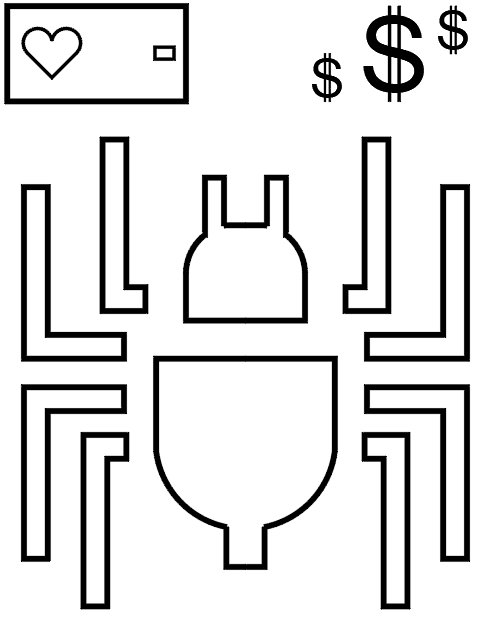
**Loyalty Crawler**

**Software Design Descriptions**

**Version 1.0**



UTD Student Group:

Anthony Spencer (Group Leader)  
Alexander Baselice  
Alexander Lundin  
Jairo Galarza  
Joseph Samonte  
Tony Nhan

RECORD OF CHANGES

\*A - ADDED M - MODIFIED D – DELETED

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VERSION NUMBER | DATE | NUMBER OF FIGURE, TABLE OR PARAGRAPH | **A\* M D** | TITLE OR BRIEF DESCRIPTION | CHANGE REQUEST NUMBER |
| 1.0 | 03/22/2019 |  |  | Fourth deliverable |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**TABLE OF CONTENTS**

