Architecture description template for Borrow My Books

Architecture Description of

Express JS, Client-server and asynchronous calls

For

Borrow My Books

"Bare bones" edition version: 2.3

Template prepared by:
Rich Hilliard
r.hilliard@computer.org

License

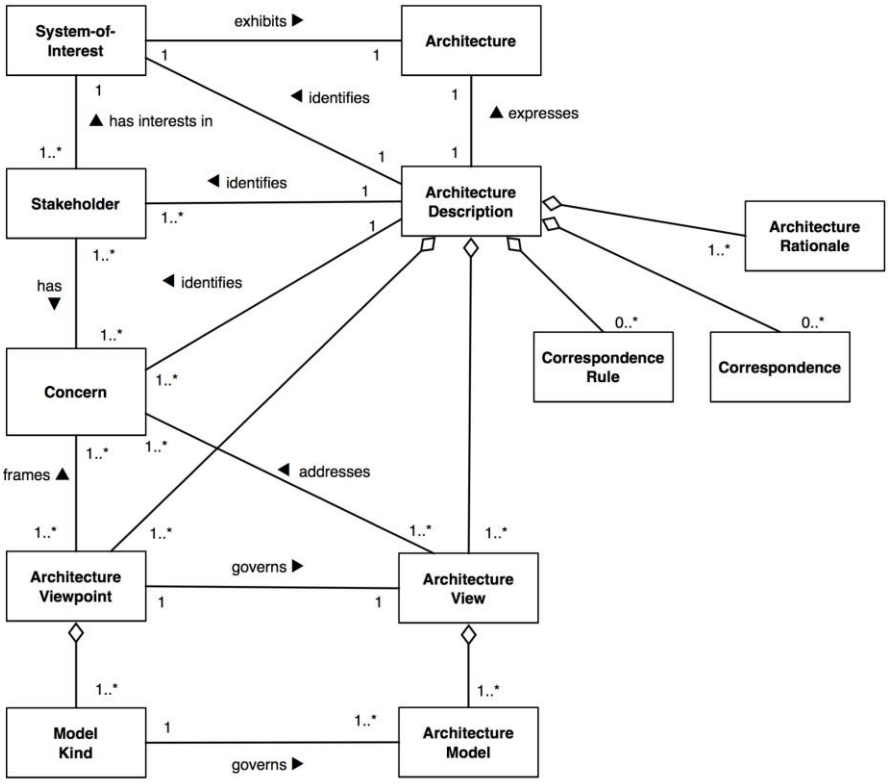

Figure 1: Content model of an architecture description

The latest version is always available at:
http://www.iso-architecture.org/42010/templates/.

Chapter 1

Introduction

This chapter describes introductory information items of the AD, including identifying and supplementary information.

1.1 Identifying information

The Software Architecture chosen for the development of the BorrowMyBooks system is a hybrid one, consisting of a Model View Controller (MVC) type of system called ExpressJS for the implementation of relationships between user interfaces and underlying data models, and a Client Server in order to manage the network aspects of the system.

1.2 Supplementary information

Massive Dynamic (MD) proposes a system which will help students trade, borrow and sell second hand books – mainly textbooks. The system's main objective is to allow users to easily search for, locate and purchase the books through it. It will have gamification aspects like Uber where users can rate other users. Additionally, it will include a reporting function in order to allow reporting of abusive users. There will be an administrative profile/s in the system, to aid for the maintenance of the system. The technology we have chosen to implement this on is a web based technology, namely the MEAN stack. The system will not cover using online payment methods and will only facilitate the locating and transferal of physical goods.

1.3 Other information

This system makes use of web 2.0 technologies. We are using JavaScript processed by the V8 engine through NodeJS. We make heavy use of the callback code pattern. The system is based on a single process model which is event driven. These event occur mainly with user input and/or asynchronous calls. Other code patterns we have used when they are necessary are both the yield/await directives and also the async.waterfall model. (Found

here: http://caolan.github.io/async/) These other patterns have been used to reduce the amount of nesting and callbacks used in the code. They are not anti-patterns but they are not the standard that comes with our use of the database connection driver (mongoose) and ExpressJS.

The live version of the site is hosted on Heroku (https://www.**heroku**.com/). The URL is https://borrowmybooks.herokuapp.com/. This site provides hosting for the server side code and public assets. The MongoDb database is stored on another site who is affiliated with Heroku called mLab (https://www.mlab.com/).

1.3.1 Architecture evaluations

? Include results from any evaluations of the <Architecture Name> being documented.

