A part affordance-based approach for capturing detailed design knowledge

Yong Chen *, Jian Huang, Zhinan Zhang, Youbai Xie
School of Mechanical Engineering, Shanghai Jiao Tong University, China

1. Introduction

In a competitive environment, engineering designers are often challenged to develop robust designs with less time and lower cost. To meet this challenge, a possible strategy is Design Knowledge Reuse (DKR), i.e. reusing the design knowledge from previous completed projects in a current one [1]. Although commercial CAD (i.e. Computer-Aided Design) or PLM (Product Lifecycle Management) systems are good at building or managing geometrical data and models, it is widely acknowledged that they are ineffective to help designers manage design knowledge for reuse. Therefore, DKR has attracted much attention from the engineering design community in recent years [1–4].

However, existing DKR approaches are primarily for modeling and reusing conceptual or early design knowledge, with very few dealing with detailed design knowledge. Here, detailed design refers to a design process that transforms a solution concept into a complete geometrical description about an artifact. In matured manufacturing enterprises, similar artifacts are often designed in a similar way, which implies that there exists a large amount of detailed design knowledge that can be reused. Therefore, our recent research has developed a formal framework for modeling detailed design knowledge for reuse [5]. However, this framework failed to provide designers with a systematic approach for capturing detailed design knowledge. As a result, much detailed design knowledge, which often hides behind geometrical models, still has to remain in designers’ memories as tacit knowledge, and can easily get lost due to oblivion or the mobility of designers [1]. This is especially the case for the contextual knowledge about an existing design, although such knowledge is often regarded as critical for designers to understand the existing design for reuse [1,3]. Note that the contextual knowledge here deals with not only functional requirements, but also various lifecycle (e.g. manufacturing, assembling, transportation, maintenance, etc.) factors, which are often implicit in a detailed design. If such lifecycle factors are not sufficiently considered, a detailed design can easily get problematic later, e.g. unable to be manufactured, difficult to be assembled, etc. Therefore, it is indispensable to have an effective approach for capturing such detailed design knowledge, in order to implement the DKR strategy successfully.

Since detailed design primarily deals with designing the parts of an artifact, this paper will develop a part affordance-based approach for assisting designers in externalizing and capturing detailed design knowledge. It is an extension of our recent work on modeling and capturing detailed design knowledge for reuse [5,6]. Here, the concept of affordance comes from the relational theory for design [7,8], which Maier and Fadel have developed from the perceptual psychology to overcome the drawbacks of existing transformation-based design theories (e.g. the systematic approach by Pahl and Beitz [9]). There are two significant reasons to adopt the affordance concept in this research. One is that the
parts in detailed design are often treated as non-transformative objects, which means existing transformation-based design theories cannot be applied [7]. The other is that the affordance concept allows designers to manage various lifecycle factors [7], and therefore makes it possible for designers to capture the contextual knowledge related to such lifecycle factors.

This paper is organized as follows. Section 2 presents a concise review of the related work. Section 3 briefly introduces our part representation model. Section 4 proposes the concept of part affordance and its representation. Section 5 develops a part affordance-based approach for capturing detailed design knowledge. With a mechanical fixture design as a case, Section 6 illustrates how the detailed design knowledge-capturing approach works. Following a discussion in Section 7, Section 8 then concludes this paper.

2. Related work

Engineering design is widely acknowledged as a knowledge-intensive process. There are multiple streams of design knowledge-related studies, e.g. engineering knowledge management [10], knowledge-based engineering [11], design rationale [12], etc. Due to limited space, only some typical DKR-related studies are briefly reviewed as below.

To achieve effective DKR, it would be desirable to have a formal model to represent design knowledge, so that a computer-based DKR tool can be developed. Therefore, Gero [13] has developed the Function–Behavior–Structure (FBS) model to represent the conceptual knowledge about a design prototype. Goel and his associates have developed the Structure–Behavior–Function (SBF) model to represent conceptual design knowledge [14]. Szykman et al. have developed an object-oriented design representation language to model the function–behavior–form knowledge about an artifact [15]. Baxter et al. [16] have developed an integrated design knowledge reuse methodology through bringing together best practice reuse, design rationale capture and knowledge-based support. Li et al. have developed an ontology-based tagging approach for transforming design documents into structured XML knowledge to achieve effective retrieval of design knowledge [17]. Bracewell et al. [18] have developed the DRed tool for capturing design rationale, where a directed graph of dependencies is employed to organize the issue–option–argument knowledge in a design. Our recent research has developed a knowledge-based approach for reusing multidisciplinary principle solutions through design synthesis [19]. It can be found that existing DKR-related approaches primarily deal with conceptual or early design knowledge. Such approaches fail to take into consideration various lifecycle factors, which are implicit but very common in detailed designs, and, therefore, are ineligible for helping designers capture and model detailed design knowledge (especially the related tacit knowledge).

Since detailed design is closely related to various lifecycle factors, it is also necessary to give a brief review of concurrent engineering studies. Concurrent engineering is regarded as a methodology which incorporates the considerations in various downstream product development phases (e.g. manufacturing, assembly, maintenance, etc.) into the design phase for producing a design with the best overall product lifecycle performance [20]. Earlier concurrent engineering studies primarily deal with how to transform geometrical model information into manufacturing process information for process planning, e.g. the feature recognition approach [21]. More recent studies have been focused on design for manufacturing, which usually employs some rules-of-thumb knowledge to evaluate the manufacturability of a mechanical part, e.g. the constraint-based approach by Feng and Kusiak [22], the domain-independent DFM Shell by Zhao and Shah [23]. Although some concurrent engineering approaches have been adopted in industry, they cannot effectively support the DKR strategy. A primary reason is that concurrent engineering systems are often black-box systems, where the design knowledge has been encapsulated and hard-coded and therefore is invisible to designers.

