Approach to Rapid Software Design of Mobile Applications' User Interface

Vyacheslav Chernikov
Voronezh State Technical University
Voronezh, Russia
slava@binwell.com

***Abstract*–Mobile applications have a lot of unique challenges during the development process. One of them is monolith architecture that can't be skipped even for large Mobile Enterprise applications. Software engineer should keep mind the whole source code structure to be effective. This article will introduce an approach to Rapid Software Design that allows to make a working documentation related to code and understandable by all team members. Rapid Software Design is suitable for Waterfall/Agile processes of developing applications with graphical user interface (web, mobile, desktop, embedded) and initially was created for mobile projects. This article contains only the description of User Interface related documentation despite the whole Rapid Software Design also covering the usage of architecture patterns, business logic, services, features of mobile operation systems and writing automated tests.**

I. Introduction

During the process of software development, it is necessary to consider the interests of several groups of participants: business customers, software and graphic designers, testers, and developers. One of the specifics of mobile business applications is their low complexity in comparison to corporate backend-services.

Most of software development teams face the following problems during the long-term development of a project [1], [2], [3]:

1) The absence or non-observance of architectural patterns, which leads to a chaotic arrangement of files in the structure of the solution. It increases the so called 'technical debt' when it's became hard to maintain and improve the source code.

2) The lack of a single documentation (except for Requirements and Backlog) for the whole team, which would be pretty compact and accessible, while at the same time making it easier to find a ubiquitous language.

3) "Documentation separately, code separately" - names from the documentation used in the code rarely, which complicates its development and progress. Also, it makes a barrier in communication between business and engineering teams – every team use its own documentation and terms (naming mind algorithms).

This article describes a step-by-step process of preparing technical documentation that will allow software engineers to *create a "skeleton" for the source code based on the user interface*. The article is focused on the simple online documents which the team members can use in communication and development process. Main questions covered by the article are:

1) How to expand the software product documentation so that it is easier for the developer to prepare the source code structure?
2) How to use the technical documentation for rapid software design?
3) How to organize team communication with a technical documentation during the software design process?
4) How to use the technical documentation as a checklist with possible automation of code review process?

Described approach allows to combine the software design with writing technical documentation. This approach was created for Mobile Enterprise applications and it's is easy to learn and use. Also, it can be applied to any business applications with Graphical User Interface (web, desktop, embedded).

II. Software project documentation

The core idea of the approach to Rapid Software Design is: *a mobile application is primarily a Graphical User Interface (GUI) with low "under the hood" complexity – large data and complex business logic should be implemented on the backend online service.*

In fact, the Business Domain Model within mobile application can be described by the User Interface. Software Designer should consider only those data (and their derivatives) that are entered by the user, displayed on the screen, and control the application behavior. Business scenarios are also directly tied to the behavior of the user interface.

At the same time, most Requirements/Technical Specification docs are prepared for business customers and describe whole business scenarios and functional blocks but no not specific screens or classes. Requirements documentation and graphic design specifications are used by the development team. Software and Quality Assurance engineers should create its own documentation for making a high-quality software. This documentation includes Architecture description and Test Cases. In the following chapters the minimum required set of additional documents will be described.

First of all, let's look at the development process as a whole. For simplicity, we've chosen a Linear development process - Fig. 1.

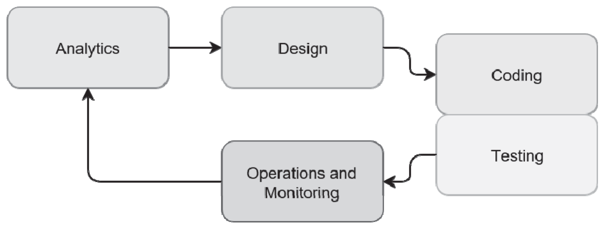


Fig. 1. Linear development process

So, the project usually allocates the following tasks:

- Analysis
- Design
- Coding
- Testing
- Operations

At the Analytics stage business analyst should gather general requirements for the application - the Specifications document. These requirements are used during the Design phase (design of User Experience (UX), Graphic Design and Software Design).

Approach to Rapid Software Design suppose that development team already has the Specifications Document and all screens/pages UI design or at least rough schemes.

