Home Furniture Application  
Software Architecture Document (SAD)

PRODUCED BY: JED BRUNDIDGE & KOMI ABOTSI

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Problem Definition

Throughout a lifetime, everyone will be faced with the task of furnishing a home. Our application seeks to aid prospective clients in choosing the perfect furniture for their space by allowing them to create a virtual representation of their unique situation and then choosing from an assortment of different furniture to fill their rooms. This allows people to make choices about what type of furniture is best for them and where it should be placed. This application will allow buyers to be more informed about their purchases by giving them a virtual representation of the type of furniture that will be in the room, the color of the furniture, and the best options for placing the furniture. This allows a smoother more informed purchase and allows sellers to focus on sending out quality products that satisfy their customers.

# Stakeholders

Primary Stakeholders

* Development Team (Jed & Komi)
* Sponsor (Prof. Arun)
* Shoppers
* Sellers
* Manufacturers

Secondary Stakeholders

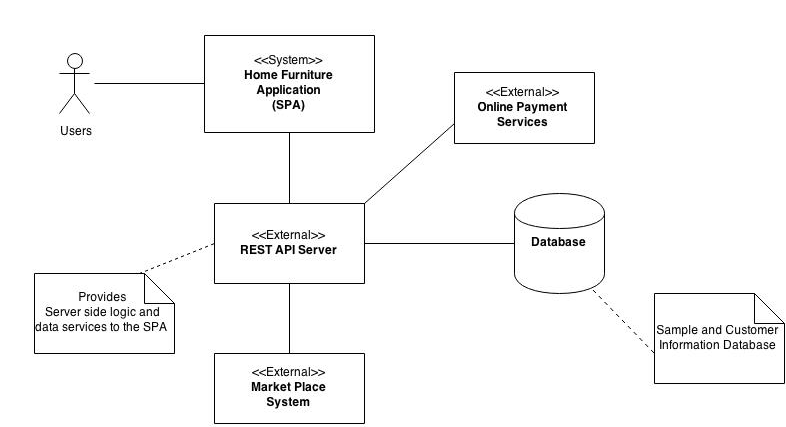
* SEIS 708 classmates
* Home furniture designers

# Views from the viewpoint catalog

## Context viewpoint

### Scope Definition

The system is a web-based application that will provide a home design environment to the user. The user will be able to try out several styles of furniture, based on the shape of a room he/she would like to furnish.



**Figure I.1.** UML Context Diagram of the HFA

### System Entities

As a web application, the Home Furniture Application (HFA) is backed by a Web Server. The web server handles all the business logics and data transactions between the HFA and the external entities and is responsible of loading the templates, and providing the web services that make up the single page application at start-up.

### Database

The database stores user data, and analytics data on user logging and activity history.

### External entities

#### Market Place Systems

The web server consumes services from Market Place Systems. Such systems propose samples of models that the user is interested in. The HFA will be designed so that it provides services that will allow such systems to receive requests for samples from the user.

### Stakeholders Concerns for the context viewpoint

|  |  |
| --- | --- |
| **Stakeholder Class** | **Concerns** |
| Developers | All concerns |
| Sponsor | System scope |
| Users | System scope, external entities, services provided, consistency and coherence |
| Assessors (Classmates) | All concerns |

**Table I.1.** Stakeholder concerns for the context viewpoint

## Functional Viewpoint

### Functional elements and Capabilities

* The system provides a set of virtual rooms displayed on a webpage, categorized by size, shape and type (living room, bathroom etc.)
* The virtual rooms can be furnished by drag-and-drop of furniture, load
* The final design will yield a room with the selected furniture, arranged by the user of optimized based on settings that are the closest to the user’s design. (An algorithm will compute the similarity)
* At user’s discretion buying options can be displayed as well

|  |  |
| --- | --- |
| **Functional Elements** | **Capability** |
| Virtual room engine | * Displays virtual room * Displays furniture catalog provided by Market Place Systems * Optimize room furnishing based on user selection |
| User account manager (variable) | * Authenticates user * Manages user logging and browsing history * Provides analytics features |
| UI Controller (variable) | * Dispatch requests |
| Catalog Manager | * Interacts with Market Place systems to load the user interface with furniture catalogs |