**Section Page**

[**ABSTRACT**](#_4qtpzccd7j9d) **6**

[**SECTION 1. OVERVIEW**](#_ksk2dqcvgda) **7**

[1.1 Scope](#_lm30927p9bui) 7

[1.2 Purpose](#_mox6dn4z2pb3) 7

[1.3 Intended Users](#_joj6wkqcu9vq) 8

[1.4 Conformance](#_dtnlktczq8ln) 8

[1.5 Overview](#_ssfihx9st2pj) 8

[**SECTION 2. REFERENCES**](#_y1g4nvyybxv2) **9**

[**SECTION 3. DEFINITIONS**](#_lv3g1mrwfyyv) **10**

[**SECTION 4. CONCEPTUAL FRAMEWORK FOR SOFTWARE DESIGN DESCRIPTIONS**](#_1ajy84cmbmwr) **12**

[4.1 SOFTWARE DESIGN IN CONTEXT](#_ivtpb5yn579b) 12

[4.2 SOFTWARE DESIGN DESCRIPTIONS WITHIN THE LIFE CYCLE](#_6rw74imscpzy) 14

[4.2.1 Influences on SDD Preparation](#_th5fkwqll6lo) 14

[4.2.2 SDD Influences on Software Life Cycle Products](#_4eomipm0fqub) 14

[4.2.3 Design Verification and Design Role in Validation](#_3rrrq3luh9fs) 15

[**SECTION 5. DESIGN DESCRIPTION INFORMATION CONTENT**](#_by42auf9ixfm) **16**

[5.1 Introduction](#_a9u0u3bva9lp) 16

[5.2 SDD Identification](#_a5d1g867sjhm) 16

[5.3 DESIGN STAKEHOLDERS AND CONCERNS](#_6zaps5u5me1u) 17

[5.4 DESIGN VIEWS](#_gff2vw1pwae1) 17

[5.5 DESIGN VIEWPOINTS](#_8hwnzeh9y8w6) 18

[5.6 DESIGN ELEMENTS](#_hx36du870osm) 19

[5.6.1 Design Entities](#_u8belzu80pt6) 19

[5.6.2 Design Attributes](#_yz787rsldjt4) 19

[5.6.3 Design Relationships](#_70s0tsyepkqd) 20

[5.6.4 Design Constraints](#_fnwko51gxu3g) 20

[5.7 DESIGN OVERLAYS](#_3h3sgvjyytcc) 20

[5.8 DESIGN RATIONALE](#_2i5af8tp60v1) 21

[5.9 DESIGN LANGUAGES](#_9fnhu5j8yigy) 22

[**SECTION 6. DESIGN VIEWPOINTS**](#_rhmu01cqyt1v) **24**

[6.1 INTRODUCTION](#_y93180ua4pq6) 24

[6.2 COMPOSITION VIEWPOINT](#_lq3p39eiatrj) 26

[6.3 LOGICAL VIEWPOINT // STATIC MODEL](#_ntr61179l4y4) 26

[6.4 DEPENDENCY VIEWPOINT](#_t7mt2qldouwz) 27

[6.5 INFORMATION VIEWPOINT](#_u8019q83l56u) 27

[6.7 INTERFACE VIEWPOINT](#_ix6qdmdh402c) 28

[6.8 STRUCTURE VIEWPOINT](#_x3cnawln8cmk) 28

[6.9 STATE DYNAMICS VIEWPOINT // DYNAMIC MODEL](#_wszr41hptoce) 29

[6.10 ALGORITHM VIEWPOINT](#_h54hsn2r3tc7) 29

[6.11 RESOURCE VIEWPOINT](#_s9zpuluwk9mu) 30

[6.12 GUI VIEWPOINT](#_9wmzd9ja1xc2) 30

[6.13 DATABASE VIEWPOINT](#_3pr5dcfxn9q7) 30

**LIST OF FIGURES**

**Figure Page**

Figure 3-1. MVC pattern control flow [1](#_meukdy)0

Figure 5-1. Application-Usage-Viewpoint-1.0 [1](#_meukdy)7

Figure 5-2. Package-Diagram-1.0 [1](#_meukdy)8

Figure 5-3. Architecture-1.0 19

Figure 5-4. Archive-View-1.0 20

Figure 5-5. File-View-1.0 21

Figure 5-6. Session-View-1.0 22

Figure 5-7. Device-Diagram-1.0 23

Figure 6-1. Hardware-View-1.0 26

**LIST OF TABLES**

**Table Page**

[Table 4-1. Architectural Activities 1](#_meukdy)2

Table 5-1. Business Process Viewpoint14

Table 5-2. Product Viewpoint15

Table 5-3. Application Behavior Viewpoint15

Table 5-4. Application Cooperation Viewpoint16

Table 5-5. Application Structure Viewpoint16

Table 5-6. Application Usage Viewpoint17

# ABSTRACT

***Deliverable Guidance from Professor***

***ABSTRACT  
 // brief summary of entire document***

This document will describe

# SECTION 1. OVERVIEW

## 1.1 Scope

***Guidance from IEEE-P1016-d50***

*This is a standard for software design descriptions (SDD). An SDD is a representation of a software design to be used for recording design information and communicating that design information to key design stakeholders.*

*This standard is intended for use in design situations in which an explicit software design description is to be prepared. These situations include traditional software construction activities, when design leads to code, and “reverse engineering” situations where a design description is to be recovered from an existing implementation.*

*The standard can be applied to commercial, scientific, or military software that runs on digital computers. Applicability is not restricted by the size, complexity, or criticality of the software. This standard can be applied to the description of high-level and detailed designs. This standard is not limited to use with specific methodologies for design, configuration management, or quality assurance. This standard does not require the use of any particular design languages, but establishes requirements on the selection of design languages for use in an SDD. The standard can be applied to the preparation of SDDs captured as paper documents, automated databases, software development tools or other media.*

*NOTE—The requirements in P1016 are intended to be consistent*

## 1.2 Purpose

***Guidance from IEEE-P1016-d50***

*This standard specifies requirements on the information content and metadata organization of SDDs. This standard specifies requirements for the selection of design languages to be used for software design description, and requirements for documenting design viewpoints to be used in organizing a software design description.*

## 1.3 Intended Users

***Guidance from IEEE-P1016-d50***

*This standard is intended for technical and managerial stakeholders who prepare and use SDDs. It will guide a designer in the selection, organization, and presentation of design information. For an organization developing its own design description practices, the use of this standard will help to ensure that design descriptions are complete, concise, consistent, interchangeable, appropriate for recording design experiences and lessons learned, well organized and easy to communicate.*

## 1.4 Conformance

***Guidance from IEEE-P1016-d50***

*A Software Design Description conforms to this standard if it satisfies all of the requirements of this standard. Requirements are denoted by the verb shall.*

## 1.5 Overview

***Guidance from IEEE-P1016-d50***

# 

# SECTION 2. REFERENCES

**List of References**

[1] (2019). Available at: http://www.san.uri.br/~pbetencourt/engsoftII/IEEE-P1016-d50.PDF (Accessed: 5 March 2019).

