispersed groups as well as between public and private organizations.

Government agencies have been placing a special emphasis on the importance of managing requirements for the design life cycle, with a focus on the quality of the building end product. The Whole Building Design Guide (WBDG) is an example to these efforts (WBDG 2003). WBDG is a collaborative effort among federal agencies, private sector companies, non-profit organizations, and educational institutions lead by the National Institute of Building Science (NIBS). The goal of the 'Whole Building' design approach is to create successful high-performance buildings by focusing on a holistic process. To achieve this goal, an integrated design approach to the project during the planning and programming phases is promoted. The premise of WBDG is that buildings must be competently planned, functionally adequate, appropriate in form, cost-effective, constructible, adaptable, durable, and contextual. Its aim is to emphasize the interdependence among building systems and provide resources for professionals that share these goals, many times organized by facility type.

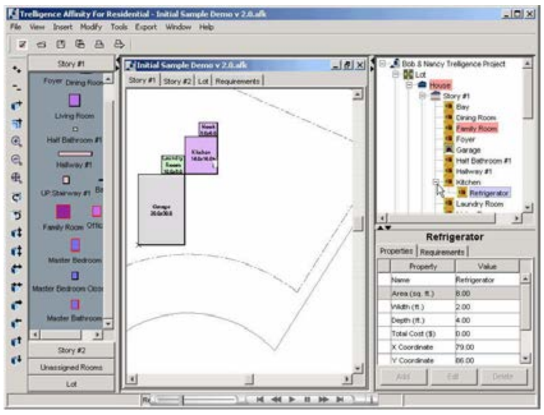
Another significant effort that government agencies have pursued is to move towards an electronic publishing format for the requirements they use for their facilities. The Army Criteria Tracking System (ACTS 2003), RPLANS (RPLANS 2003), and the Army Corps of Engineers Engineering Regulations repository websites (EngRegs 2003) are a few of these online resource examples. The mission of these resources is to provide policy guidance and program management on all matters relating to the overall management of Army installations worldwide. The software applications used and created are directed towards easing collaborative information creation, sharing and maintenance. They also try to alleviate bottlenecks caused by manual techniques, which are both prone to errors and do not support change management well.

These government initiatives have been also pursuing software support to maintain the information generated and its integrity. SpecIntact (Specifications-Kept-Intact) is an automated system for preparing standardized facility construction specifications (UFGS 2003). It is developed for worldwide use by NASA, the U.S. Naval Facilities Engineering Command (NAVFAC), and the U.S. Army Corps of Engineers (USACE). The functionalities of the software mainly focus on creating an automated collaborative environment where project stakeholders can develop, share, and reuse building information. The software is intended to manage the Unified Facilities Guide Specifications (UFGS) under Construction Criteria Base (CCB). CCB is an extensive electronic library of construction guide specifications, manuals, standards and many other essential criteria documents by the National Institute of Building Sciences (NIBS). The tool can be viewed as an advanced, but specialized text editor, allowing editing and organizing requirements using standard generalized markup language (SGML), an international standard that allows defining and tagging of information within documents. The tool is used in assisting information exchange management.

3.2. COMMERCIAL TOOLS

There are only a few commercial examples illustrating any type of integrated design environment that supports requirement elicitation while also providing features for the design process, allowing a designer to review and compare the current state of a design proposal to the specified user requirements. Trelligence is an exception as it is a good example of tracking functionality dealing with requirement elicitation, definition and design (Figure 1). It provides visual feedback, context specific indication of relevant performance objectives, and graphical aids for space layout (Trelligence 2003).

Commercial tools integrated with database support are also available. For example, SpecLink and Perspective by Building Systems Design, Inc. offer comprehensive search mechanisms and change management for keeping track of requirement information. User interfaces developed specifically for managing requirements increase the usability of these applications in CARM (BSD 2003). And finally, there are also tools that integrate requirement management with a building product data model back-end. Solibri Model Checker by Solibri Inc. is a typical example. The application acts as the equivalent in concept to a spell checker program for building models based on Industry Foundation Classes (IFC) by allowing automated checking of design errors that can be specified by the end user (Solibri 2003).


Trelligence Affinity For Residential - Initial Sample Demo v 2.0.afk
File View Insert Modify Tools Export Window Help
Initial Sample Demo v 2.0.afk
Story #1 | Story #2 | Lot | Requirements
Bob & Nancy Trelligence Project
Lot
House
Story #1
Bay
Dining Room
Family Room
Foyer
Garage
Half Bathroom #1
Hallway #1
Kitchen
Refrigerator
Laundry Room
Refrigerator
Properties | Requirements
Add
Edit
Delete

Property	Value
Name	Refrigerator
Area (sq. ft.)	8.00
Width (ft.)	2.00
Depth (ft.)	4.00
Total Cost ($)	0.00
X Coordinate	79.00
Y Coordinate	86.00