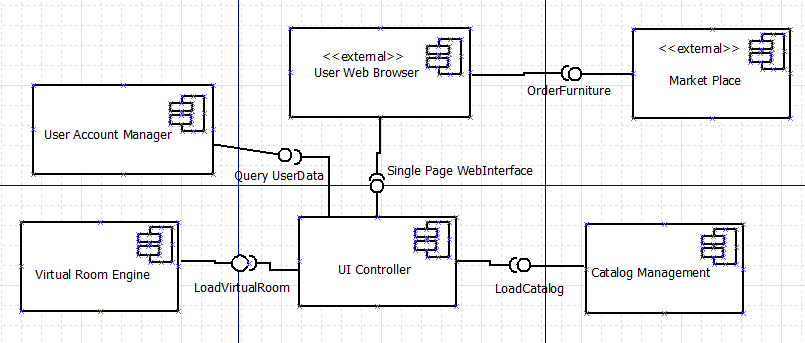
### External Interfaces

* Data flows between the Server and Client
* Services exchange between Server and market Place Systems
* Services exchange between Client and Payment Services.
* Relational data flows between the Server and the database

### Stakeholders concern

|  |  |
| --- | --- |
| **Stakeholder Class** | **Concerns** |
| Developers | Design logics, functional capabilities and external interfaces |
| Sponsor | Mostly functional capabilities |
| Users | Functionalities and external interfaces |
| Assessors (Classmates) | All |

### UML Component Diagram

****

## Development Viewpoint

Several design alternatives are available when it comes to the development viewpoint. We will analyze how we can satisfy critical quality scenarios (to be presented), using one or the other architectural style.

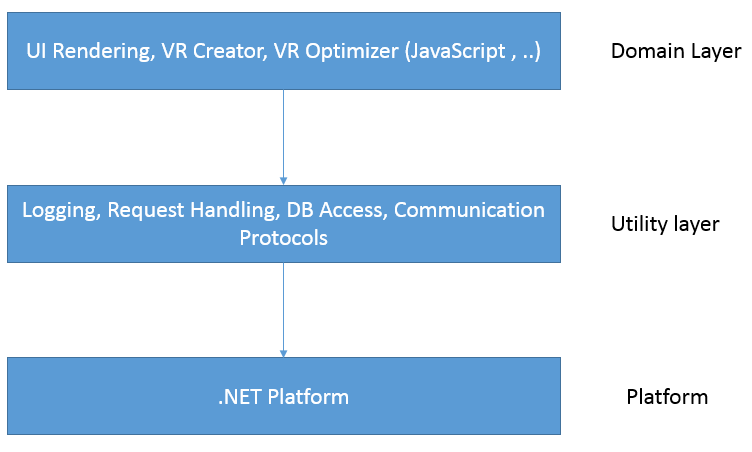
### Strategy

The HFA can publish and retrieve content from and to the REST server. At start-up all the catalogs and template files for content will be loaded as expected for a normal website. We will contemplate using a single page architectural style for quick responsiveness and a delightful usability.

### API’s

We assume that Online Shopping Systems provides the services for catalogs and shopping to be consumed by our server. At the same time the HFA provides API’s for Market Place Systems to query user’s design and build up batches of catalogs that will satisfy the user.