1.3.2 Rationale for key decisions

? An architecture description shall include rationale for each decision considered to be a key architecture decision (per ISO/IEC/IEEE 42010, 5.8.2).

See §A for further guidance about decisions and rationale.

Chapter 2

Stakeholders and concerns

2.1 Stakeholders

There are 4 main stakeholders which take part in the functioning of the system. These are:

Stakeholder 1: General user (Seller)

Stakeholder 2: General user (Purchaser)

Stakeholder 3: Administrator

Stakeholder 4: Owner

2.2 Concerns

The purpose of the BorrowMyBooks system is to create an easy to use platform where people can trade their textbooks. The architecture we have chosen to apply to this system is suitable due to the fact that the system will be developed as a Web application, where the separation of concerns (as supported by the MVC architecture) and the isolation of application logic from the user interface is necessary.
Furthermore, the Client-server network architecture will be suitable for the functioning of the BorrowMyBooks system as the system is mainly based on a large database of data relating to users, administrators and books listed for trading. This architecture will aid in the handling of the database and the communications of each user with it.
The feasibility of the system is dependent on the demand which students have for second hand textbooks. At tertiary education level, the demand is high due to the fact that brand new textbooks are expensive, and in some cases, textbooks are only used for 6 months, after which they become useless unless they are sold. This system will therefore make it easier for students to trade their textbook with others which is currently a cumbersome task. The deployment of the system is feasible.
There are few risk factors involved for the stakeholders of the system. The system will include password protection for the users, namely password salting, which will minimize the risk of user accounts being hacked into.
Furthermore, users will not be requested to submit financial data, such as credit card numbers and bank details. No transaction handling will be done through the system, it is solely developed for the location and transferal of textbooks between pupils.

The maintenance of the system will be carried out by the main developers, who will make sure that the system runs fluidly, any bugs are eliminated and that database communications are fast and reliable.
Lastly, the evolution of the system will depend on the future requirements of its users.

The main concerns are laid out as follows:

- CN1: All users must be able to log in to the system in a secure manner.
- CN2: An administrator must be able to log in to the system with their credentials which will grant them access to admin functionality.
- CN3: An unregistered user must be able to sign up through the system's sign up page.
- CN4: Any user on the system must be able to browse available books, their prices and descriptions.
- CN5: A seller on the system must be able to upload a book which they wish to trade and provide all relevant details for the book.
- CN6: A purchaser on the system must be able to browse all available books and view their relevant details through the explore page.
- CN7: A purchaser on the system must be able to search for a specific book on the system through the built in search engine.
- CN8: A purchaser must be able to perform a successful transaction on the system for a book which they are interested in purchasing.
- CN9: A seller must be able to view transaction requests for books which they are selling and accept or reject them.
- CN10: A user of the system must be able to report another user for bad conduct through the site's functionality.
- CN11: An administrator must be able to view and act upon any reports made relating to bad user conduct.
- CN12: An administrator must be able to view all transactions made for all books through the system and sort out any issues which may arise which are beyond the control of the purchaser and seller, and are system-related.

2.3 Concern–Stakeholder Traceability

The table below depicts the relationship between the various stakeholders in the system and the concerns which they directly relate to them.

	Purchaser	Seller	Administrator	.Unregistered User	Owner
CN1	x	x	x	-	-
CN2	-	-	x	-	-
CN3	-	-	-	x	-
CN4	x	X	x	x	-
CN5	-	X	x	x	-
CN6	X	-	-	-	-
CN7	X	-	-	-	-
CN8	X	-	-	-	-
CN9	-	x	-	-	-
CN10	x	x	x	-	-
CN11	-	-	x	-	-
CN12	-	-	x	-	-
. . .					-

Chapter 3

Viewpoints+

3.1 *Context Viewpoint*

3.1.1 Overview

All systems exist in some larger environment, be it a department, an organisation's IT environment, a mobile communications system or even a virtual world. The Context view aims to elaborate on the existence and technical relationships that this system has with elements of the wider environment.

3.1.2 Concerns and stakeholders

3.1.3 Concerns

Identity and Responsibilities of External Entities:

The BorrowMyBooks system has a set of entities, internally and externally, with which it interacts through its processes. This set consists of human entities, mainly general users of the application. The responsibilities of the general user which interacts with the system is to:

- Sign up on the system as an active user
- Login to the system with their personal account details
- Browse the available books and their details
- Transact through the system, through either purchasing or buying a book
- Upload books to be sold or rented out
- Rate users for their reliability as buyers/sellers on the system
- Report users for abusive or unethical behavior on the system

Another external user is the administrator of the system. The administrator of the system has more complex responsibilities regarding the functioning of the system which are stated below.