[2] Faroque, O. (2019) nanofaroque/Software\_System\_Design\_Docs, GitHub. Available at: https://github.com/nanofaroque/Software\_System\_Design\_Docs (Accessed: 7 March 2019).

**Reference Descriptions**

[1] This is the PDF standard for IEEE Draft Standard for Software Design Descriptions

[2] This github link contains an example for a completed Software Design Description from another group, who also completely adhered to the IEEE standard

# 

# SECTION 3. DEFINITIONS

1. model-view-controller (MVC)

*This term refers to the architectural pattern that separates three core features from each other. Model to represent data. View for the user to interact with and the controller to facilitate data transfer between the view and the model.*

1. artifacts

*The term artifacts refers to any and all documents retrieved by the web crawler, as well as the documents stored in the crawler cache and artifact file reserve. This includes, but is not limited to, HTML files, image files (.png, .jpeg, etc.), and text files (.txt, .pdf, .docx, etc.).*

1. search session

*The term refers to an instance of activating the web crawler for a duration of time, and the resulting artifacts for which it finds.*

1. crawler cache

*This term refers to the architectural component (GUI & database structures) that shall store artifacts found during a single search session.*

1. artifact file reserve

*This term refers to the architectural component (GUI & database structures) that the user shall use to organize and store artifacts deemed relevant from the crawler cache.*

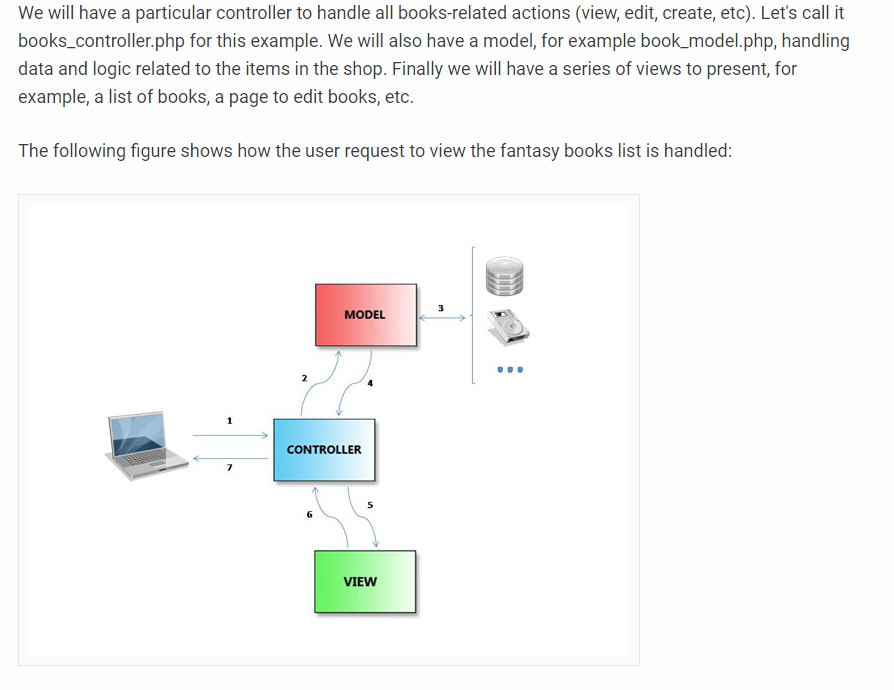
1. validation

*process of checking if specification meets customers needs*

1. verification

*process of checking if software implementation meets specification*

Figure 3-1. MVC pattern control flow [2].



# 

# SECTION 4. CONCEPTUAL FRAMEWORK FOR SOFTWARE DESIGN DESCRIPTIONS

***Guidance from IEEE-P1016-d50***

*This clause establishes a conceptual framework for Software Design Descriptions. The conceptual framework includes basic terms and concepts of software design description, the context in which SDDs are prepared and used, the stakeholders who use them, and how they are used.*

## 4.1 SOFTWARE DESIGN IN CONTEXT

***Guidance from IEEE-P1016-d50***

*A design is a framework which demonstrates a means to fulfill the requirements for some software item and to guide the implementation of that software item. A design subject is any software item to be constructed or which already exists and is to be analyzed, without loss of generality we will also refer to a design subject as the system under design or software under design. This standard does not establish what a design subject may be. Examples of design subjects include systems, subsystems, applications, components, libraries, application frameworks, application program interfaces (APIs) and design pattern catalogs.*