In addition, since design knowledge often deals with design constraints, it is also helpful to briefly review the design constraints-related research work. The constraint solving approaches in commercial CAD software (e.g. [24]) are primarily for achieving parametric geometrical modeling, and therefore need not be reviewed here. Many constraint-based engineering design approaches have been developed for conceptual design or embodiment design, which usually assume that a set of design constraints can be given and are aimed at developing suitable representations and algorithms for generating an optimal solution that can meet those constraints (e.g. in [25–27]). However, these approaches can merely result in case-specific solutions, and cannot serve as a general design knowledge processing platform for different kinds of artifacts. They also do not address the issue of how to capture the constraint knowledge in detailed design. There is also some constraint-related design research work focused on constraint management. For example, Goonetilleke et al. [28] have developed a version management approach for managing the evolution process of design constraints; Navinchandra et al. [29] have proposed a design fusion approach for managing various lifecycle constraints in a black board system, which allows stakeholders in different lifecycle periods to collaboratively evaluate a proposed solution. The constraint management approach proposed by Ajit et al. [30] should be most relevant to our work, which is implemented as the computer-based system, ConEditor+, to allow expert designers (rather than knowledge engineers) to model and maintain design constraints. A common drawback of these constraint management approaches is that they lack a systematic approach for helping designers capture the contextual knowledge behind design constraints, which would make it difficult for designers to understand the context about such constraints before the related design knowledge is reused.

It can be concluded that there still exists no effective approach for helping designers externalize and capture the tacit knowledge in detailed design for reuse. As mentioned before, detailed design knowledge is often related to various implicit lifecycle factors, which can be managed through the affordance-based design theory. Therefore, it is reasonable to take an affordance-based approach for capturing and modeling detailed design knowledge.

3. Part

According to prevalent commercial CAD systems (e.g. Pro/Engineer, UG, etc.), a part refers to an individual entity that cannot be decomposed into smaller entities. An entity here is equivalent to a thing in Bunge’s Scientific Ontology [31]. The information model for representing a part is shown in Fig. 1. Based on the generic bills-of-materials approach [32], a part is conceptually described with a name, and some physical properties (e.g. density, rigidity, stiffness, etc.). In addition, the part information model also has a form information sub-model and a behavior information sub-model to represent detailed design-related information.

The form information sub-model describes the form information of a part. It includes not only a 3D model built with a commercial CAD system, but also a descriptive feature-based model to represent the geometrical information, which is stored in a relational database to facilitate our design knowledge capturing and modeling system to access the form data [5,6]. The descriptive feature-based model of a part is composed of some descriptive geometrical features and some location relation parameters. A geometrical feature, which can be user-defined, is further described
with some geometrical parameters, while a relation parameter describes a location relation between two geometrical features in a part. Based on the object-oriented syntax, a geometrical feature (i.e. \( \text{Feat} \)) of a part can be described as, \( \text{PART} \rightarrow \text{Feat} \), while a parameter (i.e. \( \text{param} \)) of a part feature can be represented as, \( \text{PART} \rightarrow \text{Feat} \rightarrow \text{param} \). Similarly, a relation parameter (\( \text{rel}\_\text{param} \)) of a part can be represented as: \( \text{PART} \rightarrow \text{rel}\_\text{param} \).

The behavior information sub-model describes the part behavior information of a part. Here, a part behavior describes a (possible) dynamic change of a part in a specific situation. Note that the behavior concept here is consistent with the scientific ontology [23], while fundamentally different from that mentioned in most engineering design studies, which, according to our recent research [33,34], is more like the concept of (physical) action. For example, different from Gero’s argument that a window has a behavior of restricting ventilation (i.e. restricting airflow) [13], we argue that the window has an action of restricting air flow [25,26]. In our opinion, a behavior of a window is, for example, to extend to a larger open angle [25,26]. In addition, it should be pointed out that part behaviors here include not only the function-related (i.e. intended) behaviors (e.g. the translation of a piston in an engine), but also other negative or positive behaviors in various lifecycle periods of a part (e.g. the deforming behavior of the piston, the wear behavior of the piston, etc.), since such lifecycle behaviors may also be related to detailed design knowledge.

A part behavior is represented with a behavior name and some parameters to indicate its quantitative degrees. Note that a part may have multiple behaviors, and a behavior may also have multiple parameters. For example, the piston in an engine has multiple behaviors, such as translation, deforming, wearing, etc., while its translation behavior also has multiple parameters, e.g. \( \text{max-disp} \) (i.e. maximal displacement), \( \text{max-vel} \) (i.e. maximal velocity), etc. With the object-oriented syntax, the behavior of a part here can be described as: \( \text{PART} \rightarrow \text{Behavior} \), while the parameter of a behavior can be described as: \( \text{Behavior} \rightarrow \text{parameter} \). For example, the maximal displacement of the translation of a piston can be described as: \( \text{PISTON} \rightarrow \text{Translation} \rightarrow \text{max-disp} \).

4. Part affordance

According to Maier and Fadel [7,8], an affordance refers to a relationship between two systems in which a potential behavior (i.e. action in the Scientific Ontology [31]) can occur that would not be possible with either subsystem in isolation. They have developed several affordance-based approaches (e.g. the affordance structure matrix) to support various conceptual design tasks (e.g. synthesis, evaluation, etc.). In this research, we will study how to employ the affordance concept to assist designers in capturing detailed design knowledge.

4.1. Concept

According to our previous studies on detailed design [5,6] and the relational theory for design [7,8], a part affordance here is regarded as a perceived interplay relation that a part has with another entity in any of its lifecycle periods. Note that the perception subject of a part affordance is usually a designer, which is different from that in perceptual psychology. For example, the top part of a computer table is perceived by a designer to afford support to a computer (i.e. is for supporting a computer). It is self-evident that there is always a subject (e.g. the top part) and an object (e.g. the computer) in a part affordance. Since the subject of a part affordance is always a part, it can also be called a subject part. Note that the perceived interplay in a part affordance cannot only be based on a physical action (e.g. the force-based action), but also can be based on other kinds of relations (e.g. layout relation). For example, the top part mentioned above not only has a force-based affordance of supporting a computer, but also has a layout-based affordance of accommodating a human user. In addition, it should be pointed out that a part affordance may often involve the interplay relation with an undesirable object or with an undesirable behavior. For example, the affordance of a cover in a machine is for blocking dust, where dust is an undesirable object.