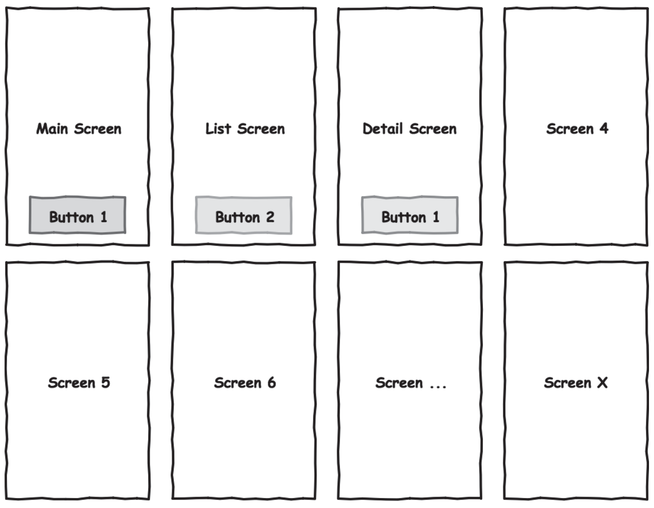


Fig. 2. Screens list from graphical designer

The Graphic Design is actually derived from UX and fills the original screen schemes with emotions, composition, animations and other aspects of visual behavior. The screen schemes itself already create the structure of the application and data models – which data to display on the screen, how data fields will be grouped, and how they will affect the interface behavior.

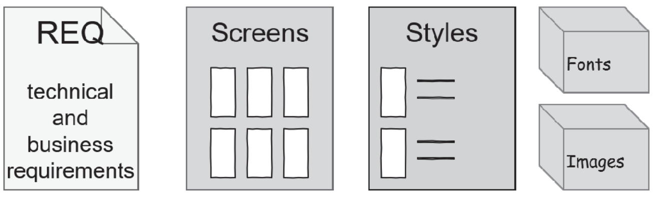


Fig. 3. Initial list of artifacts from regular Design phase

When the Design phase is complete a set of necessary specifications will be created. These documents and artifacts include:

- Technical and Business requirements document
- Graphic design of all screens/pages
- Graphic Style Specifications
- Architecture overview
- Fonts, Images and other graphic files

In most cases development team have only this set of artifacts. Rapid Software Design should be added to the end of Design phase. In the next chapter Rapid Software Design will be introduced.

III. Rapid Software Design

Let's remind once again that mobile applications are primarily a user interface. It is necessary to determine steps that the user has to go through in order to obtain the desired result/goal. Business applications are created for a specific set of key scenarios (user interaction sequences) to achieve business goals. These goals are related to initial use Tasks or Intents – Fig. 4.

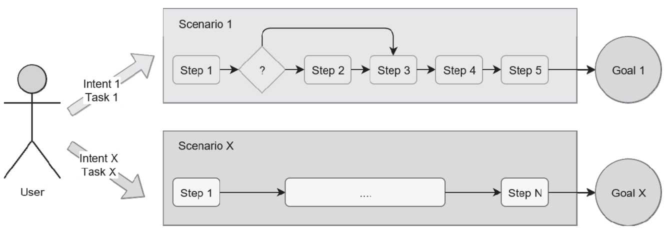


Fig. 4. User scenarios

The average time of user contact with a smartphone is only a few minutes [4] – Fig. 5. That is why the number of steps in mobile applications should not be large. User should be able to get a necessary result with only a few steps. It’s better to split the application into relatively short scenarios of not more than 10 steps each.

Rapid Software Design embrace this concern to provide a set of additional documents. One of them, Map of Transitions and States, is used to visualize user scenarios.

First step of Rapid Software Design is screens/pages grouping, naming and numbering.

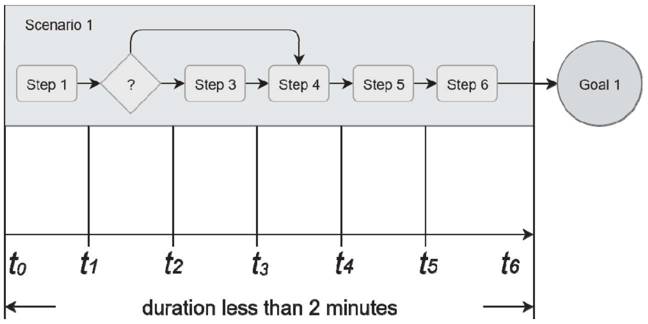


Fig. 5. User scenarios duration

A. Screen Grouping, Naming and Numbering

Let's remind that whole mobile project team typically includes a number of sub-teams: business customers, analysts, graphical designers, development and testing engineers and marketing team to gather and analyze user feedbacks and behavior. Every sub-team prefer using its own terms and naming algorithms. To use ubiquitous terms for the whole team it is necessary to start with screen grouping, naming and numbering. Page names and numbers should be used in project documentation.