*A software design description (SDD) is a representation of some design subject of interest. An SDD is prepared to represent exactly one design subject. An SDD can be produced to capture one or more levels or layers of concern with respect to its design subject. These levels or layers are usually defined by the design methods in use or the life cycle context; they have names such as architectural design, logical design, or physical design. An SDD can be prepared and used in a variety of design situations. Typically, an SDD is prepared to support the development of a software item to solve a problem, where this problem has been expressed in terms of a set of requirements. The contents of the SDD can then be traced to these requirements. In other cases, an SDD can be prepared to understand an existing system lacking any design documentation. Typically, there is a system under development or under review for which an SDD is to be described such that information of interest is to be captured, organized, presented and disseminated to all interested parties. This information of interest can be used for planning, analysis, implementation and evolution of the software system, by identifying and addressing essential design concerns. A design concern is any area of interest in the design, pertaining to its development, implementation, or operation. Design concerns are expressed by design stakeholders—those parties which may be individuals, groups and organizations with an interest in the design of the system. Frequently design concerns arise from specific requirements on the software, others arise from contextual constraints. Typical design concerns include functionality, reliability, performance, and maintainability. Typical design stakeholders include users, developers, software designers, system integrators, maintainers, acquirers, and project managers.*

*An SDD is organized using design views. A design view addresses one or more of the design concerns.*

*Each design view is governed by a design viewpoint. Each design viewpoint focuses on a set of the design concerns and introduces a set of descriptive resources (or view elements) that are used to construct and interpret the design view. E.g., a viewpoint may introduce familiar elements such as functions, input and outputs; these elements are used to construct a functional view.*

*There are four kinds of view elements: design entities, design relationships among entities, design attributes and design constraints on those elements. A design viewpoint will define the view element types to be used in any design views it governs. Each design view used to represent a software system is expressed as a collection of instances of design entities, attributes, the relationships among design entities and constraints on those elements. The design information needs of stakeholders of the system under design are to be satisfied through use of these elements.*

*To facilitate automation, exchange and long-term relevance of SDDs, the design state and design rationale information is accompanied by metadata describing both design state and design rationale. Metadata are organized around viewpoints and design state around design views to include instances for design entities, attributes and relationships. Figure 2 depicts the state transitions of an SDD in this respect.*

*Most importantly, this standard assumes use of models in software design. Models and their representations can be used in different modes: as sketches or rough drafts; as blueprints suitable for implementation; and as executable specifications. The use of models as sketches, while highly recommended in practice, is not governed by this standard; the intended modes are blueprints and executable specifications, as formal engineering documents. The primary use of a sketch is as an aid for thinking, and in conversation about ideas before those ideas can be systematized into designs as either blueprints or executable specifications. Blueprints are developed under general requirements of consistency and reasonable completeness and intended to communicate designs to humans such as to implementers or to maintainers trying to understand the design in order to change it. Executable specifications further restrict descriptions to those that can be automatically translated into implementations on real machines, without the intervention of human intelligence. Software design descriptions covered by this standard are not only formalized using defined languages, but are also intended to be precise i.e., rigorous irrespective of the medium to be used to record them. It is the intent to communicate specific ideas only and not to present complete designs to be implemented as is, that distinguishes sketches from blueprints and executable specifications.*

*There is no restriction by this standard to the use of any design language in sketches or to the use of sketches in documentation, but the expectation is to use the P1016 standard to govern blueprints and/or executable specifications in the full scope of design responsibility. In the anticipated lifetime of this standard (2005 to 2010), design automation is far more feasible and likely than in paper-based designs, but that expectation by no means eliminates the convenience of sketches (including paper, whiteboards and PCtablets to capture them) as designers are humans and the purpose of design information is human communication, particularly for the purpose of critiquing designs.*

## 4.2 SOFTWARE DESIGN DESCRIPTIONS WITHIN THE LIFE CYCLE

***Guidance from IEEE-P1016-d50***

*In this standard, a typical cycle will be used to describe the various design situations in which an SDD can be created and used. This life cycle is based on IEEE/EIA 12207.*