It can be found that the concept of affordance somewhat resembles that of function. However, there are also some major differences between these two concepts, as discussed in [7]. One major difference is that an affordance (e.g. the aforementioned affordance, \( \text{accommodating a human user} \)) does not require an input–output flow transformation, which, however, is necessary for a function. Another major difference is that an affordance can deal with an affordance object perceived in any lifecycle (e.g. manufacturing, assembling, operation, maintenance, etc.) period of an artifact, while a function can merely deal with an object (i.e. a flow) in the operation (i.e. working) period of an artifact. For example, when designing a mechanical part, a designer probably perceives the use of a wrench for assembling it, which means that this mechanical part should have an affordance of accommodating a wrench.

Since part affordances can deal with various lifecycle factors that are implicitly considered (i.e. perceived) in a detailed design, it is then reasonable to employ them to capture the tacit knowledge related to the lifecycle factors implicit in a detailed design. Furthermore, since such lifecycle factors actually represent the context of a detailed design, the concept of part affordance therefore paves a way for designers to capture the contextual knowledge about a detailed design. In contrast, existing CAD software or PLM systems are not designed for designers to capture such lifecycle factors (e.g. the wrench mentioned before), and therefore cannot allow designers to capture affordance-based design knowledge.
According to the types of the affordance objects, part affordances can be classified as external part affordances and internal part affordances. An external part affordance describes a perceived interplay relation between a subject part and an external object that is not a part of the artifact that the subject part belongs to, while an internal part affordance describes a perceived interplay relation between a subject part and another part (i.e. the affordance object) in the same artifact. For example, for a computer table, its top part has an external part affordance of supporting a computer, while its leg part has an internal part affordance of supporting the top part. In an internal part affordance, the object can also be called an object part. Internal part affordances can be further classified as static structure-focused part affordances and dynamic behavior-focused part affordances. In a static structure-focused part affordance, the object part (i.e. the affordance object) is treated as a static structure. For example, the leg part of a computer table has a static structure-focused part affordance, i.e. for supporting the top part. In a dynamic behavior-focused part affordance, what is focused on is the dynamic behavior of the object part. For example, the cylinder in an engine has a dynamic behavior-focused part affordance, accommodating the translation (i.e. a moving behavior) of the piston, where what is of interest is the moving behavior of the piston.

Note that the above part affordance classification is primarily for representing part affordances, since different kinds of affordance objects require different representation models, which will be illustrated later (see Section 4.2). Therefore, this classification is different from the affordance classification introduced in the relational theory for design [7], where affordances are classified as AUAs (Artifact–User Affordances) and AAAs (Artifact–Artifact Affordances). It is evident that all internal part affordances here belong to AAAs, since they do not deal with human users. However, external part affordances can be either AUAs or AAAs, depending on whether their objects are human users or external entities. For example, the external part affordances, for supporting a computer and for accommodating a wrench for assembling, are AAAs, while the external part affordance, for accommodating a human user, is an AUA.

4.2. Representation

In existing studies [7,8,35], affordances are often described in a loose manner, e.g. accessibility to all windows to passenger, frustrating user by unnatural mapping to window locations, reducing electronic redundancy, etc. As a result, it is very difficult to employ a formal model to represent affordances to support the capture of detailed design knowledge. Therefore, a formal model is proposed here to represent part affordances.

Since a part affordance can be regarded as a perceived interplay relation between a subject and an object, it is then reasonable to represent it as a triple, (S, V, O). Here, S denotes the affordance subject, V is a verb for describing the affordance (i.e. interplay) relation, and O is the affordance object. Different from the verb–noun pair description for representing functions (e.g. in [36]), both the subject and the object here are associated with their formal representations. Since part affordances fall into three categories (i.e. external part affordances, static structure-focused part affordances and dynamic behavior-focused part affordances), the above model for describing part affordances has three variants. The first variant (A1) is for representing an external part affordance, which can be conceptualized as the triple, (S, V, O). Here, O indicates the external object, which should be further represented with some quantitative parameters related to detailed design. For example, the part affordance of the top part in a computer table, for supporting a computer, can be described as (TOP, SUPPORT, COMPUTER), while the external object, COMPUTER, should be further represented with a set of parameters related to the table design, i.e. \{length, width, height, weight, \ldots\}. To simplify its representation, the parameters of an external object are not classified as form parameters and behavior parameters, which, therefore, is different from the representation model of an internal object (i.e. a part).

The second variant (A2) is for describing a static structure-focused part affordance, which can be represented as the triple, (S, V, Oi). Here, Oi refers to an internal object (i.e. a part different from the subject part), which is associated with its formal part representation. Based on this model variant, the part affordance that a leg part has in a computer table, for supporting its top part, can then be described as \{LEG, SUPPORT, TOP\}.

The last variant (A3) is for describing a dynamic behavior-focused part affordance, which can be represented as the triple, (S, V, Oi). Here, Oi denotes the dynamical behavior of the internal object part of interest. Note that since each behavior belongs to a part, the behavior description here already implicitly contains the object part of a dynamic behavior-focused part affordance. Based on this model variant, the aforementioned dynamic internal part affordance of the cylinder part in a combustion engine, for accommodating the translation of the piston, can then be described as \{CYLINDER, ACCOMMODATE, PISTON \rightarrow Translation\}, where “PISTON \rightarrow Translation” denotes the translation behavior of the piston part.

5. Capturing detailed design knowledge

Detailed design knowledge here is primarily composed of two parts, i.e. the contextual knowledge and the constraint knowledge. The contextual knowledge describes what factors have been considered in a detailed design, and therefore can help a designer understand the context of a detailed design before reuse. The constraint knowledge can illustrate how various factors are formally considered in a reused detailed design, and therefore can be employed to help designers determine the values of design parameters in a new design. A systematic approach is proposed here for helping designers capture detailed design knowledge from part affordances, which is primarily composed of four sequential stages, i.e. capturing part affordances, deriving basic affordability constraints, defining Extreme Affordance Combinations (EACs), and finally building detailed design constraints.