- Signing in to the system with their relevant admin details.
- View reported users and block or suspend their interaction with the system depending on the severity of their abusive behavior on the system.

- Solve problems that arise with transactions which user attempt to make through the system
- Monitor the performance of the system and make sure that any bugs which may arise are eradicated.

Most important for the proper functioning of the system, is the database system which BorrowMyBooks relies on to store and manage all the data. The responsibilities of the database are as follows:

- Storing all information of users who are signed up on the system in a secure way.
- Storing all information of books which are registered on the system for sale or rent, and the availability of each specific book.
- Provide a reliable and secure connection to the system to avoid external entities tapping into private system data

Nature of External Connections:

Each external actor on the system has its own way of interacting with the system. The different communication methods for each user is stated below.

- General user interaction: There is a dedicated User Interface controller developed using the Model View controller architecture through which users of the system will interact with the system for input and output data. This interface has limited functionality to cater only for the needs and responsibilities of a general user of the system as described above.
- Admin user control: There is a dedicated user interface through which the administrators of the system can carry out their responsibilities for the functioning and maintenance of the system, and for any other issues concerning general users, system properties, system functionality and database management.
- Database system communications: The system communicates with the database through an external API which allows for secure and reliable database connections.

3.1.4 Typical stakeholders

The potential stakeholders in this view are listed below:

- General users of a system willing to purchase or rent a book
- General users of the system who are willing to sell or rent a book out through the system.
- Administrators.
- Database administrators

3.2 *Functional Viewpoint*

3.2.1 Overview

It is essential that the system's functional elements, their responsibilities, interfaces and primary interactions are described in the functional viewpoint. Its purpose is to drive the shape of other system structures such as the information structure, concurrency structure and deployment structure.

3.2.2 Concerns and stakeholders

3.2.3 Concerns

Identity of the system's functional elements:

The Borrow My Books system has various elements which lead to its overall functioning.

The elements are described below:

- A web based interface/browser (on the end user's side) which allows for:
    - A general user to interact with the system and its functionality
    - An administrator to perform administrator-related actions
- A server side system which allows for:
    - The storage of system information and data which can be accessed through the web based interface mentioned above.
    - The acquisition of the web interface for users to access the system's full functionality through their web browsers.

Responsibilities of the functional elements:

Server side:

The server side of the system is managed and built through the Heroku application management system. This system is responsible for management and development of the Borrow My Books application in the cloud. Furthermore, the MongoDB DBMS data model is used, and is responsible for storing all information related to the functioning of the application, and is accessed through the cloud.

Web Browser:

Any web browser can be used to access the site. The Borrow My Books application is created using Javascript which can be deployed to any web-based interface.

Primary interactions of the functional elements:

In order for the Borrow My Books system to be fully functional, it is essential that the cloud database used is fully accessible by the server side system which deploys the application to a user's web interface. The interactions of the elements all take place through online communications and server connections.

3.2.4 Typical stakeholders

The potential stakeholders in this view are listed below:

- Requirements Analysts
- System Developers
- Administrators
- General users

3.4 Model kinds+

3.5 *Model-View-Controller (MVC)*

3.5.1 Model View Controller conventions

The model view controller conventions allow for the independence of the various fundamental parts which make up the architecture. The constituent parts are described below:

- Model: The purpose of the model is to represent the data in the system. It is independent of all other parts of the system.
- View: displays the model data, and sends user actions to the controller. Again, it is independent of the model and the controller.
- Controller: the controller provides model data to the view, and interprets user interactions with the system.

The language used for the development of the Borrow My Books system is Javascript.

Conventions:

Conventions include languages, notations, modeling techniques, analytical methods and other operations. These are key modeling resources that the model kind makes available to architects and determine the vocabularies for constructing models of the kind and therefore, how those models are interpreted and used.

It can be useful to separate these conventions into a *language part*: in terms of a metamodel or specification of notation to be used and a *process part*: to describe modeling techniques used to create the models and methods which can be used on the models that result. These include operations on models of the model kind.

The remainder of this section focuses on the language part. The next section focuses on the process part.

The Standard does not prescribe *how* modeling conventions are to be documented. The conventions could be defined:

I) by reference to an existing notation or language (such as SADT, UML or an architecture description language such as ArchiMate or SysML) or to an existing technique (such as *M/M/*4 queues);

II) by presenting a metamodel defining its core constructs;

III) via a template for users to fill in;

IV) by some combination of these methods or in some other manner.

Further guidance on methods I) through III) is provided below.