Graphical designers provide the set of transitions between screens/pages – Fig. 6.

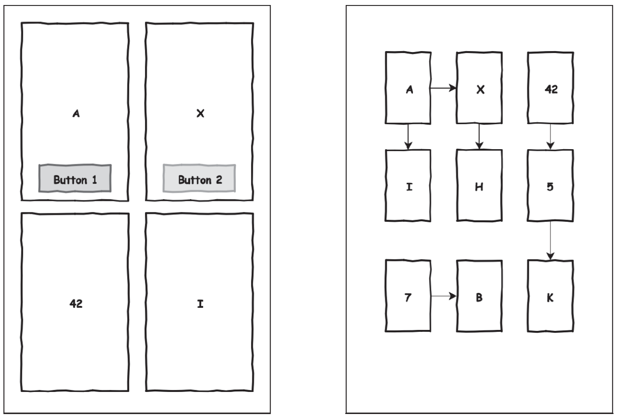


Fig. 6. Screen transitions from graphical designer

Rapid Software Design starts with splitting the application into parts (sections of the Business Domain Model) based on the screens list. Mobile applications are primarily an interface with the user, that's why screens/pages directly reflect the domain model available to user.

The first step is to select screens that are linked together – Fig. 7. Usually such screens are already logically grouped within user scenarios. For example, in most applications Account section can be distinguished where use can view or edit personal information – Fig.8.

After this step Screens List document will be created. An example above contains the following sections:

1. Account
2. Help
3. Checkout
4. Catalog

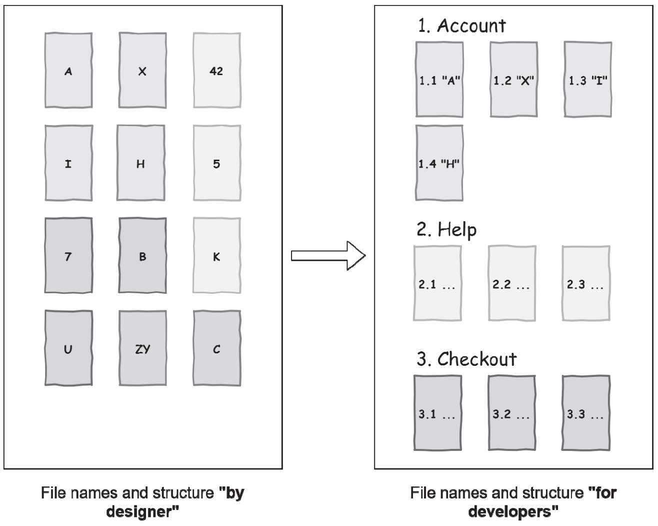


Fig. 7. Screens grouping for developers

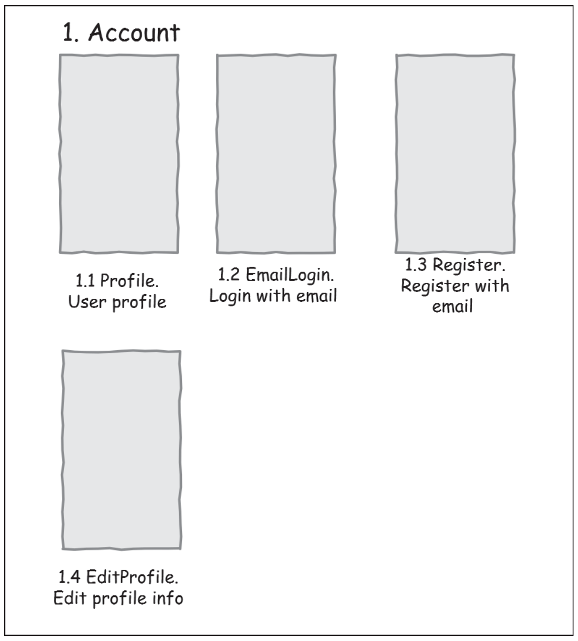

1. Account
1.1 Profile. User profile
1.2 EmailLogin. Login with email
1.3 Register. Register with email
1.4 EditProfile. Edit profile info

Fig. 8. Screens group with numbers, names and short description

Each section must have a name and number. Section names should be used to horizontally split the Data Access Layer, Business Logic and User Interface. For example, the Data Access Layer (groups of methods for interacting with server APIs and accessing to local database) will be divided into sections, and each of them will serve its own set of screens:

DAL\DataServices

AccountDataService

HelpDataService

CheckoutDataService

CatalogDataService

Each of the data services can completely hide all the work with the server, disk cache and local database. Data service should be a kind of a "black box" and provide external methods described in Table of Screens covered by section B.