### Structure Model (TBD)



### Stakeholder Concerns

|  |  |
| --- | --- |
| **Stakeholder Class** | **Concerns** |
| Developers | Architectural style, and design pattern |
| Sponsor | Feasibility |
| Users | Services |
| Assessors (Classmates) | All |

## Deployment Viewpoint

### Runtime Platform Required

The Home Furniture System is a web-based application that is intended to run in any browser on any operating systems, as well as on mobile devices (tablets preferably, due to usability concerns)

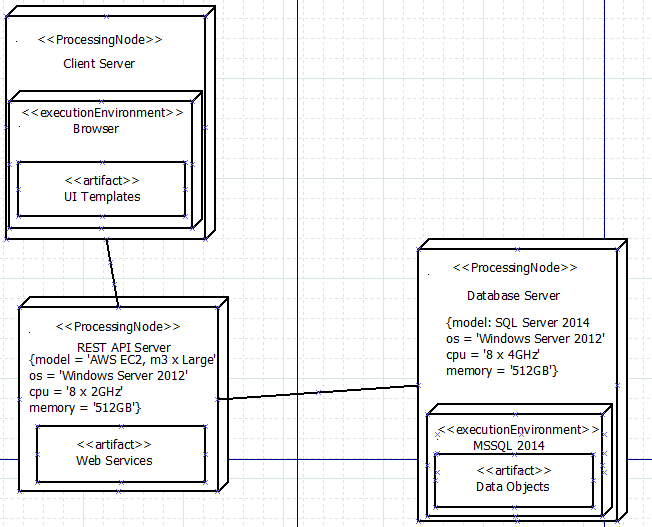
### Hardware and Hosting

We will turn to Amazon Cloud for hardware and hosting services. This decision primarily is motivated but the easy set-up and the possibility to scale-up and scale-out as needed, and also the flexibility that is offered for testing. Two m3xlarge instances (one for the server and the other for the database) will be our dedicated servers.

|  |  |  |  |
| --- | --- | --- | --- |
|  | vCPU | Mem (GB) | Storage |
| M3 x Large | 4 | 15 | 1T + |

**Fig 4.1 m3 x Large Specs**

### UML Deployment Diagram



### Stakeholders’ concerns

|  |  |
| --- | --- |
| **Stakeholder Class** | **Concerns** |
| Developers | Testing and deployment concerns |
| Sponsor | Optimization |
| Users | Accessibility and Performance |
| Assessors (Classmates) | All |

# Perspectives

## Security Perspective

Desired Quality: It is desired that the system be able to monitor and protect user information and not allow outside forces to manipulate the site or the information in a way that will harm the business or its customers.

### Sensitive Resource Identification

|  |  |  |  |
| --- | --- | --- | --- |
| **Resource** | **Sensitivity** | **Owner** | **Access Control** |
| User account records | Identity theft and privacy violation risks | Customer Care Group | No direct data access |
| Product Catalog | Price and inventory manipulation | Stock Management Group | No direct data access |
| User activity records | Intrusive analytics | Customer Care Group | Access to individual record by authenticated user and system |
| Business operations with third party | Data access and integrity | Business management Group | Authenticated principal only |

### Security Policy (Access Control)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **User Account Records** | **Product Catalog** | **User activity records** | **Business Operations with Third-party** |
| Data Admin | Full with audit | Full | Full with audit | Full with approval of Business admin |
| Catalog Manager | None | Full | Full | Full with approval from Business admin |
| Business Admin | None | Read-only | Full | Full |
| Customer Care | None | Read-only | None | Read-only |
| Registered Customer | Full on own records | Read-only | Full on own-records | None |
| Unknown user | None | Read-only | None | None |

### Security Tactics

* Username and password authentication.
* Access token authentication for session timeout.
* Leverage security infrastructure technologies provided in AWS Cloud.

## Performance and Scalability

Desired Quality: It is desired that the system be able to execute and perform all required tasks within an acceptable time frame with the ability to not lose performance as traffic increases.