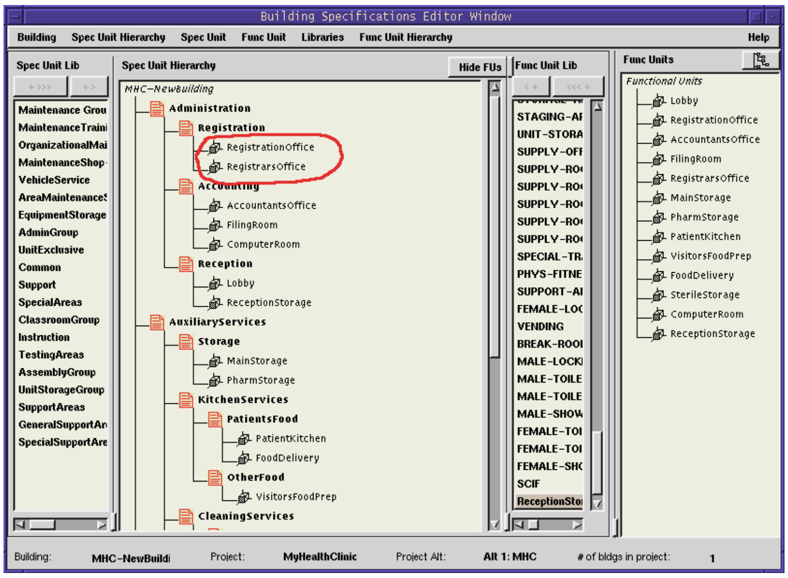
Figure 1: *Screen shot from Trelligence showing the integrated design and requirement information medium.*

3.3 RESEARCH-BASED INITIATIVES

There are multiple research-based initiatives that are of relevance. We will review systems that we consider seminal in this area. They allow the possible use of requirement data in design life cycle, and also provide strategies to manipulate this data. They offer possibilities to extend information management into design beyond requirement documents management. We have selected SEED-Pro, Rabbit, Facility Composer, Design Intent Tool and Metracker systems. SEED-Pro (Akin et al. 1995) and Rabbit (Erhan 2003) have been developed at the School of Architecture at Carnegie Mellon University as a result of the Software Environment for Early Phases of Building Design (SEED) research (Flemming et al. 1995). The Facility Composer suite of tools has been developed by the Construction Engineering Research Laboratory (CERL 2003); the Design Intent Tool (DIT) by Lawrence Berkeley Labs (DIT 2003); and Metracker by The Public Interest Energy Research (PIER) (Metracker 2003). In this section, we will briefly present these systems focusing on their features that we believe are influential in providing a vocabulary for more advanced support for CARM.

3.3.1. SEED-Pro

SEED is a computer supported design environment featuring an open-ended modular architecture, where each module focuses on a design activity that takes place in the early design stages. SEED provides support for a range of tasks starting with architectural programming and spanning to 3D configuration management, for the entire design life cycle process. Each module consists of five main components: input, specification, generation, evaluation and output. These are supported by a database to store and retrieve information, as well as a user interface to facilitate the interaction with designer.



Building: MHC-NewBuildi Project: MyHealthClinic Project Alt: Alt 1: MHC # of bldgs in project: 1

Figure 2: *Main requirement modeling window of SEED-Pro*

To support design generation, a well-defined set of explicit requirements is needed. SEED-Pro, the building requirement specification module of SEED, has been designed with the intention to assist the modeling and generation of design requirements in a form that is usable by other modules of SEED (Akin et al. 1995). It fulfills the following objectives:

- Storing and handling multiple aspects of requirement information including site characteristics, codes, client preferences, and design performance criteria.
- Using criteria established during the requirement specification phase as a basis for design generation.
- Integrating building requirement specification and architectural design as a seamless process.
- Making requirement specification decisions and improving the computability of requirement specification information by allowing non-numerical types of reasoning in the database.

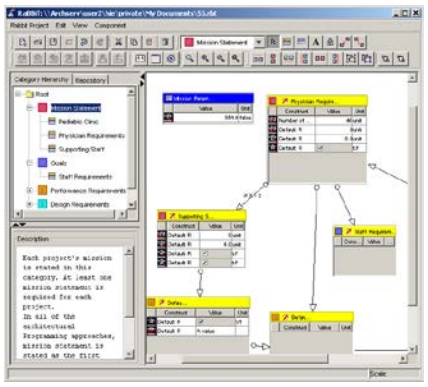
- Providing a flexible interaction that does not tie the user to a specific requirement specification model.
- Using past requirement specifications in future projects.

The main interaction window in SEED-Pro supports the development of a hierarchical decomposition of the requirement for design (left two columns in Figure 2). In this hierarchy, all components of the buildings functional, programmatic, budgetary, site-related and organizational aspects can be modeled. Each of these components is further specified through a predefined attribute-value model. Furthermore, SEED-Pro allows the user to create functional constraint hierarchies (right two columns in Figure 2), which correspond to these requirement specifications. In this way the designer can specify the precise area, adjacency, and other performance constraints to be met by succeeding modules of SEED, whether these are for space planning (SEED-Layout, Flemming and Chien 1995), system configuration (SEED-Config, Woodbury and Chang 1995) or standards conformance (Garrett et al. 1995).

Through the sharing of domain object classes, SEED-Pro provides seamless interaction with all of the other modules of SEED and shared data across these modules. SEED-Pro maintains a robust record of design requirements, criteria, and constraints to be used persistently during design. Seed-Pro, in its second version called SP-II, handles all of these functionalities in addition to providing flexibility to dynamically specify the parameters of the modeling environment, including, variables, data types, relations and dependencies between variables. This overcomes the inflexibility caused by the pre-defined attribute-value sets present in the earlier version of SEED-Pro.