Then it is necessary to number and name every screen/page. This step will create a flat tree-like interface structure without nesting and transitions – Fig. 9.

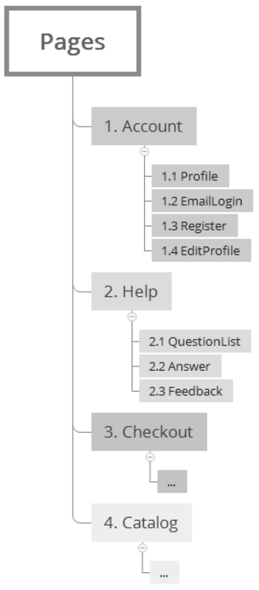


Fig. 9. Screens tree

Screen names will be used in class names. For example, with MVVM (Model-View-ViewModel) architecture pattern it's necessary to have 2 classes: View (or Page/Screen) and ViewModel. These classes will have the common part from the Screens List:

1.1 Profile

ProfilePage

ProfileViewModel

1.2 EmailLogin

EmailLoginPage

EmailLoginViewModel

Using this naming algorithm, it's possible to create almost complete project structure:

- Classes for every data service with name from Screens List (section name is the name of the Data Service)
- Classes
- and folders for every ViewModel/Controller
- Classes, folders and XML description of user interface per page.

After this step, the development team will get a "project skeleton". This "skeleton" should also rely on architecture and base classes (e.g. BasePage, BaseViewModel).

An example of the project structure is shown below.

User Interface (UI)

UI\Pages

\Account

ProfilePage

...

Business Logic (BL)

BL\ViewModels

\Account

ProfileViewModel

...

Data Access Layer (DAL)

DAL\Models

ProfileModel

ProductModel

...

DAL\DataServices

AccountDataService

...

Now it's possible to focus on the behavior of every screen/page. To achieve this goal, it's necessary to use online Table of Screens.

B. Table of Screens

During the development process software engineer should not only keep in mind the whole project structure (folders and classes) but also a full text description of every screen/page. This information could be gathered in Table of Screens (regular Google Sheet or Microsoft Excel Table) – Fig. 10. The key columns of the table are:

A. Screen number and Section color

B. (Name) Short Name

C. (States) List of possible states

D. (Validation) Input fields for Validation

E. (Behavior) Text description of the screen and its behavior

These columns provide all necessary "external" requirements for the page. It's better to mark every section with unique color - this will simplify the work with the Map of Transitions and States described in section "D. Map of Transitions and States".

Additionally, the following columns can be added to this table:

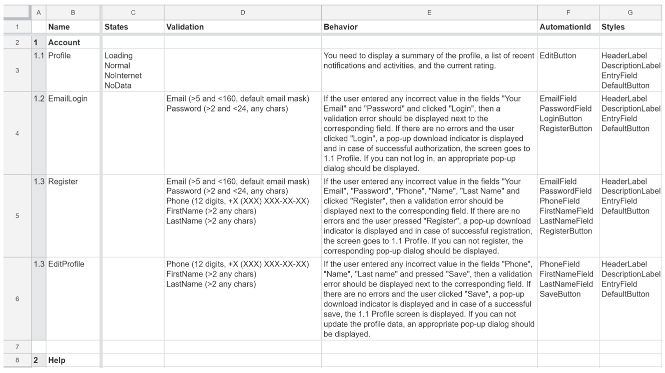
F. (AutomationId) UI-control identifiers (e.g. LoginButton) for writing automated UI-tests.

G. (Styles) Style names for nested page controls and views.

H. (Models) Models that are used on the screen/page - e.g. ProfileModel.

I. (DAL Methods) DAL methods that are used on the screen to get ready-to-use Models.

Every column, except Behavior, contain minimum of the information required for coding. The Behavior column should contain a detailed screen/page description.