5.1. Capturing part affordances

In detailed design, a designer usually should think of (i.e. perceive) various factors that should be considered for a part at first, before assigning values to its design parameters. Since such factors can be transformed into some intended affordances of a part, it is then possible to capture detailed design knowledge from the affordance descriptions of a part. Note that it is also encouraged that a designer capture some unintended (i.e. unresolved) affordances at this stage, since such unintended affordances, which, although they have not been considered in an existing design and therefore can be regarded as unresolved issues, would be helpful for designers to innovate the existing design in the future.

Given a subject part, the basic process of capturing its part affordance(s) is as follows. At first, a designer should perceive an affordance object that should be considered in a lifecycle period. If the affordance object is related to a dynamic behavior-focused part affordance, the designer should then indicate its related dynamic behavior as well. Thereafter, s/he should choose a suitable verb to describe the desired affordance relation between the subject part and the affordance object. For example, for the top part of a computer table, an affordance object, computer, can be perceived at
first, and then the verb, support, is selected for describing the perceived affordance relation. As a result, a part affordance relation, \( (\text{TOP}, \text{SUPPORT}, \text{COMPUTER}) \), can then be captured. Note that the verb in a part affordance is primarily for human designers to understand the desired affordance relation between a subject part and an affordance object. For a specific kind of artifact, a standard collection of such verbs can usually be provided. Since a subject part often has multiple lifecycle periods, which often deal with different lifecycle objects, it is encouraged that a designer consider each lifecycle period (e.g. manufacturing, assembling, operation, maintenance, etc.) of a subject part one by one. During this process, a designer can employ a checklist to perceive those lifecycle objects in a systematic manner. The checklist can be conceptualized as a tuple: \((\text{subject.part}, \text{lifecycle.name}, \text{affordance.object}, \text{Checked})\). For example, when defining the part affordances of the top part mentioned above, a designer should perceive and then list all possible affordance objects at various lifecycle periods, e.g. the saw machine and the drilling machine in the manufacturing period, the assembling worker, the table legs, and the screwdriver in the assembling period, the human user and the computer in the operation period, etc. Thereafter, he then should select a suitable verb for each perceived lifecycle object to describe a desired affordance relation.

### 5.2. Deriving basic affordance constraints

A part affordance qualitatively describes a desired interplay relation of a subject part with another object. To achieve the detailed design of a subject part, a designer should have some quantitative design constraints on it, so that s/he can assign appropriate values to its design parameters. Therefore, it would be desirable if the intended part affordances captured before can be transformed into some quantitative design constraints on a subject part. Note that the unintended part affordances need not be considered in this step. In this section, we will illustrate how to employ the Affordance Constraint Axiom to derive some basic affordance constraints from intended part affordances. Here, the affordance constraint axiom is extended from the functional constraint axiom we have identified in our previous research [5,6].

**The affordance constraint axiom:**

If an entity \( S \) (i.e. the Subject of a part affordance) is said to have an affordance relation with an entity \( O \) (i.e. the Object of the part affordance), then some design parameter(s) of \( S \) will be constrained by some related design parameter(s) of \( O \).

The affordance constraint axiom holds in each successful design case. A significant reason is that a designer often employs a geometrical feature in a subject part to achieve a part affordance, which makes it possible to build some constraints between the parameters of the subject part’s feature and those of affordance objects [6]. For example, since the top part of a computer table has a part affordance, for supporting a computer, which is achieved by a cuboid feature, it is then reasonable to build two basic affordance constraints, i.e. the width and the length of the top part’s cuboid feature should be constrained by (i.e. bigger than) the width and length of a computer, respectively. Note that a designer can also employ the neighbor space of a subject part to achieve a part affordance [6]. In such a situation, it is also possible to employ the parameter(s) of the subject part to denote the neighbor space [6], which, therefore, still makes it possible for a designer to build some basic affordance constraints between the subject part and the affordance objects. For example, since the above top part (with a rectangular cuboid feature) of a computer table has a part affordance, to accommodate a human user, which is actually fulfilled by the space neighboring the top part, it is then reasonable to build a basic affordance constraint as, \((\text{TOP} \rightarrow \text{Rect.Cuboid} \rightarrow \text{width} >\)\

### Table 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{ij}) ((\text{V} + \text{O}_j))</td>
<td>( p_{ub1} )</td>
<td>( p_{ug1} )</td>
<td>( p_{ub2} )</td>
</tr>
<tr>
<td>( &gt;)</td>
<td>( &lt;)</td>
<td>( &gt;f)</td>
<td>( f)</td>
</tr>
<tr>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( A_{ij}) ((\text{V} + \text{O}_j))</td>
<td>( p_{ub1} )</td>
<td>( p_{ug1} )</td>
<td>( p_{ub2} )</td>
</tr>
<tr>
<td>( &lt;)</td>
<td>( &gt;)</td>
<td>( f)</td>
<td>( &gt;)</td>
</tr>
<tr>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( A_{ij}) ((\text{V} + \text{B}_0)</td>
<td>( p_{ub1} )</td>
<td>( p_{ug1} )</td>
<td>( p_{ub2} )</td>
</tr>
<tr>
<td>( &lt;)</td>
<td>( &gt;)</td>
<td>( f)</td>
<td>( &gt;)</td>
</tr>
<tr>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
</tr>
</tbody>
</table>

**User \( \rightarrow \) width**, which means that the width of the top part then should be bigger than the width of a human user.