### Performance Table

|  |  |
| --- | --- |
| **Processes** | **Benchmark** |
| Authentication | 100 ms |
| Design Action Visualization | 100 ms |
| Connect to market Place | 150 ms |
| Round trip to database with “basic” query | 50 ms |

### Workload and Traffic Management

* Quality of service policies:

During peak period a threshold of 10,000 requests/s is set as limit to enact QoS based on user profile (Unknown, Registered, Member)

* Infrastructure policies

Deciding to deploy the system’s infrastructure in AWS Cloud gives the advantage of scaling out the infrastructure if the QOS policies can’t handle performance problems.

## Availability and Resilience

Desired Qualities: It is desired that the system be fully operational when required and effectively handle system failures.

Applicability:

Context - Gather information on how external systems may affect the availability of our application. Determine how the application will interact with these external systems.

Functional - Changes to the functional requirements may need to be changed to accommodate the availability of the system.

Information - Determine backup procedures for the system. System should be able to backup information and restore information in an acceptable amount of time. This should have minimal impact on the availability of the system.

Concurrency - Consider which hardware features will improve concurrency. Development - Design constraints on modules may occur when attempting to achieve availability.

Deployment - The deployment environment has a major impact on the availability and resilience of a system. Disaster recovery should be quick and have minimal effect on the system.

Operational - Identify recovery issues and provide a process for recovering the system.

Concerns: Planned downtime, issues that lead to unplanned downtime, disaster recovery.

Activities: Availability schedules, asses against requirements, rework the architecture.

Tactics: Log transactions, find appropriate recovery solutions, design for failure.

Pitfalls: Single point of failure, ineffective error detection, overestimation of resilience.

## Evolution

Desired Qualities: It is desired that the system be flexible in that it will be able to handle the changes that will come over time.

Applicability:

Context - Any part of the external environment that may evolve over time should be discussed.

Functional - Reflect the amount of evolution that is expected.

Information - Flexible model is need for any evolution that may happen in the information environment.

Concurrency - May impose some constraints on the concurrency structure.

Development - Evolution has the opportunity to impact the development of a system in a major way. Development needs to keep in mind the changes that can occur in the future.

Deployment - Minimal discussion is needed between this perspective and viewpoint

Operational - Minimal discussion is needed here.

Concerns: Likelihood of change, timescale for change, when to pay.

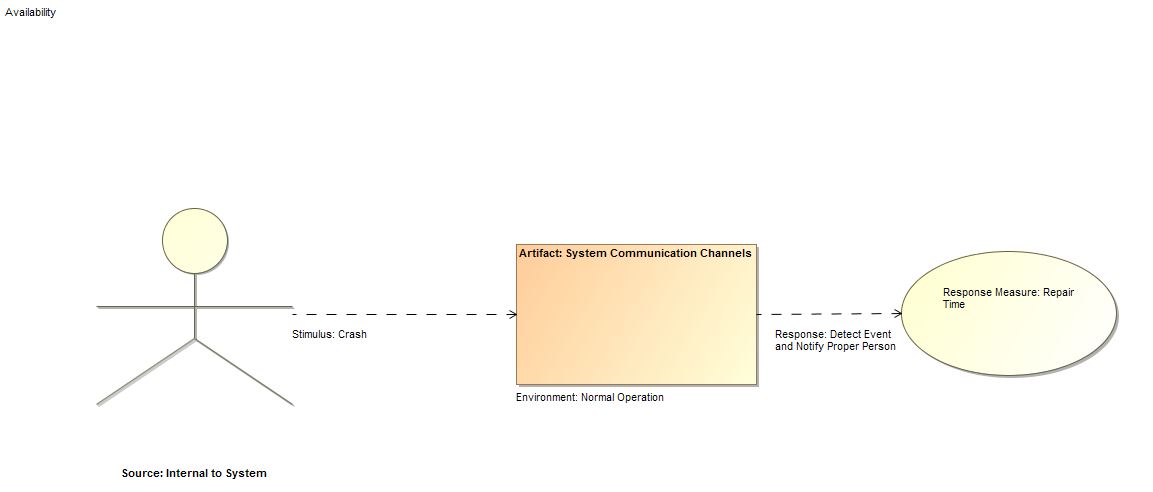
Activities: Characterize needs, apply tactics, consider tradeoffs, rework the architecture.

