Computer-Aided Design 44 (2012) 432–444

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Ontology based interface design and control methodology for collaborative product development

Keyvan Rahmani *, Vincent Thomson [1]

Department of Mechanical Engineering, McGill University, 817 Sherbrooke St. West, Montreal, Quebec, H3A 2K6, Canada

ARTICLE INFO

Article history:
Received 20 January 2011
Accepted 11 December 2011

Keywords:
Ontology
Interface design
Interface control
Collaborative product development
Interface compatibility
Rule based system

ABSTRACT

Interfaces between subsystems in collaborative product development projects are presently defined by interface control documents. This paper presents a computer aided methodology for defining and controlling subsystem interfaces. Interfaces are considered as interconnections between subsystem ports. Ports are specified by using an ontology that ensures consistency of interface definitions among different design teams. Every port that is based on the ontology is eventually defined by a set of attributes that are derived from its form and function. Interfaces between ports are formed when ports are mated. The essence of port mating is described by logical information that is expressed in two forms. First, a set of requirements are defined for an individual port to ensure that it functions properly. Second, connectivity rules are expressed between ports to guarantee that they integrate correctly. A software architecture that operates on port information and controls the status of subsystem interfaces during collaboration is described. A piece of software is implemented based on the proposed architecture and its functionality is demonstrated by two examples. The examples show how the software can be used to replace interface control documents and support collaboration. The software allows designers to load subsystem descriptions from a shared repository and connect them together by defining connectivity rules. The software reports errors to designers when port requirements or connectivity rules are violated.

1. Introduction

An interface refers to any logical or physical relationship required to integrate the boundaries between systems or between systems and their environment. Here, the word 'system' refers to a set of interoperable elements compatible with each other in form, fit and function to achieve a specific outcome [1]. Interfaces can be regarded as places where the boundaries of two subsystems come together. The places of intended interactions among subsystems are called ports [2].

Interface control is the process of identifying all functional and physical characteristics of interacting entities from different organizations, and of ensuring that proposed changes to these characteristics are assessed and approved before implementation [3]. Interface definition and control is an indispensable part of a systems engineering process. It usually occurs after the conceptual design phase. The more carefully subsystem interactions are defined in these early phases, the more likely it is that products are delivered on time with fewer design errors [4].

Interfaces between subsystems in collaborative engineering design projects are defined by interface control documents (ICD). Documenting agreements and committing to them is a crucial means of preventing design conflicts. The purpose of an ICD is to guarantee that subsystems designed by different engineering agencies are compatible. An ICD specifies what is required to correctly connect subsystems in an overall product.

Use of ICDs is particularly helpful when a product is composed of subsystems that are described by different models, e.g., mechanical, electrical, hydraulic, etc. In such a situation, it is very difficult (if not impossible) to have a product representation that includes all such miscellaneous subsystems and defines the interfaces between them. In these cases, ICDs can be supplied as separate documents that describe the interfaces among subsystems.

An important consequence of using ICDs during collaborative product development is that ICDs make projects document driven. This naturally has some drawbacks since the form of ICDs differ substantially from one organization to another. There are standards for the format of ICDs in certain domains, such as aircraft stores [5], but there is no universal standard for the content and form of ICDs. The common practice of using natural language, technical drawings, graphs, etc. to create ICDs leads to ambiguities

* Correspondence to: Macdonald Engineering Building, 817 Sherbrooke Street West, Montreal, Quebec, H3A 2K6, Canada. Tel.: +1 514 398 6296; fax: +1 514 398 7365.

E-mail addresses: keyvan.rahmani@mail.mcgill.ca, keyvan.rahmani@gmail.com (K. Rahmani), vincent.thomson@mcgill.ca (V. Thomson).

[1] Macdonald Engineering Building, 817 Sherbrooke Street West, Montreal, Quebec, H3A 2K6, Canada. Tel.: +1 514 398 2597; fax: +1 514 398 7365.

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.12.002

in the presence of diversely different terminologies. This not only makes the interface control process manual and time consuming, but also makes it difficult to find common interfaces for reuse. These difficulties can be alleviated by using a computer aided interface control methodology in which interface information is given in formal machine readable form.

This paper proposes a computer assisted methodology for interface design and control. The main entities in this methodology are a port ontology that explicitly specifies port related concepts, and interface rule sets that describe how these ports are related in a product development project. The ontology provides a common vocabulary for interface definitions; so, it helps to overcome the lack of commonality in interface terminologies and improves information sharing among agents. When all collaborating agents commit to a shared ontology, the interface specifications provided by them are consistent and can be managed by software tools. The rule sets are also essential components of a computer aided interface control process because ICDs actually define the *logic* for using and connecting subsystems in a product development process. This work can be regarded as a foundational step toward making a standard model for computer aided interface design and control.