	A	B	C	D	E	F	G
1		**Name**	**States**	**Validation**	**Behavior**	**AutomationId**	**Styles**
2	**1**	**Account**					
3	1.1	Profile	Loading Normal NoInternet NoData		You need to display a summary of the profile, a list of recent notifications and activities, and the current rating.	EditButton	HeaderLabel DescriptionLabel EntryField DefaultButton
4	1.2	EmailLogin		Email (>5 and <160, default email mask) Password (>2 and <24, any chars)	If the user entered any incorrect value in the fields "Your Email" and "Password" and clicked "Login", then a validation error should be displayed next to the corresponding field. If there are no errors and the user clicked "Login", a pop-up download indicator is displayed and in case of successful authorization, the screen goes to 1.1 Profile. If you can not log in, an appropriate pop-up dialog should be displayed.	EmailField PasswordField LoginButton RegisterButton	HeaderLabel DescriptionLabel EntryField DefaultButton
5	1.3	Register		Email (>5 and <160, default email mask) Password (>2 and <24, any chars) Phone (12 digits, +X (XXX) XXX-XX-XX) FirstName (>2 any chars) LastName (>2 any chars)	If the user entered any incorrect value in the fields "Your Email", "Password", "Phone", "Name", "Last Name" and clicked "Register", then a validation error should be displayed next to the corresponding field. If there are no errors and the user pressed "Register", a pop-up download indicator is displayed and in case of successful registration, the screen goes to 1.1 Profile. If you can not register, the corresponding pop-up dialog should be displayed.	EmailField PasswordField PhoneField FirstNameField LastNameField RegisterButton	HeaderLabel DescriptionLabel EntryField DefaultButton
6	1.3	EditProfile		Phone (12 digits, +X (XXX) XXX-XX-XX) FirstName (>2 any chars) LastName (>2 any chars)	If the user entered any incorrect value in the fields "Phone", "Name", "Last name" and pressed "Save", then a validation error should be displayed next to the corresponding field. If there are no errors and the user clicked "Save", a pop-up download indicator is displayed and in case of a successful save, the 1.1 Profile screen is displayed. If you can not update the profile data, an appropriate pop-up dialog should be displayed.	PhoneField FirstNameField LastNameField SaveButton	HeaderLabel DescriptionLabel EntryField DefaultButton
7							
8	**2**	**Help**					

Fig. 10. Table of Screens sample

C. Screen/Page states

Most of mobile applications receive data over the Internet and should display the downloading state correctly:

- Display loading progress bar/indicator
- Display loaded data
- Display a message about offline mode (smartphone have no Internet connection)
- Display an error message when server is unavailable or return error response
- Display a message when server returned no data (e.g. an empty products list)

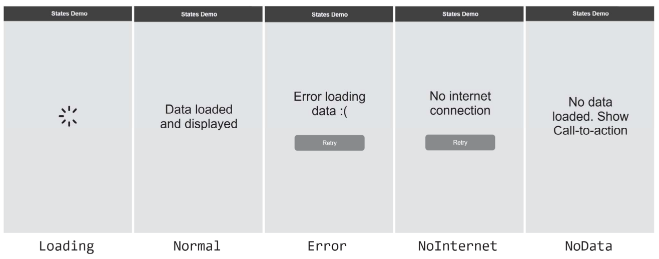


Fig. 11. Screen states

"State" is a set of UI controls/views that should be displayed to user within a single screen. Business Logic (ViewModels/Controllers) should switch screen from one state to another during the data loading process. This states transitions can be animated to improve user experience. Negative states can be tracked with diagnostic tool (e.g. online error tracking service for mobile applications). Samples of screen states – Fig 11. There are 2 types of states:

1) "Good states" don't break the user scenario – user can go father and achieve his goal (e.g. Loading and Normal states).

2) "Bad states" brake the user scenario (e.g. Error, NoInternet, NoData)

Using states is mandatory with Rapid Software Design approach. States should be added to Table of Screens document.

With screen sections, names, numbers, colors, states and transitions it is possible to create a Map of Transitions and States.

D. Map of Transitions and States

Map of Transitions and States reflects the user journey map where every step is represented by a single screen. User go from one screen to another to achieve desired goals. Map of Transitions and States can also be used for better understanding of user scenarios.

Map of Transitions and States begins with the starting point - the moment an application was launched by the user. There can be several starting points, for example, one point for an authorized user, another for an unauthorized user, and a third one for launching from Push notification.