Tactics: Create interfaces, achieve reliable change

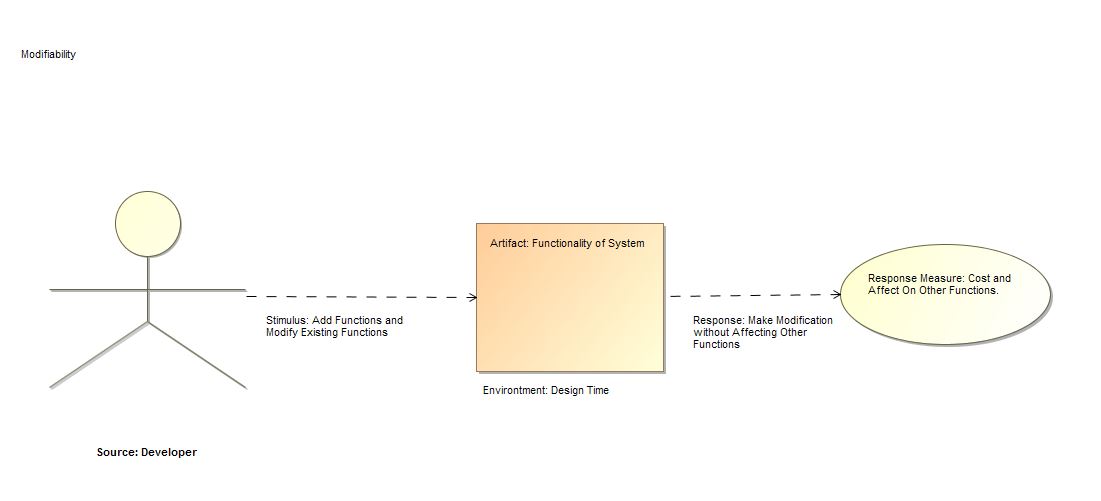
Pitfalls: Changes that never happen, lost development environments.