Affordance constraints cannot only deal with geometrical parameters, but also can be related to behavioral parameters. For example, from the aforementioned part affordance of a cylinder, for accommodating the translation of the piston, a basic affordance constraint can then be derived between the maximal displacement (a behavioral parameter) of the piston and the length of the cylinder, i.e. \((\text{CYLINDER} \rightarrow \text{Cylinder} \rightarrow \text{length}) > (\text{PISTON} \rightarrow \text{Translation} \rightarrow \text{max} - \text{disp})\). In addition, it should be pointed out that our part representation model does not include any physical actions (e.g. force or moment). Affordance constraints dealing with such actions should be converted into behavior-related constraints. For example, if a cantilever beam is designed to support a heavy object at its end, a basic affordance constraint will be that the maximal elastic deformation (i.e. the deflection behavior) of the beam should be constrained by the weight of the heavy object [5].

Based on the affordance constraint axiom, many basic affordance constraints can then be captured between the parameters of a subject part and those of the affordance objects. Such basic affordance constraints can be illustrated with an affordance constraint matrix shown in Table 1. In the table, the row headed “Sub. param.” contains the parameters of the subject part, which include both geometrical parameters (e.g. \( p_{ub1} \)) and behavioral parameters (e.g. \( p_{ub1} \)), and the column headed “Part aff. info.” denotes the part affordances of the subject part described in the forms shown in Section 4. To derive basic affordance constraints, part affordances here are also further appended with their affordance objects’ parameters, such as \( p_{P} \), \( p_{G1} \), \( p_{G2} \), \( p_{A1} \) where, \( p_{P} \) refers to a parameter of an external object, \( p_{G1} \) denotes a geometrical parameter of an object part, and \( p_{A1} \) is a behavioral parameter of an object part. In the matrix, an intersection cell filled with a comparison symbol (e.g. “\( >\)”, “\( <\)”, “\( =\)”, “\( \rightarrow\)”, “\( \leftarrow\)”, “\( \cdots\)”, “\( \cdots\)”) indicates the constraining relation between a constraining parameter of an object affordance and a constrained parameter of an affordance subject. For example, the symbol “\( >\)”, in the crossing cell, \((1, 1)\), means that \( p_{ub1} \) should be bigger than \( p_{ug1} \). A comparison symbol with “\( \rightarrow\)” means that a design parameter should be processed with a mathematical function \( f \), e.g. \( p_{ug2} > p_{ub1} \).

### 5.3. Defining EACs with part affordances

It is a fact that a subject part often has multiple part affordances. When determining the value of a parameter of a subject part, a designer often should consider some part affordances jointly as a combination, since such part affordances probably should be fulfilled at the same time. However, it is neither possible nor necessary to enumerate all possible part affordance combinations.
in various situations. Instead, a designer should consider some extreme working situations, which can be regarded as some extreme combinations of part affordances, i.e. Extreme Affordance Combinations (abbreviated as EACs later).

A simple approach to define an EAC would be to enumerate all possible part affordances that could occur at the same time and then to combine them as a group. However, since a part affordance often implies multiple basic affordance constraints that are related to different design parameters, this approach would make it difficult later for a designer to derive a detailed design constraint from an EAC.

Therefore, it is stipulated in this research that each EAC must be associated with only one Design Parameter (i.e. constraint from an EAC). This makes it difficult later for a designer to derive a detailed design constraint from an EAC.

Based on the affordance constraint matrix, the process for defining the EACs related to a subject part’s design parameter can be illustrated as below.

Step 1: From the part list of a given artifact, select a part as the current subject part;
Step 2: From the Affordance Constraint Matrix (abbreviated as ASM), select a Design Parameter (abbreviated as DP) of the subject part as the constrained DP;
Step 3: Set the ASM column indexed by the constrained DP as the current column, and select from it a temporary cell that is non-empty and that has not been set as a temporary cell;
Step 4: According to the temporary cell, retrieve the corresponding Part Affordance (abbreviated as PA) as the temporary PA, and get the comparison (i.e. constraining relation) symbol (e.g. “>”);
Step 5: From the current column, get all other non-empty cells that are filled with comparison symbols similar to that in the temporary cell, and then retrieve from ASM the PAs corresponding to these cells;
Step 6: With the help of designers, judge whether the retrieved PAs should happen with the temporary PA at the same time and whether they can interfere, generating a simultaneous PA group (including the temporary PA);
Step 7: Combine the constrained DP and the simultaneous PA group as a temporary EAC;
Step 8: Check whether the temporary EAC has been generated or not; if not, save the temporary EAC as an EAC in a database; otherwise, reject the temporary EAC;
Step 9: From the current column, select another cell as the temporary cell; if successful, return to Step 4 for further EAC definition; otherwise, i.e. all cells in the current column have been analyzed, continue;
Step 10: From the ASM, select another DP of the current subject part as the constrained DP; if successful, return to Step 3 for further EAC definition; otherwise, i.e. all DPs of the subject part have been explored, continue;
Step 11: From the part list, select another part as the current subject part; if successful, return to Step 2 for further EAC definition; otherwise, i.e. all parts of the given artifact have been analyzed, continue;
Step 12: Output all EACs stored in the database, and then exit.

A brief diagram to illustrate the above EAC-defining process is shown as below in Fig. 2. Note that all part affordances in an EAC should be with similar constraining relations, which can be easily found from the comparison symbols. For example, the constraining relation “>” is similar to those of “>,” “>,” “f”. In addition, it should be pointed out that the EAC-defining process is an interactive one, where designers are required to give some information when necessary. For example, since a computer cannot judge whether some part affordances can happen at the same time, a designer then should help a computer make the judgment.