2. Related work

In this article, a port based ontology is used to share interface information. The two concepts that are worth paying attention to here are ports and ontologies. Both of these concepts have been used in engineering design to improve collaboration in different design phases. Ports are important because they are the primary locations through which subsystems interact. Ontologies are important because they improve communication and information sharing, and therefore collaboration.

There has been reports of using port based representations during assembly modeling and conceptual design. For example, Singh and Bettig [6] suggested that port information be added to part models in order to automate the process of applying mating constraints in assembly models. They discussed different schemes to capture the attributes of assembly ports; so, their method was only limited to geometric design. While their work gave some guidelines about how to group a part's faces to define assembly ports, it did not discuss a communication methodology and a conceptual model to define generic connectivity relationships among subsystems.

Counsell et al. [7] formalized the connections between different port classes to support the design of mechatronic systems. In their formalization, material, information and power attributes were considered, but there was no indication of geometric attributes. Paredis et al. [8] used port based representation to define behavioral models of mechatronic system components. The port based paradigm was used to simulate system behavior to see whether the functional requirements of systems are met or not.

Because the approaches of Counsell et al. [7] and Paredis et al. [8] were intended for conceptual design and simulation of system dynamics, they did not discuss how interface knowledge could be represented in a collaborative environment where it continuously evolves; but, this is required for interface control, which happens after conceptual design. To address these issues in this paper, a rule based system is proposed that allows designers to externalize interface logic and to define compatibility among subsystems as rules. Externalization means that interface rules are defined outside of the subsystem specifications or any prospective software that manages them.

We seek to use ontologies as a means of ensuring consistency of interface definitions. This is important because interface data is so amorphous that there is no single data model that can represent it. If automatic interface control software is to be supplied, one needs to ensure at least that information coming from different engineering teams is consistent. Ontologies are very useful in enforcing consistency.

Ontologies have been initially proposed by the artificial intelligence community as a means of overcoming difficulties caused by disparate terminologies, approaches and tools in knowledge representation [9]. Ontologies have also been used in the engineering design community for similar purposes. For example, use of ontologies has been proposed during conceptual design [10] and assembly design [11] to improve information sharing.

By combining port based representations and ontologies, Liang and Paredis [2] proposed a port ontology to support incremental refinement of design decisions made during conceptual design. The ontology contained classes to define ports and their attributes. Port attributes were defined by taking into account three different design perspectives: form, function and behavior. Form attributes described all geometric characteristics of ports. Function attributes expressed the intended use of ports and were associated with flows of material, energy and signals. The behavior attributes were described by effort and flow power conjugate complements [12]. In this article, we propose a similar, but broadened ontology for the purpose of interface control.

As in the work of Counsell et al. [7] and Paredis et al. [8], the work of Liang and Paredis [2] also did not discuss a design support software for collaborative interface control. Although they made the definition of ports and interfaces formal, they lacked a connectivity model that allows full communication among independent subsystem developers. Besides, they did not provide a model that could cope with continuously changing interfaces; a model that can be managed by a piece of software without reprogramming it.

Representing interface information for use in CAD systems has also been the subject of some research. Bettig and Gershenson [13] investigated how interfaces of modular products could be represented in CAD and product data management (PDM) systems. Interfaces were regarded as functional relationships among modules and different ways of representing them were compared based on effort and flexibility criteria.

However, Bettig and Gershenson's approach did not discuss the model and mechanisms to enforce connectivity of subsystems when they are developed by collaborative teams. In such a set up, not only is it important to define interface information consistently and allow the definitions to evolve continuously, but also it is important to have mechanisms that allow tracking the source of conflicts between subsystems; that is, a mechanism is required that shows which interfaces are incompatible, what caused the incompatibility, and where exactly it occurred. In this way, the designers can rely on software tools to remedy the problem, rather than organizing a design review meeting to review ICDs and see if any interface requirement is violated.

The core of our paper defines a formal computer based methodology for interface control. The essential application of such a methodology is during collaborative product development to prevent design errors, not in conceptual design or simulation of system dynamics as proposed by most of the above references. To the authors' knowledge, very little research has been published to date on the issue of computer aided interface control despite its importance. In Section 3, we propose the essential components of a computer aided interface control methodology. A piece of software that is built based on these components as well as its application to the product development process is illustrated by two examples in Section 4.