Every screen of an application consists of colored (section color) rectangles (one screen/page = one rectangle) and transition arrows. Also, it's possible to add AutomationIds of the buttons or events that caused the transition. Optionally, it's possible to add navigation data that passes from one screen to another (e.g. "email" or "questionId").

Also, Map of Transitions and States should display all possible states for every screen. "Good states" are marked with "+" (plus = good) and "Bad states" with "-" (minus = bad). There is no need to add the "back" arrows because "back navigation" is available for all screen by default on mobile platforms. Sample of Map of Transitions and States shown on Fig. 12.

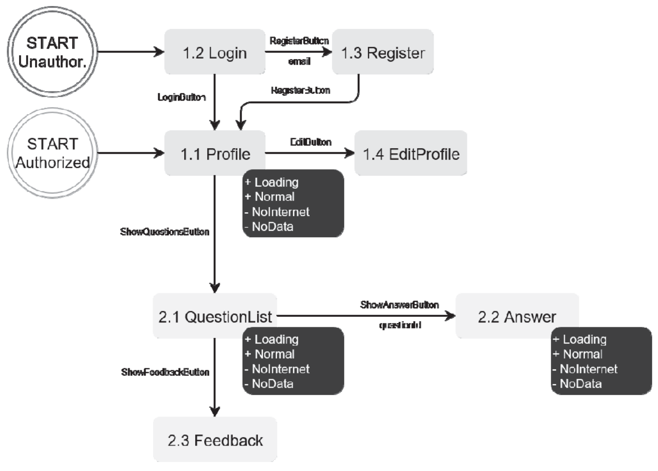
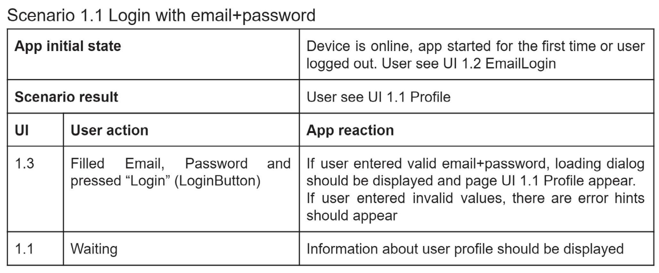


Fig. 12. Sample Map of Transitions and States

It's mandatory to use colors, numbers, names and states from Table of Screens within Map of Transitions and States. AutomationIds and navigation data are optional.

IV. User scenarios

All applications are creating to solve specific user tasks. To achieve goals user should go through screens and make action on that screens (button click, selecting an item from list, etc.). That's why it's necessary to indicate screen numbers, names and AutomationIds within user scenarios description. Typical user scenario description displayed on Fig. 13.



Scenario 1.1 Login with email+password

App initial state		Device is online, app started for the first time or user logged out. User see UI 1.2 EmailLogin
Scenario result		User see UI 1.1 Profile
UI	**User action**	**App reaction**
1.3	Filled Email, Password and pressed "Login" (LoginButton)	If user entered valid email+password, loading dialog should be displayed and page UI 1.1 Profile appear. If user entered invalid values, there are error hints should appear
1.1	Waiting	Information about user profile should be displayed

Fig. 13. Sample of user scenario description

Screen names and AutomationIds will be used in automatic UI testing when QA engineer write scripts (e.g. with Appium framework) that mimic the user behavior. These scripts are used as a part of Continues Integration/Continues Delivery pipeline.

V. The final set of software design documents

With the described set of documents engineering team (development and testing) gets the complete vision of application structure. It's possible to use regular and familiar tools to create these online documents:

- Text editor (e.g. Google Docs)
- Table editor (e.g. Google Sheets)
- Graphic editor (e.g. Draw.io)

The preparation of each document takes only one or two days, but it greatly simplifies the process of development and testing of the product.

As a result of Rapid Software Design phase team gets a complete set of documentation required by software engineers and understandable by business and marketing sub-teams – Fig. 14.