The upper part of Fig. 3 shows an extreme working situation of an office table, where its top part has multiple part affordances at the same time, such as for holding a Notebook (A_{NB}), for holding a Laptop Computer (A_{LC}), for holding a Printer (A_P), for holding a Book Shelf (A_B), etc. The lower part of Fig. 3 shows some basic affordance constraints derived from the related part affordances, which are equivalent to the affordance constraint matrix representation. For example, the item, (BS \rightarrow I, <, \top \rightarrow \text{Rect}_\text{Cuboid} \rightarrow I), is equivalent to a matrix cell located by a parameter of the subject part, Top \rightarrow \text{Rect}_\text{Cuboid} \rightarrow I, and a parameter of the external object, BS \rightarrow I. Based on this extreme working situation, some EACs can then be derived, e.g. \{TOP \rightarrow \text{Rect}_\text{Cuboid} \rightarrow I, (A_{FP}, A_{ALC}, A_{AOP}), \{\text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w, (A_{ABS}, A_{AIP}, A_{ACUB})\}, \{\text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w, (A_{ABS}, A_{AIP}, A_{AUB})\}, etc. Note that the above extreme working situation does not include all part affordances of the top part. For example, the top part probably should still have a part affordance, for supporting an adult (A_{AB}), so that it can support the adult to change a bad light bulb under the ceiling; it also has a part affordance in its transportation period, for passing through a door. Such part affordances should be considered in other extreme working situations.
5.4. Building detailed design constraints

Based on the affordance constraint matrix and the EACs, it is then relatively easy for a designer to construct detailed design constraints. A detailed design constraint consists of four kinds of elements, the constrained parameter of the subject part, the constraining relation, the constraining parameters, and the mathematical operators for connecting these constraining parameters. From an EAC, a designer can obtain the constrained parameter, and the related part affordances. According to the constrained parameter and the part affordances, s/he can then find the constraining relations (i.e. the comparison symbol) and the constraining parameters from the affordance constraint matrix. Since the part affordances in an EAC should be jointly considered, the mathematical operator is often the plus (+). In some situations, there will be other mathematical operators, when some complex constraining relations (e.g. f >, < f, etc.) are involved. In such situations, the mathematical operators usually depend on the domain-specific solution knowledge and should be determined by designers, as discussed in our previous detailed design knowledge modeling research [5].

Still with the extreme working situation shown in Fig. 3 as an example, how to build a detailed design constraint is briefly illustrated here. From the EAC, \( \{\text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w, (A_{BS}, A_{FP}, A_{CUP})\} \), the constrained parameter can be obtained as, \( \text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w \). According to the constrained parameter and the three part affordances in this EAC, the constraining relation can be found as “\( > \)”. The constraining parameters can be obtained from the basic affordance constraints shown in Fig. 3 as, \( BS \rightarrow w, FP \rightarrow l, \) and \( CUP \rightarrow d \). Since these three affordances should jointly impose a constraint on the office table’s top part, the mathematical operator here is then plus (+). Therefore, a detailed design constraint can then be built as, \( \{\text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w, >, BS \rightarrow w + FP \rightarrow l + CUP \rightarrow w\} \).

With some detailed design constraints as above, it is then convenient for a designer to determine the feasible value range of a constrained parameter of a subject part. Note that the values of the constraining parameters of the related affordance objects in a design constraint are either known to designers in advance or should be determined before the constrained parameter. This can also explain why the constrained parameter of a detailed design constraint is a design parameter of a subject part, rather than a parameter of an affordance object. For example, in the aforementioned design constraint, \( \{\text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w, >, BS \rightarrow w + FP \rightarrow l + CUP \rightarrow w\} \), the values of the related parameters of the external affordance objects, \( BS \rightarrow w, FP \rightarrow l, \) and \( CUP \rightarrow w \), are usually known, before the value of the parameter, \( \text{TOP} \rightarrow \text{Rect}_\text{Cuboid} \rightarrow w \), is determined.

6. An illustrative example

With Microsoft Visual Studio (.NET) and Microsoft SQL Server, we have developed a Design Knowledge-Capturing System (DKCS), which has been integrated into our computer-based DKR platform. This platform has been implemented in a mechanical fixture design group for testing. Here, the process of how to capture the detailed design knowledge of a mechanical fixture is employed as an example to illustrate how the proposed design knowledge capturing approach works. An expert designer with about 10 years of experience in fixture design has been requested to model detailed design knowledge with DKCS. The primary process is as below.

The mechanical fixture in this example is designed for fixing a Turbocharger Shell (TS) at a machining stage. Fig. 4 shows two...
primary drawings of the fixture, which comprises 18 parts, such as the Base Board ([BB], ①), the Shell Seat ([SS], ②), Double-Head Bolt ([DHB], ④), the Press Top ([PT], ⑤), the Adjusting Pillar ([AP], ⑥), the Pillar Shelf ([PS], ⑦), four Shell Seat Screws ([SSS], ⑨), four T-Bolts ([TB], etc. Note that the abbreviated names of all internal parts of the fixture are included in square brackets, to differentiate them from the external objects mentioned later; the number in a circle denotes the part number in Fig. 4(b).

First, the expert designer input the basic information about the fixture into DKCS. Here, the primary information is the part list of the fixture. Fig. 5 shows a User Interface (UI) for a designer to manage the part list of an artifact. The added parts are shown in the tables shown in the bottom. Note that identical parts in an artifact here are often given different local names since they could fulfill different affordances. For example, two local names, "6-Ang-Flang-Nut (Up)" and "6-Ang-Flang-Nut (Down)", are given to the two identical 6-Angle-Flange-Nuts, since these two nuts will have different part affordances. This is different from the existing bills-of-materials approach, where identical parts are only given one identification number.