3.3.2 Rabbit

The Rabbit environment (Erhan 2003) primarily focuses on eliciting requirements for recurring building types interactively (Figure 3). Erhan’s study aims to help designers, facility owners, and planners interactively define (enter program components with constructs as design program parameters), generate, and modify program requirement information by providing a visual navigation system to the user. Similar to Seed-Pro (Akin et al. 1995) and SP-II (Donia 1998), this study focuses on the computer’s ability to generate architectural programs that can be accepted as input by other applications, like generative design and decision support tools. Of particular interest with this approach are the underlying graph transformation concepts, the flexibility in ability to express the design requirement assumptions, and the visual display tools that illustrate dependencies, as well as the ability to provide immediate feedback on allowable interactive transformations of data.



Each project's mission is stated in this category. At least one mission statement is required for each project.
In all of the architectural Programming approaches, mission statement is stated as the first

Figure 3*: Rabbit's programming environment*

3.3.3. Facility Composer

The Facility Composer suite is a set of tools developed by the Army Corps of Engineer's Construction Engineering Research Laboratory (CERL) that supports the notion of criteria-based facility modeling (CERL 2003). The suite consists of three primary tools: Requirements Composer, Planning Composer, and Layout Composer. These correspond to capabilities to: assist the creation and modification of reusable, building-type specific, default criteria stored in criteria libraries; support for architectural programming and other project-specific criteria value specification during interactive design charrettes or at the designer's desktop; and support for the creative and analytical aspects of architectural conceptual design in a 3D environment, respectively.

Features of Facility Composer include an interface that supports high-level domain specific actions in an interactive graphical environment (such as a project tree that is tied to the project's graphical representation as seen in Figure 4), the notion of a project hierarchy and the ability to specify criteria at different levels of detail (with inheritance and specialization) organized by user-definable design disciplines, support for the generation of alternatives, and multiple geometric representations for any given solution ("Above/Current/Below" which corresponds to architectural plan delineation conventions, "Bubble Diagram", and "Color by Function").

It is envisioned that after having built numerous buildings of a particular type, say barracks (or an apartment complex as a similar example in the private sector), over time much of the relevant criteria will have emerged and have been expressed. These types of collections of criteria can be stored in a 'criteria library' of sorts, which would allow for rapid reuse by reducing the amount of up front repeated data entry. Use of criteria libraries are also expected to result in improved customer requirement elicitation, less errors and omissions resulting in fewer problems due to inconsistent expectations, and better preliminary budget estimation in cost, capability, and time.

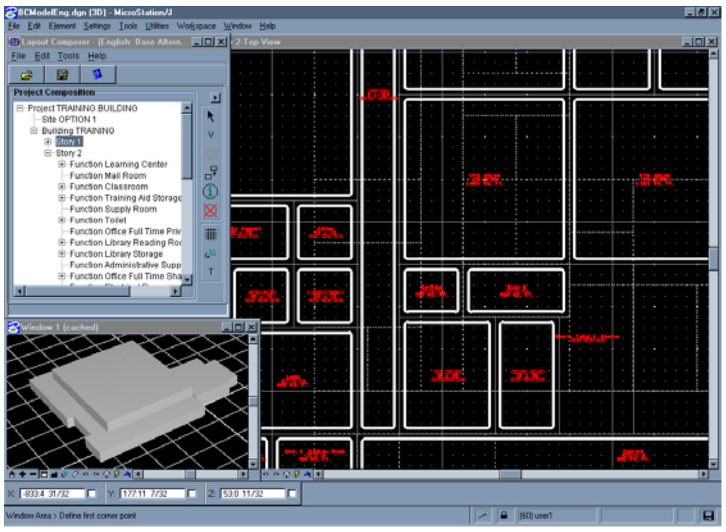

BCModelEng.dgn (3D) - MicroStation/J
File Edit Element Settings Tools Utilities Workspace Window Help
Layout Composer - [English: Base Altern...
File Edit Tools Help
Project Composition
Project TRAINING BUILDING
Site OPTION 1
Building TRAINING
Story 1
Story 2
Function Learning Center
Function Mail Room
Function Classroom
Function Training Aid Storage
Function Supply Room
Function Toilet
Function Office Full Time Priv
Function Library Reading Roo
Function Library Storage
Function Administrative Supp
Function Office Full Time Sha
2-Top View
Window 1 (cached)
Window Area > Define first corner point

Figure 4: *Integrated environment of layout composer with requirements compose as implemented in Microstation/J*

The application allows architectural programming processes to support top-down, bottom-up, or middle-out approaches within project hierarchies that consist of *Project*, *Site*, *Building*, *Storey*, *Function* (space types), and *Space* instances. For example, in the project composition hierarchy, one could program the requirements for 1000 sf. of offices in a *Building*, for which during design, *Storey* A ends up with 400 sf. and *Storey* B ends up with 700 sf. Note the allowance for discrepancy from the initial programmed amount. The user can visually discern that they have committed too much area for which they could reduce the higher-level goal or the lower level assignment. The reverse (starting with specification at the lower level and working up), and hybrid approaches are possible as well.