# Quality Scenarios

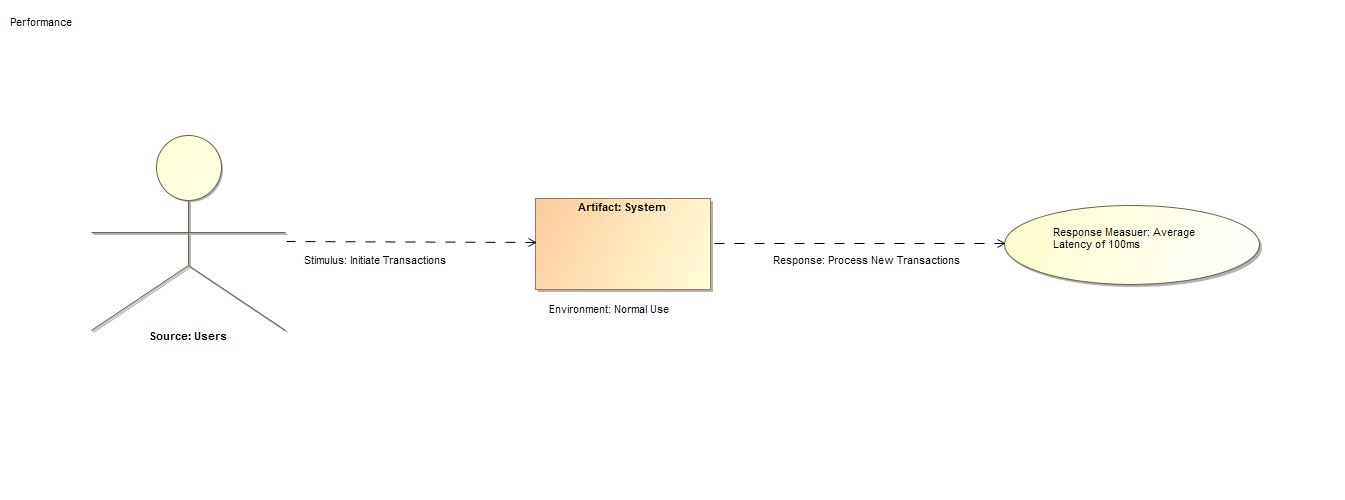
## Availability



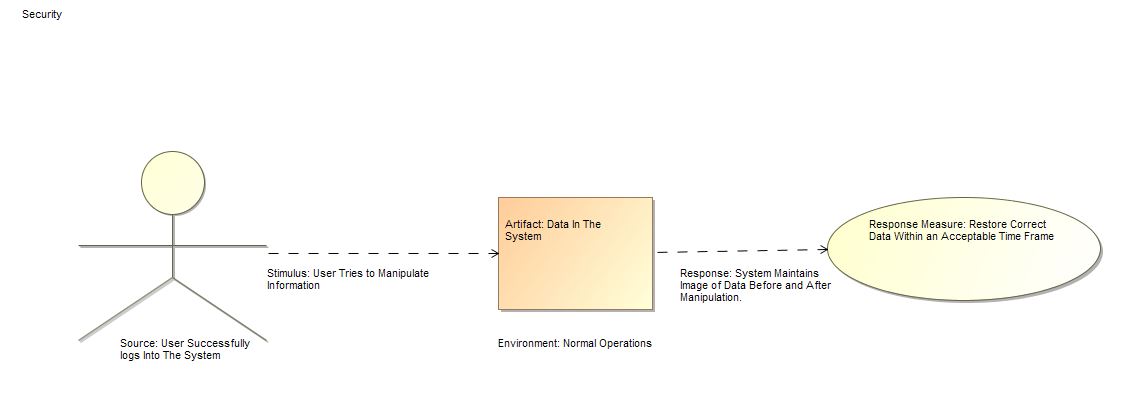