Second, the designer then built the part representation for all parts. With the form representation as an example, Fig. 6 shows a UI for managing the information of the [BB] part. To save the input efforts, the expert designer employed a complex geometrical feature, BB_A_Feat (i.e. Base-Board-A-Feat in Fig. 6), which is defined in another UI, to represent the form of the [BB] part. It can be found that the feature, BB_A_Feat, has many design parameters, such as thickness (h), width (b), length (a), slot-dist (d), etc. Note that there are no relational parameters defined, since the [BB] part is only defined with one geometrical feature. In addition, this mechanical fixture also has some dynamic behaviors in various lifecycle periods. For example, a [TB] part has a translation behavior along the key slot of the [BB] part, when assembled to the [BB] part; the [DHB] part has both a helical motion along the screw hole in the [BB] part, and an extension deform when used to fix a turbocharger; etc. The UIs for managing such behaviors and their parameters are very similar to those for managing geometrical features and their parameters. Due to limited space, these UIs are not shown here.
Third, the designer then captured the part affordances of each fixture part. Fig. 7 shows a UI for managing the part affordances of the fixture parts. From the UI, the designer can select a lifecycle period, an affordance object and an affordance verb to define a part affordance, with the result shown in the bottom table. Note that the affordance verbs here are selected from a standard vocabulary, which are defined in another UI. About 100 part affordances have been captured for all parts of the fixture, which
involve not only internal part affordances, but also external part affordances in various lifecycle periods. For example, the [BB] part not only has some internal part affordances (e.g. for holding [SS], for positioning [SS], for holding [PS], etc.), but also multiple external part affordances in various lifecycle periods, such as for holding TS, for holding Drill Head (DH), for holding Milling Cutter Head (MCH), for occupying the Machining Operation Platform (MOP), for partially holding Adjustable Wrench (AW), etc. Note that the drill head is for making holes in the [BB] part, the milling cutter head here is for manufacturing the key slot on the [BB] part, while the adjustable wrench is for assembling the [BB] part with the machining tool. Therefore, part affordances allow designers to capture various objects (both internal parts and the lifecycle external objects) that have been considered in a detailed design.

Fourth, the designer then derived some basic affordance constraints from the part affordances. Fig. 8 shows a UI for helping a designer derive basic affordance constraints from the part affordances. To define a basic affordance constraint, a designer should select a part affordance (including the subject, verb and object) and the related design parameters from the UI, while the defined basic affordance constraints will be shown in the bottom table. As a result, many basic affordance constraints can then be defined. For example, according to the part affordance of the [BB] part, for holding TS, two basic constraints can be derived, i.e. ([BB] → BB_A_Feat → length, >, TS → length), and ([BB] → BB_A_Feat → width, >, TS → width); according to the part affordance of the [BB] part, to hold [PS], two basic constraints can be derived, i.e. ([BB] → BB_A_Feat → length, >, [PS] → PS_Feat → length), and ([BB] → BB_A_Feat → width, >, [PS] → PS_Feat → width), as shown in row 6 and row 7 in the table of Fig. 8; according to the part affordance, to position [PS], a basic constraint can be derived, i.e. ([BB] → BB_A_Feat → hole_dist, =, [PS] → PS_Feat → hole_dist), where hole_dist is a parameter for describing the distance between two holes, as shown in row 3 in the table of Fig. 8; according to the part affordance, to partially hold [AP], a basic constraint can then be derived, i.e. ([BB] → BB_A_Feat → length, > f, [AP] → length), where f is a mathematical function for indicating the "partially-holding" relation, as shown in the table of Fig. 8; etc.

Fifth, the designer defined the EACs for the parameters of the fixture parts. Fig. 9 shows a UI for helping a designer define EACs, where the designer should select an affordance subject, a design parameter of the subject and a comparison symbol at first, before he gives a name for an EAC and selects possible part affordances for it. Assume that the parameter, ([BB] → BB_A_Feat → length) is selected as a constrained parameter, the process of defining a related EAC is explained here. According to the basic affordance constraint list (partially shown in Fig. 8), this constrained parameter is associated with multiple affordance object parameters, e.g. TS → length, [SS] → SS_Feat → diameter, [PS] → PS_Feat → length, [AP] → AP_Feat → length, [TB] → TB_Feat → diameter, MOP → width, etc. Here, most of the parameters (e.g. TS → length) are in similar constraint relations ("\\( \geq \)" or "\\( > \)")) with the constrained parameter, except the parameter, MOP → width. Therefore, the corresponding part affordance, to occupy MOP, can then be ruled out from those part affordances. According to the fixture layout, it is found that the space indicated by the parameter, [SS] → SS_Feat → diameter, overlaps with that indicated by the parameter, TS → length. Therefore, the part affordance, to hold [SS], should then be removed from the resulting affordability group. As a result, an EAC will then be built, i.e. ([BB] → BB_A_Feat → length, \( \{ \text{TS, PS}_\text{F}, \mathcal{A}_\text{TB}_1, \mathcal{A}_\text{TB}_3, \mathcal{A}_\text{AP} \} \)), as shown in the first EAC in Fig. 9. Similarly, another EAC can also be defined, i.e. ([BB] → BB_A_Feat → length, \( \{ \mathcal{A}_\text{MOP} \} \)).

Finally, based on the basic affordance constraints and the EACs, a designer can then build the detailed design constraints for related design parameters. Fig. 10 shows a UI for a designer to input detailed design constraint into DKCS. Here, a designer should first select an affordance subject, a constrained parameter and an EAC; thereafter, he can then select related constrained parameters from the drop-down list controls to build a detailed constraint. Note that when a designer sets an operand in Fig. 10 as null, he can then expand the null operand into a combinatorial operand in the future, which thus allows him to define a complex constraint. With the aforementioned EAC, ([BB] → BB_A_Feat → length, \( \{ \mathcal{A}_\text{TS}, \mathcal{A}_\text{PS}, \mathcal{A}_\text{AP}, \mathcal{A}_\text{TB}_1, \mathcal{A}_\text{TB}_3 \} \)), as an example, the process of building a detailed design constraint is explained here. The constrained parameter here is ([BB] → BB_A_Feat → length, \( \{ \mathcal{A}_\text{MOP} \} \)).
and the EAC deals with five part affordances, i.e., holding TS, holding [PS], partially holding [AP], holding [TB], and holding [TB]. From the basic affordance constraint list, the constraining relations (i.e. “<” and “>”) can also be found. Here, the symbol, “f”, in the constraining relation, “> f”, originates from the “partially holding” affordance relation, and can be replaced with
an experience-based constant (0.5). Therefore, a detailed design constraint can then be built as: 

\[
[BB] \rightarrow BB\_A\_Feat \rightarrow length \rightarrow \text{TS} \rightarrow length + [PS] \rightarrow PS\_Feat \rightarrow length + 0.5 = ([AP] \rightarrow AP\_Feat \rightarrow length) + [TB]_1 \rightarrow TB\_Feat \rightarrow head\_diameter + [TB]_3 \rightarrow TB\_Feat \rightarrow head\_diameter, \text{ i.e. } \frac{A}{0.5} + \frac{1}{1.5} + \frac{1}{1.5} + \frac{1}{1.5} \quad \text{in Fig. 10.}\]