### 

### 4.2.1 Influences on SDD Preparation

***Guidance from IEEE-P1016-d50***

*The key software life cycle product that drives a software design is typically the software requirements description (SRD). An SRD captures the software requirements which will drive the design, and may contain design constraints that must be considered or observed.*

### 4.2.2 SDD Influences on Software Life Cycle Products

## 

***Guidance from IEEE-P1016-d50***

*The SDD influences the content of several major software life cycle products. Developers of these products will be recognized among the SDD’s intended audience.*

*− Software Requirements Description. Design decisions, or design constraints discovered during the preparation of the SDD, may lead to requirements changes. Often traceability between requirements and design is maintained to manage these changes.*

*− Test Documentation. Test planning can be influenced by the SDD, but any white-box testing activities at the level of unit, integration, and system testing, are directly influenced by the SDD. Developers of any test specifications and test cases that relate to this type of testing should cover the design functionality, relationships, objects, and data descriptions contained in the SDD.*

### 4.2.3 Design Verification and Design Role in Validation

***Guidance from IEEE-P1016-d50***

*Verification is a process for determining whether the software products of an activity fulfill the requirements or conditions imposed on them in the previous activities. [IEEE/EIA 12207.0] An SDD can be subject to design verification to ascertain whether the design: is consistent with stated requirements; implements intended design decisions (such as those pertaining to interfaces, inputs, outputs, algorithms, resource allocation, and error handling); achieves intended qualities (such as safety, security, maintainability); and conforms to an imposed architecture. Verification therefore raises a set of design concerns which can be dealt with in the SDD and subjected to inspection or analysis.*

*Validation is a process for determining whether the requirements and the final, as-built system or software product fulfills its specific intended use. [IEEE/EIA 12207.0] The SDD can play a role in this process mainly by providing: an overview necessary for understanding the implementation; the rationale justifying design decisions made; and traceability back to the requirements on the software item.*

Table 4-1. Architectural Activities.

# 

# SECTION 5. DESIGN DESCRIPTION INFORMATION CONTENT

## 5.1 Introduction

***Guidance from IEEE-P1016-d50***

*The required elements of an SDD are:*

*− an identification of the SDD,*

*− its identified stakeholders,*

*− its identified design concerns,*

*− its selected design viewpoints, each with type definitions of its allowed design elements and design languages,*

*− its design views,*

*− its design overlays, and*

*− its design rationale.*

*These are described in the remainder of this clause.*

## 5.2 SDD Identification

***Guidance from IEEE-P1016-d50***

*An SDD shall include the following descriptive information:*

*− date of issue and status;*

*− scope;*

*− issuing organization;*

*− authorship (responsibility or copyright information);*

*− references;*

*− context;*

*− one or more design languages for each design viewpoint used;*

*− body;*

*− summary;*

*− glossary;*

*− change history.*

## 5.3 DESIGN STAKEHOLDERS AND CONCERNS

***Guidance from IEEE-P1016-d50***

*An SDD shall include the following descriptive information:*

*− date of issue and status;*

*− scope;*

*− issuing organization;*

*− authorship (responsibility or copyright information);*

*− references;*

*− context;*

*− one or more design languages for each design viewpoint used;*

*− body;*

*− summary;*

*− glossary;*

*− change history.*

## 5.4 DESIGN VIEWS

***Guidance from IEEE-P1016-d50***

*A software design description shall be organized into one or more design views. A design view is a representation consisting of design entities, design entity attributes, design relationships and design constraints to address an identified set of design concerns from a specific viewpoint.*

*The purpose of a design view is to address design concerns pertaining to the design subject, to allow a design stakeholder to focus on design details from a different perspective or design viewpoint, and effectively address relevant requirements.*

*Design views are the means of organizing an SDD to satisfy the needs of each design stakeholder and to promote separation of concerns. Each design view addresses one or more design concerns. Together, these views provide a comprehensive description of the design in a concise and usable form that simplifies information access and assimilation. Each software design stakeholder can have a distinct perspective on what are the essential aspects of a software design. Other design information may be extraneous to that stakeholder.*

*An SDD is complete when each identified design concern is the topic of at least one design view, all design attributes refined from each design concern by some viewpoint have been specified for all of the design entities and relationships in its associated view and all design constraints have been applied.*

*An SDD is consistent if there are no known conflicts between the elements of its design views.*