The Layout Composer application integrates the required functionalities of CARM in a CAD environment. Once the architectural programming has been conducted, the design architect can take the expression of desired functions and their area allowances and adjacencies, as well as other project criteria, to begin establishing one or more preliminary spatial configurations graphically within the CAD environment. The application provides domain-specific functionality such as the ability to create stories and placing spaces on them, while leaving CAD-specific functionality such as snapping and other geometric editing capabilities, as well as 3D volume rendering with transparency and animation to the CAD environment.

One of the goals is to make the user interaction dramatically easier than current modeling approaches by providing specific tools and components (for example the hierarchical project tree and criteria table), as well as by having the system automatically keep track of the evolving design in order to illustrate variances with the desired criteria. The system is designed such that operations and properties at higher levels in the hierarchy automatically propagate to lower levels, and lower levels can also provide specialized values. This enables features like being able to delete or move all of the spaces on a storey by changing the story instance, or by changing a floor-to-floor height between stories, which automatically adjusts the height of the spaces on each of the stories, including a space that might span across multiple stories. This is in contrast to current commercial application approaches that require file-based approaches or are not capable of these features.

Another feature of the suite is a concept called wizards. These allow an organization to define chunks of design logic in a computable format along with their associated criteria. A wizard allows new design processes and calculations to be included without requiring a recompiling of the main application. In addition to application extensibility, it also enables more effective information management and reuse by decoupling design logic from the main application, and by providing greater cohesion between the design logic and the criteria it uses. Take, for example, a Parking Lot wizard. Chances are the user could use such a wizard to aid in the creation of suitable parking lots for any number of building types. There may be certain building types that would have to have a specialized version of the wizard that might, for example, account for more security.

Development of the system was heavily influenced by prior collaboration with the SEED project and from prior work at CERL on the Modular Design System (MDS 2004). In particular, there was the observation of the benefits of clearly delineating between user need specification, functional requirement description (spatial types and their associated criteria), spatial layout exploration, and spatial configuration representations (Flemming et al. 1995). This approach was shown to be robust as it recognized the intrinsic nature of many designs in open or “wicked” problem domains (Rittel 1973) in allowing for each upstream solution to serve as a specification of a desired state for which one or more downstream solutions could be developed. Other desirable characteristics were the aspects of extensibility, and also the importance of providing separate and specific interfaces for different phases in the process within an integrated environment. SEED was one of the first implementations in which these primary tenets were observed in an integrated environment. This was a departure from prior work at CERL where requirements were implicitly encapsulated in graphical representations of pre-designed geometric configurations.

3.3.4.Design Intent Tool

Design Intent Tool (DIT) helps building owners, architects, and engineers develop a “design intent document” (DIT 2003). This document is aimed to facilitate record keeping and ensure that the owner's and designer's up front vision and goals are achieved and periodically verified through performance measurement. The focus of the “document” as it is scoped in DIT is on energy-efficiency, while intended for any aspect of design. The documents evolve as the project moves through the milestones of programming, design, and construction, into building occupancy and potential future renovation and retrofits. Design intent documentation is crucial to the post-construction commissioning process (verifying the proper installation, operation, and performance of energy-efficiency features), and is the essence of communication and contractual obligation between the building owner, architects, engineers, builders, and commissioning agents. This intent also keeps track of the requirement information as it goes through various stages of design. The document provides a reference medium for design life cycle management and collaboration.

3.3.5 Metracker

Metracker is a prototype computer tool designed to demonstrate the specification, tracking, and visualization of building performance objectives and their associated metrics across the complete life cycle of a building (Metracker 2003). The underlying concept is that, in order to better assure the intended performance of a building, it is necessary to establish a baseline for expected performance and periodically compare actual performance to this baseline. This process requires a standardized, yet flexible, format for archiving performance data, and sharing these data between various software tools and their users throughout the building life cycle. Ideally, these performance data are archived with, and related to, other information about the building. To these ends, Metracker is based on the IFC data standard.

4. Building product data modeling

Generation of requirements such that other applications can extract design information from the modeled information has been a natural outcome of the studies on shared information models and building product data modeling. One of the core research questions in studies involving AEC and information technology related tasks is semantic data modeling stemming from the lack of effective data exchange mechanisms (Bjork 1991, Eastman 1999, Galle 1995, Ekholms & Fridqvist 1999, and others). The core IFC model focuses on the building as a product. While this approach encapsulates some of requirement information, it is not complete. Recently, a number of studies aiming to augment the IFC model with requirement information have been increasing. In this section we will present sample CARM approaches in building product data modeling.

4.1. REQUIREMENT DATA MODELING

There are several ongoing projects that aim to study building product data models and their capabilities in supporting early design. The aim of such studies is the capturing of requirements and information from multiple stakeholders. The International Alliance for Interoperability (IAI) Early Design Project hosted by CERL and the Architectural Ecology organization (IAI 2003), and the Product model extension for requirement management interfaces (Premiss) project at Stanford University (Kviniemi and Fisher 2003) are examples of such efforts.

The IAI Early Design project at CERL aims to provide IFC based modeling of data to support early design processes. The intent of the project is to make early design information available in an interoperable product data model form, so that the information can be used at other stages of the design life cycle. In this project, performance based requirement specification is the priority for early design information modeling. Specifically, the project aims to identify and define the information objects that are used by owners; encourage users to define what is required during the early design phase; develop IFC support for refining the building concept, spatial functions, and spatial programming; and design criteria specification and early geometric representations, for example, for blocking and stacking and bubble diagramming.