Here the comparison relation symbol is enclosed in a pair of brackets, e.g. ( ), to avoid ambiguity. Note that the parameters in the above constraint (e.g. TS → length, [PS] → PS_Feat → length, etc.) are described with simplified symbols in the constraint in Fig. 10 (e.g. l₁, l₂, d₁, etc.). When a designer moves the cursor on such a symbol in the UI, he can know its exact meaning through a floating tip, as seen in Fig. 10. Similarly, some other detailed constraints can also be built. For example, according to the EAC, 

\[
[[BB] \rightarrow BB\_A\_Feat \rightarrow key\_slot\_width, (A_{MCH})], \text{ a detailed design constraint can be built: } \text{[BB] → BB\_A\_Feat → key\_slot\_width (\text{=}) MCH → cutter\_diameter, which means that the parameter of the [BB] part, key\_slot\_width, is determined by the parameter of the MCH (i.e. Milling Cutter Head), cutter\_diameter, to achieve the affordance of holding MCH.}\]

7. Discussions

From the above example, it can be found that our part affordance-based approach allows a designer to manage various affordance objects that are related to the detailed design of a part. In particular, it can assist a designer in capturing various external objects in different lifecycle periods (e.g. the milling cutter in the manufacturing period, the adjustable wrench in the assembling period and the machine operation platform in the operation period in the example), and can help a designer derive detailed design knowledge from the captured part affordances in a systematic manner. Therefore, our approach can help designers externalize and capture the part affordance-related design knowledge (esp. the contextual knowledge), which otherwise had to remain in designers’ memories as tacit knowledge.

It should be pointed out that our design knowledge capturing approach will impose some additional work on a design organization. In the above fixture design example, it took the expert designer about twenty hours (in several days) to complete the design knowledge capturing work. For complex artifacts, the design knowledge capturing process will need many more designers and will take much more time. However, such additional work can also save much for a design organization in the future, if the captured design knowledge can be reused. In a design test carried out in the aforementioned mechanical fixture design group, a novice designer fulfilled a routine design task with the aid of our DKR platform within 3 h, which often took at least one week before. Therefore, the proposed design knowledge capturing approach is very suitable for routine or adaptive design tasks, where there is much reusable design knowledge.

In addition, it should be admitted that the approach reported here is still in the theoretical development stage. Since the aforementioned mechanical fixture design is somewhat simple, the above case is still insufficient to demonstrate that the proposed approach is good enough to fit the designing of complex artifacts (e.g. automobiles, aircrafts, etc.). When the approach is applied to the design of complex artifacts, a possible issue might be how to define the EACs for such artifacts, since it could deal with much more complex situations. Another possible issue could be that a designer has carried out a detailed design task in an experience-driven manner (rather than in a systematic manner), which would then make it difficult for him/her to systemize such experience for capture. In addition, the complexity of such artifacts’ design processes might also worsen the design knowledge capturing issue, since many design parameters might be dependent on each other.

Finally, it should be pointed out that this paper primarily deals with how to capture tacit design knowledge in a detailed design. As to the approach for modeling the captured design knowledge, interested readers can find it in our recent work [5]. Due to limited space, this paper also does not deal with how to reuse the detailed design knowledge that has been captured, which is a major topic that will be elaborated in the future.

8. Conclusions

DKR (i.e. Design Knowledge Reuse) is a significant strategy for manufacturing enterprises to develop robust artifacts with less effort. It has attracted considerable interests from the engineering design community in the past two decades. However, there has been no effective approach for helping designers capture detailed design knowledge, which often involves various lifecycle factors that existing transformation-based design theories (e.g. [9]) usually cannot fit well. As a result, much detailed design knowledge still has to remain in designers’ memories as tacit knowledge, which, therefore, can largely hinder the successful implementation of the DKR strategy in detailed design.

To address this issue, this paper proposes a part affordance-based approach for externalizing and capturing detailed design knowledge. It introduces a model for representing the detailed design-related information (e.g. form and behavior) of a part. Based on the relational theory for design, the concept, part affordance, is then employed to help designers externalize various lifecycle factors that are implicit but should be considered in a detailed design. Based on the affordance constraint axiom, a systematic approach is then proposed for capturing detailed design knowledge from part affordances through the analysis of extreme working situations (i.e. Extreme Affordance Combinations). The proposed approach has been implemented as the Design Knowledge-Capturing System (DKCS). A fixture design case has been employed to illustrate how detailed design knowledge can be captured with DKCS. Since our approach allows designers to capture both the contextual knowledge and the constraint knowledge about a detailed design, it will be helpful for related designers to understand and then to reuse the design knowledge hiding behind a detailed design.

The primary contributions of this research can be summarized from the following two perspectives. One is the knowledge-based engineering design research. Existing knowledge-based design approaches primarily deal with the knowledge that is explicit in conceptual or early design stages, while our research attempts to provide a systematic approach for capturing and modeling tacit knowledge hiding behind a detailed design. The other is the constraint management research. Compared with the existing constraint management approaches focused on how to model and manage design constraints, our design knowledge-capturing approach has two salient features. On one hand, based on the concept of part affordances, it can help designers capture and model the contextual knowledge about various lifecycle factors that have been implicitly considered in a detailed design. On the other hand, based on the affordance constraint axiom and the analysis of extreme working situations, it can tell designers how various detailed design constraints are derived from the captured part affordances, which can show how various lifecycle factors are considered in a detailed design.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive suggestions. This research is supported by Science and Technology Commission of Shanghai Municipality (Grant No. 09QA1402800), Natural Science Foundation of China (Grant No. 51375301) and China CSR Corporation.
References


