DIO: A pattern for capturing the intents underlying designs

Monika Solanki

Department of Computer Science,
University of Oxford, UK
monika.solanki@cs.ox.ac.uk

Abstract. A critical and often overlooked aspect underlying the design and consequent realisation of an artifact is its *design intent*. Given the highly distributed and diverse nature of workflows in today's design environments, the need to have a shared understanding of the design intent, to enable effective communication and coordination between the development teams is crucial. In this paper we present DIO: a generic content ontology design pattern that provides the much required conceptualisation needed to capture the knowledge generated during design phases. We further show, how DIO can be specialised to generate a pattern - SDI (Software Design Intent) that enables the capturing of knowledge during Software design phases.

1 Introduction

The process of systematically designing an artifact is a time-consuming and laborious task. Typically, several iterations, deliberations and informal discussions are undertaken before a final agreement can be reached, on the features and attributes that need to be included in the concrete realisation of the artifact.

This has huge repercussions as design reuse becomes confined to the scope of the agents involved in the core design process. The arguments raised against specific potential solutions and the justifications supporting several rejected solutions cannot be exploited to inform future designs. The knowledge engineering that encompasses the capturing of the "Design Intent" or the "Design rationale" is therefore now a core requirement for most design environments. While several efforts in the past have focused on representing design intents, many of them are informal specification, developed for specific domains such as product design [2] or software models [1]. Further, in Software design, for the few scenarios where the intent is recorded, it is simply expressed as an informal comment, sometimes voluminous, verbose and on most occasions, out-of-sync with the finally approved design. This is clearly not enough from atleast two perspectives: (1) Informal comments do not allow us to take advantage of logical theories for detecting inconsistencies and errors in the design deliberations. (2) There are no implementation details available during the design phase. The intent is the only authoritative and unambiguous source of information that the development teams can exploit.

In this paper we present a generic content ontology design pattern, DIO (Design Intent Ontology)[1] for formally harnessing the intents or the rationales that emerge or are generated during the processes that underlie modern design decision phases. DIO is a domain agnostic pattern and most domain specific design rationale models will specialise from DIO. It provides a high level conceptualisation of the typical entities that designers encounter during the phases of designing artifacts. A significant advantage of DIO over existing design intent frameworks is its direct mapping to PROV[2].

2 Pattern description and Graphical representation

2.1 Intent

DIO provides a minimalistic abstraction and defines conceptual, generic entities for the modelling of semantically enriched knowledge required to capture the intents or rationale behind the design of an artifact. The pattern can be specialised to define domain specific design intents.

2.2 Competency questions

Given the requirements above, we define competency questions that motivate the design of a pattern-oriented conceptualisation for design intents.

- What are the solution choices for a design problem?
- Which functional requirements are being fulfilled by the accepted design solution?
- What are the alternative solutions to an accepted solution?
- What are the arguments against a proposed solution?
- What are the justifications for a proposed solution?
- Which agent has corroborated a solution?
- What assumptions have been made in arriving at a specific solution?
- What is the status of a design issue?

2.3 Conceptual Entities

Some of the key concepts and relationships encapsulating the data model defined by the pattern are described below. Note that several entities in DIO extend from or exploit concepts and relationships defined in PROV-O as further illustrated in Figure 1.

- `Design`: An entity representing the specification of an object, manifested by one or more agents, intended to accomplish goals, in a particular environment, using a set of components, satisfying a set of requirements, subject to constraints.

[1] `http://www.essepuntato.it/lode/owlapi/http://purl.org/dio/`
[2] `http://w3.org/ns/prov#`

- `DesignIntent`: An entity representing the notion of a design intent, i.e., the rationale underpinning the choices that are made from the alternatives available during various phases of the overall design lifecycle.
- `DesignIssue`, `DesignProblem`, `DesignGoal`, `DesignQuestion`: An entity representing the problem, goal, question or issue the design intent aims to address.
- `MandatedSolution`: An entity representing the solution accepted as a result of the design deliberation process.
- `Argument`: An entity representing the argument presented against a potential solution.
- `Justification`: An entity representing the justification presented in support of a potential solution.
- `fulfillsRequirements`: a relationship that identifies the design requirement being fulfilled by the design.
- `hasStatus`: a relationship that identifies the status of the design issue. Typically values are, `active`, `terminated`, `onHold` and `resolved`.
- `usesAssumption`: a relationship that identifies the assumptions that form the basis of a solution.

2.4 Graphical representation

Figure 1 illustrates the graphical representation of DIO. It depicts the entities defined for the pattern and their relationships with entities from PROV-O.

3 The DIO Axiomatisation

- One of the most significant entities in DIO is the `DesignIntentArtifact`. A design intent artifact is an argument, justification, evidence, assumption or heuristic among others which is generated as part of the design decision process. It is required to have certain mandatory properties and relationships as part of its definition. It must include a description, a version number and the time at which it was generated.

```
Class: DesignIntentArtifact
       SubClassOf:
        (description some xsd:string)
         and (version some xsd:string)
         and (prov:generatedAtTime some xsd:dateTime)
Class: dio:Argument
       SubClassOf:
        dio:DesignIntentArtifact
```

- A design intent artifact is related to the agent, `prov:Agent`, that created or generated it using `prov:wasAttributedTo`.

```
Class: DesignIntentArtifact
       SubClassOf:
        prov:wasAttributedTo some prov:Agent
```