The Premiss project suggests extending and/or refining the current IFC model with requirement specific data classes. This effort points out the inadequacies of building data models in including requirement data within the design life cycles (Kiviniemi and Fisher 2003). Kiviniemi and Fisher identify: the difficulty of keeping track of requirements during projects, especially in cases where projects are long and people change; difficulty of tracking the implications of changes in requirements and how design components relate to requirements; lack of tools to manage requirements and the need to manage requirements on class and instance levels.

4.2. BUILDING COMMISSIONING

The building commissioning (BC) project at Carnegie Mellon investigates how building commissioning of HVAC systems are conducted and represented in the building product model (Akin et al. 2003). The effort identifies process activities such as the generation of the design intent document during the programming phase, while comparing the ASHRAE guidelines and actual practices. This includes requirement data such as construction cost estimates, testing, adjusting and balancing requirements, preparation of design concepts based on design intent, construction documents based on design concepts, and verification processes based on the documents produced during these phases. The processes described in this study rely on the flow of decisions, procedures and output documents of the commissioning process. BC undertakes validation and verification of design intent. Moreover, by emphasizing the importance of approaches such as Continuous Commissioning© (an ongoing process for commissioning) and suggesting a persistent commissioning model that combines the process with the building life cycle, the project places requirement information in front-and-center of the building life cycle processes (Akin et al. 2004).

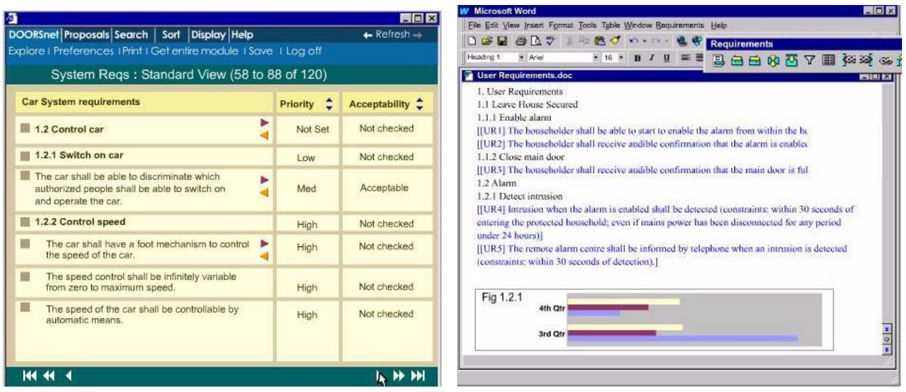
5. Requirement computation in other design disciplines

Design disciplines whose problem domains are similar to architectural problems in complexity and structure, such as software engineering, mechanical engineering, and civil engineering have similarities in requirement computation with architectural design. We will focus on software requirement engineering in this section.

Software engineering is a relatively new field of “design.” The immediate parallels between software engineering and architecture are less obvious compared to mechanical or civil engineering. While the products of architecture and software engineering differ significantly, the nature of the design process shows quite a number of parallels. In software engineering, architecture serves as a source of countless metaphors in defining the software development process (e.g. Leffingwell and Widrig 2000, Gamma et al. 1995). Architects produce blueprints as software engineers produce system diagrams, architects use design patterns as established good solutions as software engineers use them in the same way. Architects rely heavily on requirement elicitation to define their problems as software engineers do. Like software solutions, architectural solutions are not unique. Most profoundly, both architecture and software engineering are disciplines that deal with ill-defined problems.

In software engineering, requirement engineering is a distinguished sub-discipline, which addresses the problem of managing initial design requirements throughout the product life cycle. Similar to architectural design, as near as in the 1980s, requirement management in software engineering relied on manual techniques. The outcomes of these efforts were documents that were often hundreds of pages long and difficult to comprehend. Hence, they did not receive much use.

The emergence of tools specifically designed for requirement management occurred in the mid 1990s. These tools provide information management capabilities with built-in databases, document templates and functionalities to extract requirement information from existing documents. While the leading domain of applicability is software and systems engineering, most of the requirement management tools are built upon generic capabilities of the commonly utilized desktop applications like Microsoft Word, Access and Excel.


DOORSnet | Proposals | Search | Sort | Display | Help
← Refresh →
Explore | Preferences | Print | Get entire module | Save | Log off
Microsoft Word
File Edit View Insert Format Tools Table Window Requirements Help
Requirements
Heading 1
Arial
16
Fig 1.2.1
4th Qtr
3rd Qtr

System Reqs : Standard View (58 to 88 of 120)

Car System requirements	Priority	Acceptability
1.2 Control car	Not Set	Not checked
1.2.1 Switch on car	Low	Not checked
The car shall be able to discriminate which authorized people shall be able to switch on and operate the car.	Med	Acceptable
1.2.2 Control speed	High	Not checked
The car shall have a foot mechanism to control the speed of the car.	High	Not checked
The speed control shall be infinitely variable from zero to maximum speed.	High	Not checked
The speed of the car shall be controllable by automatic means.	High	Not checked