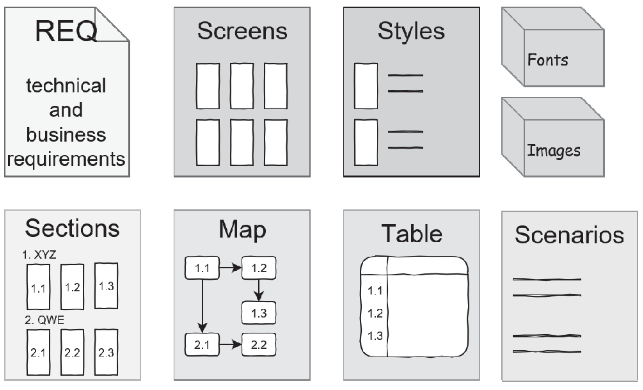


Fig. 14. Final set of design artifacts with Rapid Software Design

VI. Create a project skeleton with Rapid Software Design

Every experienced developer has a set of known base classes and architecture understanding. These classes will be used in project skeleton.

With the Rapid Software Design, it's possible to create a complete project skeleton in step-by-step way:

Step 1. Create an empty application with create base classes and empty folders for every architecture module (one architecture model = one folder; nesting of folders should also be kept in mind).

Step 2. Add subfolders for each section for ViewModels/Controllers and screens/pages. Add empty classes of every ViewModel and Page from Screens List.

Step 3. Add transitions between screens and their states based on the Map of Transitions and States. For that purpose, buttons like "Go to screen Login" should be added on every page.

Step 4. Add empty Data Services (all methods returning dumb results) and complete Models based on the Table of Screens.

Step 5. Add and implement all styles based on Table of Screens and Style Specifications by graphical designer.

Step 6. Implement Data Services with Mock data. Every Data Service method should return complete Model with data loaded from local JSON/CSV file.

After these steps developers will get the complete project skeleton. It takes one or two days to create a solid project skeleton that already have templates for all screen/pages and data services with real data. On the next steps developers can focus on single screens and implementing business and data access logic (accessing to server APIs and using local cache or database).

VII. Conclusion

This article completely the Rapid Software Design part related to User Interface. It takes 1-3 additional days (that's why it was called 'Rapid') to complete the Software Design phase and create a project skeleton based on the documentation. This dramatically reduce the complexity of the project and 'entry' barrier for new team members.

Brief description of documentation from major Rapid Software Design steps:

- Sections of grouped screens list. This document is used for grouping, naming and numbering screens
- Table of Screens. Allows to get a complete description of every single screen. Names this table correspond to the class and properties names in the code
- Map of Transitions and States. Allows to see all possible user scenarios and the entire user interface map
- User Scenarios. Describe the interaction with the user in "user action" – "system reaction" way

Rapid Software Design approach simplify the development of mobile applications and can be used for automatic generation of code from the documentation and documentation from the code. Also provided set of documents can be used as a checklist for manual/automated code reviews and quality assurance.

Rapid Software Design approach was used in mobile projects by Binwell company since 2015 with teams of 2-7 software engineers. Results of using Rapid Software Design:

- Whole team including business and marketing get a solid and simple documentation to manage product development process
- Whole team use ubiquitous language (names and numbers from the documentation) to communicate, terms from documentation are used in code
- Development team gets a project skeleton to speedup the process of creating a new product, reduce uncertainty and keep low technical debt
- Testing team gets a documentation-based checklist for manual testing. Also, this documentation is used for writing automatic user interface tests
- Described set of documents reduce "jumpstart" period for new team members.

REFERENCES

[1] J. Ousterhout, *A Philosophy of Software Design*. Yaknyam Press, 2018.

[2] R. Hoda, J. Noble, S. Marshall, "Documentation Strategies on Agile Software Development Projects", *Agile and Extreme Software Development*, vol.1, no.1, 2012, pp. 23-37.

[3] D. Verma, J. Gesell, H. Siy, M. Zand, "Lack of Software Engineering Practices in the Development of Bioinformatics Software", *ICCGI 2013: The Eighth International Multi-Conference on Computing in the Global Information Technology*, 2013, pp. 57-62.

[4] The Statistics Portal, Average mobile app session length as of 4th quarter 2015, by category (in minutes), Web: https://www.statista.com/statistics/202485/average-ipad-app-session-length-by-app-categories.

[5] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, X. Franch, "Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal", *PROFES 2017: Product-Focused Software Process Improvement*, vol. 10611, 2017, pp. 515-522.

[6] C. J. Satish, M. Anand, "Software Documentation Management Issues and Practices: a Survey", *Indian Journal of Science and Technology*, vol. 9(20), 2016, pp. 1-7.

[7] V. Chernikov, "Methodology of Developing Cross-platform User Applications", Voronezh State Technical University, in press.