Sometimes conventions are applicable across more than one model kind – it is not necessary to provide a separate set of conventions, a metamodel, notations, or operations for each, when a single specification is adequate.

I) Model kind languages or notations (optional)

Identify or define the notation used in models of the kind.

Identify an existing notation or model language or define one that can be used for models of this model kind. Describe its syntax, semantics, tool support, as needed.

II) Model kind metamodel (optional)

A metamodel presents the AD elements that constitute the vocabulary of a model kind, and their rules of combination. There are different ways of representing metamodels (such as UML class diagrams, OWL, eCore). The metamodel should present:

entities What are the major sorts of conceptual elements that are present in models of this kind?

attributes What properties do entities possess in models of this kind? **relationships** What relations are defined among entities in models of this kind?

constraints What constraints are there on entities, attributes and/or relationships and their combinations in models of this kind?

NOTE: *Metamodel constraints should not be confused with architecture constraints that apply to the subject being modeled, not the notations used.*

In the terms of the Standard, entities, attributes, relationships are *AD elements* per ISO/IEC/IEEE 42010, 3.4, 4.2.5 and 5.7.

In the *Views-and-Beyond* approach [1], each viewtype (which is similar to a viewpoint) is specified by a set of elements, properties, and relations (which correspond to entities, attributes and relationships here, respectively).

When a viewpoint specifies multiple model kinds it can be useful to specify a single viewpoint metamodel unifying the definition of the model kinds and the expression of correspondence rules. When defining an architecture framework, it may be helpful to use a single metamodel to express multiple, related viewpoints and model kinds.

III) Model kind templates (optional)

Provide a template or form specifying the format and/or content of models of this model kind.

3.5.2 <Model Kind Name> operations (optional)

Specify operations defined on models of this kind.

See §3.6 for further guidance.

3.5.3 <Model Kind Name> correspondence rules

? Document any correspondence rules associated with the model kind.

See §3.7 for further guidance.

3.6 Operations on views

Operations define the methods to be applied to views and their models. Types of operations include:

construction methods are the means by which views are constructed under this viewpoint. These operations could be in the form of process guidance (how to start, what to do next); or work product guidance (templates for views of this type). Construction techniques may also be heuristic: identifying styles, patterns, or other idioms to apply in the synthesis of the view.

interpretation methods which guide readers to understanding and interpreting architecture views and their models.

analysis methods are used to check, reason about, transform, predict, and evaluate architectural results from this view, including operations which refer to model correspondence rules.

implementation methods are the means by which to design and build systems using this view.

Another approach to categorizing operations is from Finkelstein et al. [2]. The *work plan* for a viewpoint defines 4 kinds of actions (on the view representations): *assembly actions* which contains the actions available to the developer to build a specification; *check actions* which contains the actions available to the developer to check the consistency of the specification; *viewpoint actions* which create new viewpoints as development proceeds; *guide actions* which provide the developer with guidance on what to do and when.

3.7 Correspondence rules

? Document any correspondence rules defined by this viewpoint or its model kinds.

Usually, these rules will be across models or across views since, constraints within a model kind will have been specified as part of the conventions of that model kind.

See: ISO/IEC/IEEE 42010, 4.2.6 and 5.7

3.8 Examples (optional)

Provide helpful examples of use of the viewpoint for the reader (architects and other stakeholders).

3.9 Notes (optional)

Provide any additional information that users of the viewpoint may need or find helpful.

3.10 Sources

? Identify sources for this architecture viewpoint, if any, including author, history, bibliographic references, prior art, per ISO/IEC/IEEE 42010, 7e.

Chapter 4

Views+

Much of the material in an AD is presented through its architecture views. Each view follows the conventions of its governing viewpoint. A view is made up of architecture models.

? Include an architecture view for each viewpoint selected in §3.

Repeat and complete the following section for each architecture view in the AD.

4.1 View: <View Name>

? Give the architecture view a <View Name>.

? Provide any identifying and supplementary information about <View Name>.

The details of this information will be as specified by the organization and/or project. See §1 for examples of identifying and supplementary information.

Views have their own identifying and supplementary information distinct from ADs because they may be developed and evolve separately over the lifetime of a project.

? Identify the viewpoint governing this view from among those identified in §3.

See also: ISO/IEC/IEEE 42010, 5.5

4.1.1 Models+

An architecture view is composed of one or more architecture models.

? Provide one or more architecture models adhering to the governing viewpoint.

? The models must address all of the concerns framed by the view's governing viewpoint and cover the whole system from that viewpoint.

Repeat the section below for each model.