User Requirements.doc

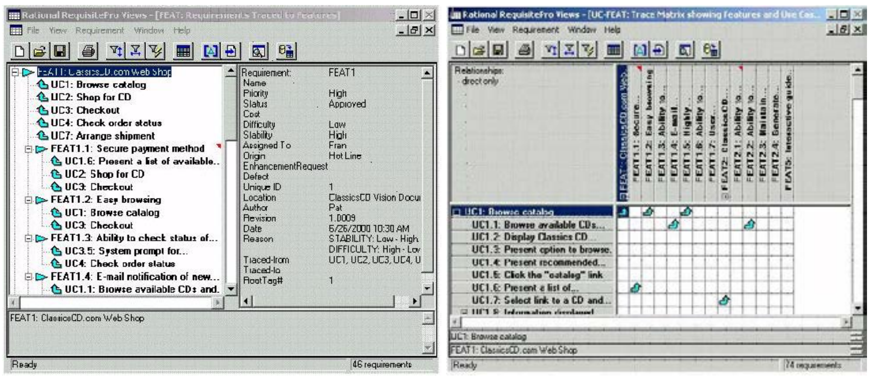
1. User Requirements
1.1 Leave House Secured
1.1.1 Enable alarm
[[UR1] The householder shall be able to start to enable the alarm from within the h
[[UR2] The householder shall receive audible confirmation that the alarm is enable
1.1.2 Close main door
[[UR3] The householder shall receive audible confirmation that the main door is ful
1.2 Alarm
1.2.1 Detect intrusion
[[UR4] Intrusion when the alarm is enabled shall be detected (constraints: within 30 seconds of entering the protected household; even if mains power has been disconnected for any period under 24 hours)]
[[UR5] The remote alarm centre shall be informed by telephone when an intrusion is detected (constraints: within 30 seconds of detection).]

Figure 5: *Example screens from Doors illustrating web based interaction and requirement definition through the Word interface*

Requirement engineering in software design is designated as an information management problem that includes organization, traceability, analysis, and visualization aspects. Most of the existing requirement management tools to support software design provide two-way functionalities to enter requirements into the system; either directly using the application interface or using variety of file formats like those of Microsoft Office applications and HTML. The structure of requirements hence is captured as strings. The tools commonly have versioning capabilities along with database backup that stores requirement information. The database support, however, does not refer to any common data model. It is usually internal to the application, thus posing semantic bridging problems when interoperability is needed.

Requirement traceability is one of the attractive functionalities of software requirement management tools. Traceability can be defined as the ability to relate requirements to each other by parent-child relationships and traced-to and traced-from relationships. In cases of change, by the aid of such defined hierarchies, the application warns the users to suspect the related requirements and revisit them for editing in accordance with the changes made.

DOORS by Telelogic, Inc. and Rational Requisite Pro by IBM are typical examples to software requirement management tools (Figure 5 and Figure 6). These, and many other quite similar ones in software engineering, provide generic, yet powerful functionalities for document management. History record definition, configuration management, version control, filtering data by project management attributes, and import and export capabilities to other applications like Microsoft Word, reduce the burden on the user significantly. Web based interfaces enhance collaborative project management and provide quicker feedback to the projects. Moreover, it is possible to view data in graphical format that enables the users to utilize analysis and visualization techniques.


Rational RequisitePro Views - [FEAT: Requirements Traced to Features]
File
View
Requirement
Window
Help
Ready
46 requirements
Rational RequisitePro Views - [UC-FEAT: Trace Matrix showing Features and Use Cas...
Ready
74 requirements

```
FEAT1: ClassicsCD.com Web Shop
    UC1: Browse catalog
    UC2: Shop for CD
    UC3: Checkout
    UC4: Check order status
    UC7: Arrange shipment
  FEAT1.1: Secure payment method
      UC1.6: Present a list of available..
      UC2: Shop for CD
      UC3: Checkout
  FEAT1.2: Easy browsing
      UC1: Browse catalog
      UC3: Checkout
  FEAT1.3: Ability to check status of...
      UC3.5: System prompt for...
      UC4: Check order status
  FEAT1.4: E-mail notification of new...
      UC1.1: Browse available CDs and.
```

Requirement:	FEAT1
Name	
Priority	High
Status	Approved
Cost	
Difficulty	Low
Stability	High
Assigned To	Fran
Origin	Hot Line
EnhancementRequest	
Defect	
Unique ID	1
Location	ClassicsCD Vision Docu
Author	Pat
Revision	1.0009
Date	6/26/2000 10:30 AM
Reason	STABILITY: Low - High DIFFICULTY: High - Lov
Traced-from	UC1, UC2, UC3, UC4, U
Traced-to	
RootTag#	1

FEAT1: ClassicsCD.com Web Shop

Relationships:
direct only

	FEAT1: ClassicsCD.com Web...	FEAT1.1: Secure...	FEAT1.2: Easy browsing	FEAT1.3: Ability to...	FEAT1.4: E-mail...	FEAT1.5: Highly...	FEAT1.6: Ability to...	FEAT1.7: User...	FEAT2: ClassicsCD...	FEAT2.1: Ability to...	FEAT2.2: Ability to...	FEAT2.3: Maintain...	FEAT2.4: Generate...	FEAT5: Interactive guide...
UC1: Browse catalog	↗		↗			↗								
UC1.1: Browse available CDs...					↗						↗			
UC1.2: Display Classics CD...														
UC1.3: Present option to browse.														
UC1.4: Present recommended...														
UC1.5: Click the "catalog" link														
UC1.6: Present a list of...		↗												
UC1.7: Select link to a CD and...									↗					
UC1.8: Information displayed														