## 5.5 DESIGN VIEWPOINTS

***Guidance from IEEE-P1016-d50***

*A design viewpoint is a specification of the conventions for constructing and using a design view. It identifies the resources from which to develop individual design views. For each design view in an SDD, there shall be a design viewpoint governing it.*

*Each design viewpoint shall be specified by:*

*− the viewpoint name;*

*− the concerns which are the topics of the viewpoint;*

*− the resources, or view elements, provided by that viewpoint, specifically the types of design entities, attributes, relationships and constraints introduced by that viewpoint or used by that viewpoint (which may have been defined elsewhere). These elements may be realized by one or more design languages;*

*− analytical methods or other operations to be used in constructing the view based upon the viewpoint, and criteria for evaluating the design based upon the viewpoint; and*

*− the viewpoint source (e.g., authorship or citation) when applicable.*

*In addition, a design viewpoint specification may provide the following information on using the viewpoint:*

*− formal or informal consistency and completeness tests to be applied to the models making up an associated view;*

*− evaluation or analysis techniques to be applied to the models; and*

*− heuristics, patterns, or other guidelines to assist in construction or synthesis of an associated view.*

*An SDD shall include a rationale for the selection of each selected viewpoint.*

*Each design concern identified in an SDD shall be addressed by at least one viewpoint selected for use. A design concern may be the focus of more than one viewpoint in an SDD.*

*NOTE—A design viewpoint specification may be included in the SDD or incorporated by reference.*

*NOTE—It is envisioned that through the selection of suitable viewpoints an SDD can achieve conformance to other development standards.*

## 5.6 DESIGN ELEMENTS

***Guidance from IEEE-P1016-d50***

*A design element (or model element) is any item occurring in a design view. A design element may be any of the following: design entity, design relationship, design attribute, or design constraint.*

*Each design element shall have the following attributes: a name, a type and an expression.*

*The type of each design element shall be introduced within exactly one design viewpoint definition.*

*A design element may be referenced in one or more design views.*

*NOTE—A design element is introduced and “owned” by exactly one design view; conforming to its type definition within the associated viewpoint definition. It may be shared or referenced within other design views. Sharing of design elements permits the expression of design aspects. [Aspect-Oriented Programming]*

### 

### 5.6.1 Design Entities

### 5.6.2 Design Attributes

### 5.6.3 Design Relationships

### 5.6.4 Design Constraints

***Guidance from IEEE-P1016-d50***

*This is not schedule constraints. Do not mention anything about schedule.*

***Guidance from IEEE-P1016-d50***

*The key software life cycle product that drives a software design is typically the software requirements description (SRD). An SRD captures the software requirements which will drive the design, and may contain design constraints that must be considered or observed.*

## 

## 

## 5.7 DESIGN OVERLAYS

***Guidance from IEEE-P1016-d50***

*A design overlay is a mechanism for presenting additional, detailed or derived information with respect to an already-defined design view. It is frequently convenient to capture such information, as an alternative to introducing a new viewpoint, using overlays upon a subset of the information in the diagrams selected in existing relevant viewpoints.*

*Each design overlay shall be clearly marked.*

*Each design overlay shall be clearly associated with a single viewpoint.*

*NOTE—Reasons to utilize a design overlay as a part of an SDD include: to provide an extension mechanism for design information to be presented conveniently on top of some view without a requirement for existing external standardization of languages and notations for such representation; to extend expressive power of representation with additional details while reusing information from existing views (i.e. without a need to define additional views or persistently store derivable design information); and to relate design information*

## 

## 

## 5.8 DESIGN RATIONALE

***Deliverable Guidance from Professor***

***RATIONALE FOR YOUR DETAILED DESIGN MODEL***

***TRACEABILITY FROM REQUIREMENTS TO DETAILED DESIGN MODEL***

***Guidance from IEEE-P1016-d50***

*Design rationale is information capturing the reasoning of the designer which led to the system as designed, including design options, tradeoffs considered, decisions made, and the justifications of those decisions.*

*Design rationale takes the form of commentary, made throughout the decision process and associated with collections of design elements. It captures the reasoning that led to the system as it has been designed. Design rationale includes: design issues raised and addressed in response to design concerns; design options considered; tradeoffs evaluated; decisions made; criteria used to guide design decisions; and arguments and justifications made to reach decisions.*

***Guidance from IEEE-P1016-d50***

*Design rationale is information capturing the reasoning of the designer which led to the system as designed, including design options, tradeoffs considered, decisions made, and the justifications of those decisions.*