4.1.2 <Model Name>

? Each architecture model shall include version identification as specified by the organization and/or project.

? Each architecture model shall identify its governing model kind and adhere to the conventions of that model kind from §3.5.

See ISO/IEC/IEEE 42010, 5.4.

An architecture model may be a part of more than one architecture view. This enables sharing of details and addressing distinct but related concerns without redundancy. Other uses of multiple models: aspect-oriented style of architecture description: architecture models shared across architecture views can be used to express architectural perspectives [7] and architecture textures [6]. Architecture models can be used as containers for applying architecture patterns or architecture styles to express fundamental schemes (such as layers, three-tier, peer-to-peer, model-view-controller) within architecture views.

4.1.3 Known Issues with View

? Document any discrepancies between the view and its viewpoint conventions. Each architecture view must adhere to the conventions of its governing architecture viewpoint.

Known issues could include: inconsistencies, items to be completed, open or unresolved issues, exceptions and deviations from the conventions established by the viewpoint. Open issues can lead to decisions to be made. Exceptions and deviations can be documented as decision outcomes and rationale.

Chapter 5

Consistency and correspondences

This chapter describes consistency requirements, recording of known inconsistencies in an AD, and the use and documentation of correspondences and correspondence rules.

5.1 Known inconsistencies

? Record any known inconsistencies in the AD.

Although consistent ADs obviously are to be preferred, it is sometimes infeasible or impractical to resolve all inconsistencies for reasons of time, effort, or insufficient information.

2 An architecture description should include an analysis of consistency of its architecture models and its views.

5.2 Correspondences in the AD

? Identify each correspondence in the AD and its participating AD elements. Identify any correspondence rules governing

Correspondences are used to express, record, enforce and analyze consistency between models, views and other AD elements within an architecture description, between ADs, or between an AD and other forms of documentation.

AD elements include instances of stakeholders, concerns, viewpoints and views, model kinds and models, decisions and rationales. Constructs introduced by viewpoints and model kinds are also AD elements.

Correspondences are n-ary mathematical relations. Correspondences can be depicted via tables, via links, or via other forms of association (such as in UML).

5.3 Correspondence rules

? Identify each correspondence rule applying to the AD.

Correspondence rules can be introduced by the AD, by one of its viewpoints, or from an architecture framework or architecture description language being used.

? For each identified correspondence rule, record whether the rule holds (is satisfied) or otherwise record all known violations.

Appendix A

Architecture decisions and rationale

It is not required by the Standard to capture architecture decisions. This section describes recommendations ("shoulds") for their recording.

A.1 Decisions

2 Provide evidence of consideration of alternatives and the rationale for the choices made.

2 Record architecture decisions considered to be key to the architecture of <System of Interest>.

Areas to consider to selecting key decisions include those:

- affecting key stakeholders or many stakeholders
- essential to project planning and management
- expensive to enforce or implement
- highly sensitive to changes or costly to change
- involving intricate or non-obvious reasoning
- pertaining to architecturally significant requirements
- requiring major expenditures of time or effort to make
- resulting in capital expenditures or indirect costs

2 When recording decisions, the following information items should be considered:

- unique identifier for the decision
- statement of the decision
- correspondences or linkages concerns to which it pertains
- owner of the decision
- correspondences or linkages to affected AD elements
- rationale linked to the decision
- forces and constraints on the decision
- assumptions influencing the decision
- considered alternatives and their potential consequences

See [3] and references there for various approaches to documenting decisions compatible with the Standard.

The template ends here!

Bibliography

Clements, Paul C. et al. *Documenting Software Architectures: views and beyond*. 2nd. Addison Wesley, 2010.

Finkelstein, A. et al. "Viewpoints: a framework for integrating multiple perspectives in system development". In: *International Journal of Software Engineering and Knowledge Engineering* 2.1 (Mar. 1992), pp. 31–57.

Heesch, Uwe van, Paris Avgeriou, and Rich Hilliard. "A Documentation Framework for Architecture Decisions". In: *The Journal of Systems & Software* 85.4 (Apr. 2012), pp. 795–820. DOI: 10.1016/j.jss.2011.10.017.

IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Oct. 2000.

ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description. Dec. 2011, pp. 1–46.

Ran, Alexander. "ARES Conceptual Framework for Software Architecture". In: *Software Architecture for Product Families Principles and Practice*. Ed. by M. Jazayeri, A. Ran, and F. van der Linden. Addison-Wesley, 2000, pp. 1–29.

Rozanski, Nick and Eoin Woods.´ *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. 2nd. Addison Wesley, 2011.

Contents