UC1: Browse catalog

FEAT1: ClassicsCD.com Web Shop

Figure 6: *Traceability matrix and traceability tree editable graphical views in Rational Requisite Pro.*

6. Observations

The tools and approaches reviewed in this paper sample some of the representative features of CARM. The large number of emerging software tools which address information tracking, change management and collaborative environments is a clear indicator that CARM is becoming increasingly more desirable. Generally we observed the following trends in our review of the current approaches:

Dependency on generic tools: The state-of-the-art in CARM relies heavily on the capabilities of generic office applications. Applications that are most often used are those which cull common features from commercial, generic office applications and bridge them with requirement management specific functionalities. Many of the existing tools provide requirement management specific interfaces on top of database functionalities. Such tools help with easier navigation in the specification space. They help in organizing and categorizing otherwise hard to track specifications. The assistance these tools provide includes ease of access and editing, multiple visualization capabilities (charts, tables, and tree views) and project management capabilities, such as versioning, time stamping, and identity tracking. While these functionalities are already common in database applications, lack of an agreed upon data model use decreases the wider applicability of such tools.

Support for complex functionalities: Requirement traceability and verification are at the top of the list of functionalities that CARM should support. One of the big challenges is to devise techniques that will be effective while keeping cognitive loads of usability under check. The tools we have reviewed do not support: the ability to trace various segments of the information captured, to verify the information along different design phases, to build a viable requirement structure to perform complex analyses like traceability, to have advanced search capabilities, and effectively visualization of the data. While some applications provide support for some of the identified characteristics, they are stand-alone and do not provide a complete feature set.

Design integration: The computational support serves mostly only for document management purposes when the information carried by design requirements is not directly utilized to assist the user with design related decision making. A sound requirement structure with design awareness is most beneficial. Tasks of generation, validation, verification and propagation are desired functionalities since they support change management, communication enhancement, error tracking and requirement information navigation.

Similarly, intent tracking is seldom provided in CARM tools. DIT, reviewed in section 3.3.4, is one of the few tools focusing on this issue in the relatively limited domain of energy efficiency. Tracing the designer’s motives requires the tracking of her moves in the requirement space as she explores the design space. Such a feature would illustrate the causal chain back to the initial design intent, which is formed mostly from qualitative requirements. It also would offer a strategy for change management.

Structured requirements in the digital medium can provide ease of communication through digitally shared CAD documents. Moreover, along with traceability functionalities, such features naturally provide the ability to track the decisions both backwards and forwards. When changes occur, notification of the designer, automated propagation of changed values and automated checking for inconsistencies, provide the requisite assistance for error tracking.

The building product data modeling studies have started to focus on extending data models to include requirement specific information. This will require CAD applications that can manipulate requirement data and populate the product models effectively. As more systematic ways to digitally represent information are developed, navigating such information spaces will become a high priority for designers since this will enable more effective ways of using the existing requirement information. Methods that provide ways of managing greater degrees of complexity should subsequently translate into improved design performance.

Still lacking, however, are approaches that treat requirement information management as part of design exploration beyond geometric attributes of design. Requirements carry other types of information, such as financial, temporal and behavioral aspects of buildings, which cannot always be captured alone with geometries. A focus on CARM necessitates the immediate focus on the maintenance of temporal and behavioral attributes of design from its inception through its usage, and subsequent possible reuse. Moreover, it is essential that these efforts should treat CARM as a medium to facilitate information based design thinking.

7. Conclusion

In this paper, we reviewed the state of the art in computer aided requirement management in design computing. The past decade has witnessed an increasing number of tools and approaches that strive to organize and utilize information that is generated during the early stages of design more efficiently and correctly. While there are numerous studies, the domain is still far from maturity. We believe one of the dominating reasons for this is the fact that research efforts applied in this domain are dispersed. The need for CARM does not yet occupy a central position in research communities, as do those dealing with building product data management.

Several common functionalities, however, emerge through the tools and studies we reviewed in this paper. Information organization, effective navigation, retrieval and change, and collaborative design environment support are crucial for successful CARM software development. History tracking, traceability and change management appear to be the underlying technologies essential to realize these primary goals within a basic computational decision support environment. We believe the maturity and productive use of early design information is more plausible with a design life cycle management approach. We have illustrated the changing perspectives to design, like LEED, Whole Building Design and Continuous Commissioning©, which increase the desirability of application support for

CARM. We believe the next decade of research and application development will bring a shared CARM vocabulary, similar to the one design computing has been enjoying with drafting tools.

References

ACSIM: 2003, http://www.hqda.army.mil/acsimweb/homepage.shtml, Last viewed November 10, 2003.

ACTS: 2003, http://acts.belvoir.army.mil/, Last viewed November 10, 2003.

Akin, O, Rana S, Magd D, Ye Z: 1995, "SEED-Pro: Computer assisted architectural programming in SEED" *ASCE Journal of Architectural Engineering*, 1(4), pp. 153-161.