*Design rationale takes the form of commentary, made throughout the decision process and associated with collections of design elements. It captures the reasoning that led to the system as it has been designed. Design rationale includes: design issues raised and addressed in response to design concerns; design options considered; tradeoffs evaluated; decisions made; criteria used to guide design decisions; and arguments and justifications made to reach decisions.*

## 5.9 DESIGN LANGUAGES

***Guidance from IEEE-P1016-d50***

*A design language is a notation, representational scheme or other modeling technique used to develop, analyze, and document a software design. There are many design languages used to describe software designs. Design languages are selected as a part of design viewpoint declaration (5.5).*

*A design language may be selected for a design viewpoint only if it supports all modeling elements defined by that viewpoint.*

*For use in SDDs, design languages shall be selected which have:*

*− a well-defined syntax and semantics; and*

*− the status of an available standard or equivalent defining document.*

*In a conforming SDD, only standardized and established\* (defined and convenient) design languages shall be used. In the case of a newly-invented design language, the language definition must be provided as a part of the viewpoint declaration.*

*NOTE—Standardized design languages in common use are preferable to established one without a formal definition. Examples of standardized languages include: IDEF0, IDEF1X, Conceptual Graphs, UML, VDL, and Z. Examples of established languages include: Petri Nets, State Machines, Automata, Decision Tables, Warnier Diagrams, JSP, PDL, Structure Charts, HIPO, JSD, Reliability Models, Queueing Models.*

*NOTE—It is acceptable to use a design language in more than one view. It is also acceptable to use more than one design language within any of the views as long as they were declared by the viewpoint. Even for the portion of the design as long as one is used as a basis for interchange whenever that is necessary due to organizational consideration such as development by non-collocated team members, subcontracting of a partial design responsibility, taking advantages of several case tools and/or key designer’s expertise, etc.*

*NOTE—Annex B establishes a uniform format for describing design languages to be used in SDDs.*

*NOTE—In case that no adequate design language is readily available for a specified viewpoint, it is the designer’s responsibility to provide an adaptation of an existing language or the definition of an appropriate new design language. In exceptional cases, the definition provided by the designer shall be included in the SDD after proper evaluation for the reference before any use of such language is to be approved for capture of a view information in an SDD. This does not restrict exploratory use in sketching designs but excludes any use in formal documents, blueprints and executable specifications without prior update of SDD metadata with adequate design language. Informal documents without notification of nonstandard language or nonstandard use of a design language in them, can not be included in a conformant SDD. This note clarifies dynamic nature of leading edge design technology and points to a process necessary in adapting SDD metadata before populating or communicating SDD (design state) view information. Explicit status of not-yet-approved SDD information is necessary if some exploratory methodological information is to be included for review purposes and archived with rationale for decisions made.*