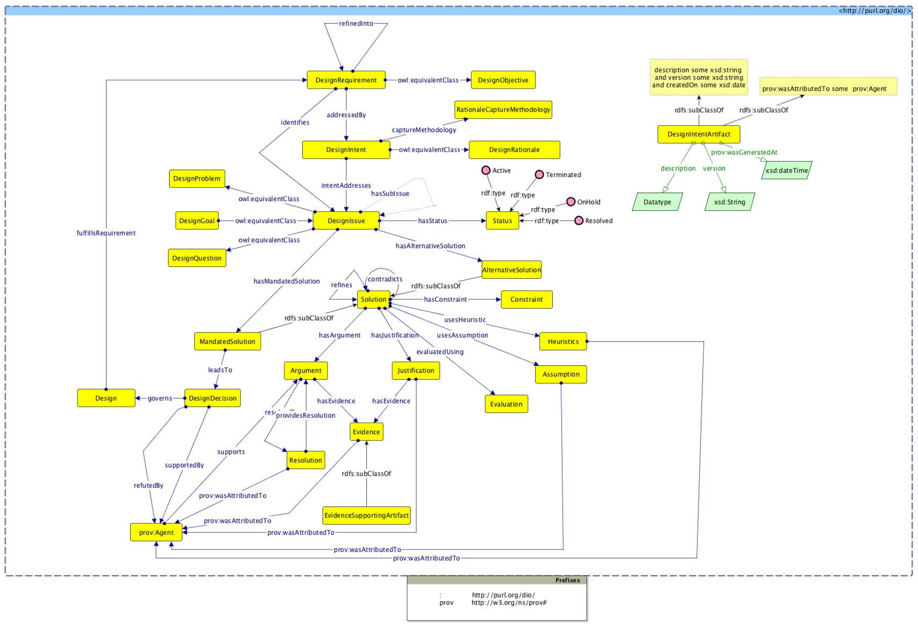

Fig. 1: Graphical Representation of DIO

– A design intent artifact is also attributed to the `DesignIntent` against which it was created.

```
Class: DesignIntentArtifact
       SubClassOf:
        prov:wasAttributedTo some dio:DesignIntent
```

4 Specialising DIO

The Software Design Intent (SDI) ontology is a minimalistic specialisation of DIO to capture the design intent during the Software design phases. As illustrated in Figure 2, SDI specialises DIO by extending the class `Design`. It associates the activity of software development, `SoftwareDevelopment`, with the design by using `prov:used`. Finally the software that is produced is related to the software development activity using `prov:wasGeneratedBy`.

5 Conclusions

The need for the representation of design intents, that originate to solve a typical design problem or a design issue, in order to fulfill a design requirement is universal across all domains. It is extremely critical to curate this knowledge

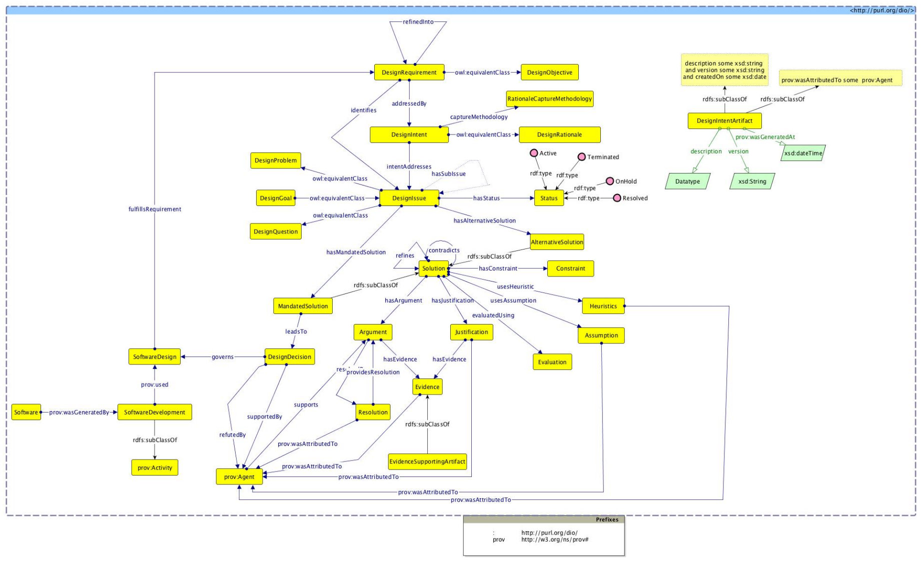

Fig. 2: Graphical Representation of SDI

for posterity and inform future generations of designs. In this paper, we have proposed a generic content ontology design pattern, DIO, that can be applied in a domain agnostic way in various design decision phases. We have shown how a minimalistic extension of DIO for the domain of Software Engineering can be utilised to associate software design rationales with the accepted software design. The work is currently in its initial stages and much still needs to be done. In future we aim to apply the pattern, for querying and deriving inferences over the knowledge curated from several design projects which are currently in various phases of making design decisions.

Acknowledgments

The research presented in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644055.

References

1. A. P. de Medeiros, D. Schwabe, and B. Feijó. Kuaba ontology: Design rationale representation and reuse in model-based designs. In *Proceedings of the 24th International Conference on Conceptual Modeling*, ER'05, pages 241–255, Berlin, Heidelberg, 2005. Springer-Verlag.
2. Y. Zhang, X. Luo, J. Li, and J. J. Buis. A semantic representation model for design rationale of products. *Adv. Eng. Inform.*, 27(1):13–26, Jan. 2013.