Akin, O, Turkaslan-Bulbul, MT, Brown, S, Kim, E, Akinci, B, and Garrett, JH: 2003, Comparison of AHSRAE Guidelines with Building Commissioning Practice *Proceedings of the 11th National Conference on Building Commissioning Palm Springs, California, May 20–22, 2003.*

Akin, O, Turkaslan-Bulbul, MT, Garrett, JH, Akinci, B, and Wang, H: 2004, "Embedded Commissioning for Building Design" unpublished manuscript submitted to *Design Computing and Cognition '04 Conference* to be held at MIT, Cambridge, MA, 19-21 July.

BSD: 2003, http://www.bsdsoftlink.com/. Last viewed November 10, 2003.

Bjork, B: 1991, "Intelligent Front-ends and product models" in Artificial Intelligence in Engineering, 6(1), pp. 47-55.

CERL: 2003, http://fc.cecer.army.mil/. Last viewed November 10, 2003.DIT: 2003, http://ateam.lbl.gov/DesignIntent/. Last viewed November 10, 2003.

Donia, M: 1998. *Computational Modeling of Design Requirements for Buildings*. Ph.D. Thesis, School of Architecture, Carnegie Mellon University, Pittsburgh, PA.

Eastman CM: 1999, *Building Product Models: Computer Environments Supporting Design and Construction*, CRC Press, NY.

Ekholm, A and Fridqvist, S: 1999, "The BAS-CAAD information system for design principles, implementation, and a design scenario" in *Computers in Building, Proceedings of the CAADfutures '99 Conference*, G. Augenbroe & C. Eastman eds, Kluwer Academic Publishers, Boston.

Elbibany, H, Bechtel, J, Brach B, and Ault, D: 1997, "Facility Management Value-Adding Functional Analysis Model" *ASCE*, *Journal of Architectural Engineering* December.

EngRegs: 2003, http://www.usace.army.mil/publications/. Last viewed November 10, 2003.

Erhan, H: 2003. *Computer Aided Support for Building Recurring Building Types*. Ph.D. Thesis, School of Architecture, Carnegie Mellon University, Pittsburgh, PA.

Flemming, U and Chien, SF: 1995, "Schematic Layout Design in the SEED Environment." *Journal of Architectural Engineering, ASCE*, 1(4), pp. 162-169.

Flemming, U & Woodbury, R: 1995, "Software Environment to Support Early Phases of Building Design (SEED): Overview", *ASCE Journal of Architectural Engineering*, 1(4), pp. 162-169.

Galle, P: 1995, "Towards integrated "intelligent" and compliant computer modeling of buildings" *Automation in Construction* 4(3) pp.189-211.

Gamma, E, Helm, R, Johnson, R, Vlissidis, J: 1995, *Design Patterns: Elements of reusable Object-Oriented Software*, Addison-Wesley, NY.

Garrett, JH, Kiliccote, H, and Choi B: 1995, "Providing formal support for standards usage within SEED". *Journal of Architectural Engineering, ASCE*, 1(4), pp. 187-194.

Hutchins, EL.: 1995, *Cognition in the Wild*, The MIT Press, Cambridge, Mass., and London, England.

IAI: 2003, Early Design IFC, Construction Engineering Research Laboratory, http://www.iai-na.org/technical/early_design.php, Last viewed December 15, 2003.

Kamara, JM, Anumba, CJ, and Evbuomwan, NFO: 2000, "Computer-Based Application for the Processing of Clients' Requirements in Construction", *ASCE Journal of Computing in Civil Engineering*, 14 (4), pp 264-271.

Kviniemi, A and Fischer, M: 2003, Requirements Management with Product Models, nD Workshop, Salford University, Manchester.

Leffingwell, D & Widrig D: 2000. Managing software requirements: a unified approach Addison-Wesley, Reading, MA.

Metracker: 2003, http://eetd.lbl.gov/btp/buildings/hpcbs/Element_2/Metracker/02_E2_Metracker.html, Last viewed December 14, 2003.

MDS: 2004, http://bc.cecer.army.mil/mds/. Last viewed March 30, 2004.

Polanyi, M: 1995, *Personal Knowledge: Toward a Post-Critical Philosophy*, Harper Torchbooks, New York, NY.

Rittel, H and Webber, M: 1973, "Dilemmas in a General Theory of Planning", *Policy Sciences*, Vol. 4, pp. 155-169, Amsterdam, Elsevier.

RPLANS: 2003, http://rplans.hqda.pentagon.mil/Overview.asp. Last viewed November 10, 2003.

Solibri, 2003: http://www.solibri.com/services/public/main/main.php. Last viewed November 10, 2003.

Suh, NP: 2001: *Axiomatic Design, Advances and Applications*, Oxford University Press, US.

Trelligence: 2003, http://www.trelligence.com/. Last viewed November 10, 2003.

UFGS: 2003, http://www.ccb.org/ufgs/ufgs.htm. Last viewed November 10, 2003.

WBDG: 2003, http://www.wbdg.org/about.php. Last viewed November 10, 2003.

Woodbury, RF and Chang, TW: 1995, Massing and enclosure design with SEED-Config. *ASCE Journal of Architectural Engineering*, 1(4), pp. 170–178.