# 

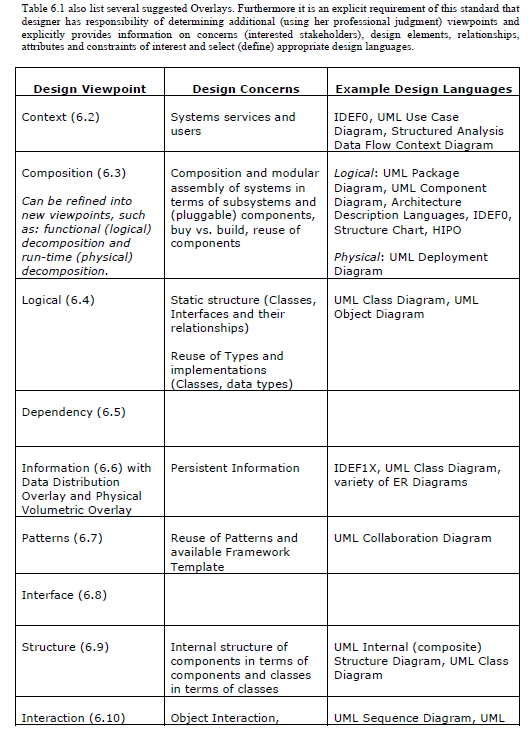
# SECTION 6. DESIGN VIEWPOINTS

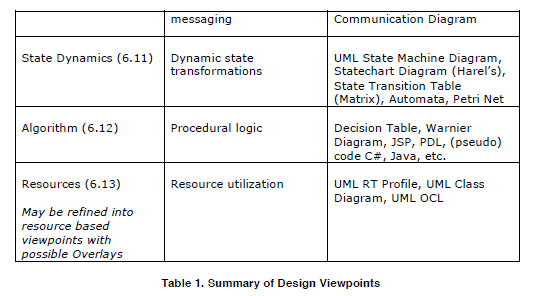
## 6.1 INTRODUCTION

***Guidance from IEEE-P1016-d50***

*This clause defines several design viewpoints for use in SDDs. It illustrates the specification of viewpoints in terms of design language selections, relates design concerns with viewpoints and establishes language- (notation-, method-, and process-) neutral names for selected viewpoints. Table 1 summarizes these viewpoints. For each viewpoint, its name, design concerns, and appropriate design languages, are listed. Short descriptions relating a minimal set of design entities, design relationships, design entity attributes, and design constraints are provided for each viewpoint. Additional references pertinent to the use of each viewpoint are also listed. These viewpoints are required with a caveat, a qualified designer judgment is necessary to tailor out viewpoints not of interest in a particular situation, or to refine viewpoints (see for example viewpoint 2).*

Seet summary table below, printed from reference 1.





## 6.2 COMPOSITION VIEWPOINT

\* write description of this only no drawing -- notes from 03/05/2019--virtual\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.3 LOGICAL VIEWPOINT // STATIC MODEL

***Deliverable Guidance from Professor***

***Static Model***

***Class Diagrams***

***// captured in Rose (other tools are also allowed)***

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.4 DEPENDENCY VIEWPOINT

Notes from meeting

Specify 1 to 1 and 1 to many

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.5 INFORMATION VIEWPOINT

\*could be a description if you want, or a picture if you want -- notes from 03/05/2019--virtual\*\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

**6.6 PATTERNS USE VIEWPOINT**

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.7 INTERFACE VIEWPOINT

\* Viewpoint based on tasks for each entity -- notes from 03/05/2019--virtual\*\*

\*Example: what tasks can the web crawler do? -- notes from 03/05/2019--virtual\*\*

\*Parse, search -- notes from 03/05/2019--virtual\*\*

\*description that ties back to methods in class diagram -- notes from 03/05/2019--virtual\*\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.8 STRUCTURE VIEWPOINT

\*description that ties back to class diagram -- notes from 03/05/2019--virtual\*\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.9 STATE DYNAMICS VIEWPOINT // DYNAMIC MODEL

***Deliverable Guidance from Professor***

***Dynamic Model***

***Sequence Diagrams***

***// captured in Rose (other tools are also allowed)***

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.10 ALGORITHM VIEWPOINT

\* describe the tool -- notes from 03/05/2019--virtual\*\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.11 RESOURCE VIEWPOINT

\*\*describe how information is gathered and used in terms of artifacts\*\*

***Guidance from IEEE-P1016-d50***

*See chart in 6.1*

*Go read this section in the IEEE document as well*

## 6.12 GUI VIEWPOINT

***Deliverable Guidance from Professor***

***GUI Design***

***// screen diagrams, drawn using tool or coded***

## 6.13 DATABASE VIEWPOINT

\*\* describe list of links, without duplicates\*\*

***Deliverable Guidance from Professor***

***GUI Design***

***// screen diagrams, drawn using tool or coded***

**APPENDICES**

# 

**DOCUMENT CHANGE REQUEST (DCR)**

|  |  |
| --- | --- |
| Document Title: **Software Requirements Specification** | Tracking Number:  DCR-AD- |
| Name of Submitting Organization:  UTD Student Group | |
| Organization Contact:  amlundin88@gmail.com | Phone: |
| Mailing Address: | |
| DCR Description: | Date: |
| Change Location:  (use section #, figure #, table #, etc.) | |
| Proposed change: | |
| Rationale for Change: | |
| Note: For the ***appropriate authority*** to take appropriate action on a change request, please provide a clear description of the recommended change along with supporting rationale.  Email to:  anthonygaganovspencer@gmail.com  alexbaselice2@gmail.com  amlundin88@gmail.com  jgalarza303@gmail.com  josephisnt@gmail.com  xclearzx@gmail.com  Submit online:  ***Print this sheet and store in this Google Drive folder***  *SE Senior Project/Deliverables/Document Change Request Folder*  DCR Form 1/2009 | |