## Modifiability



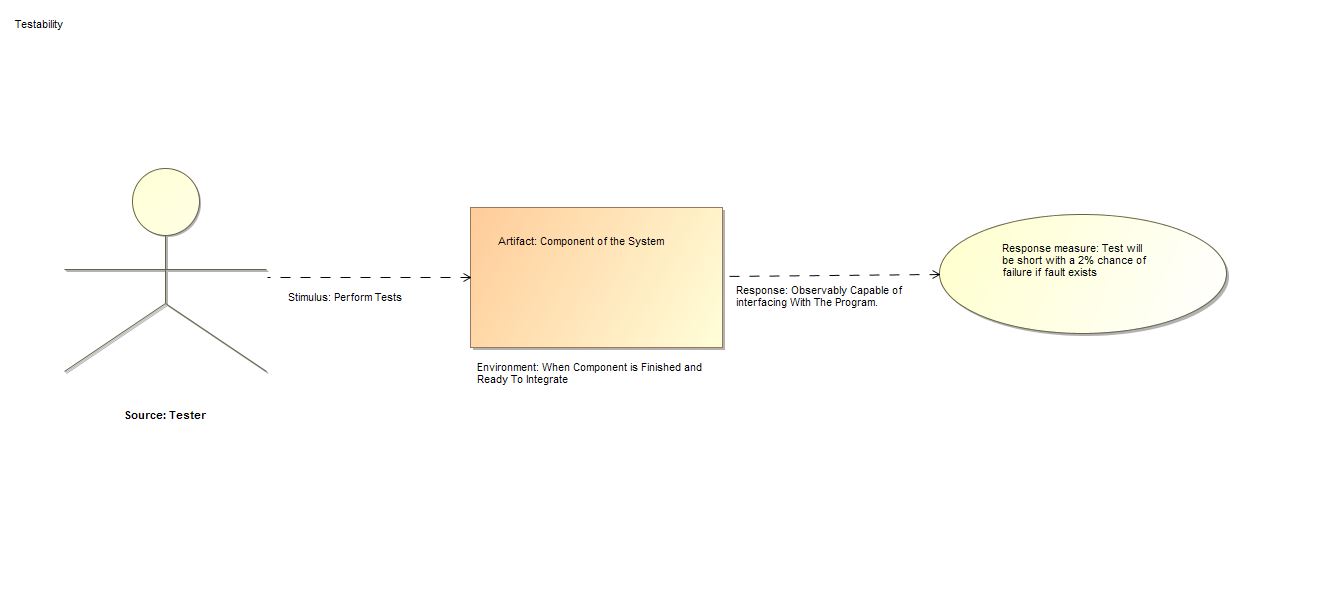
### Performance



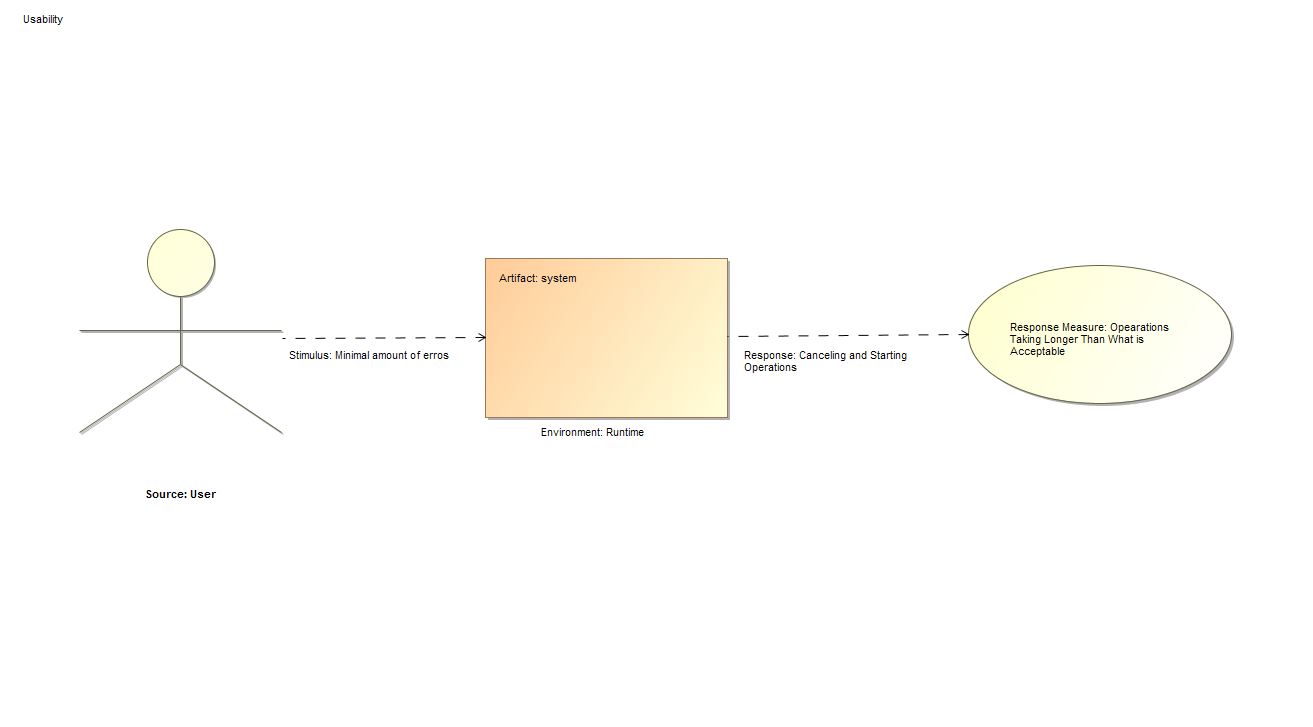
## Security



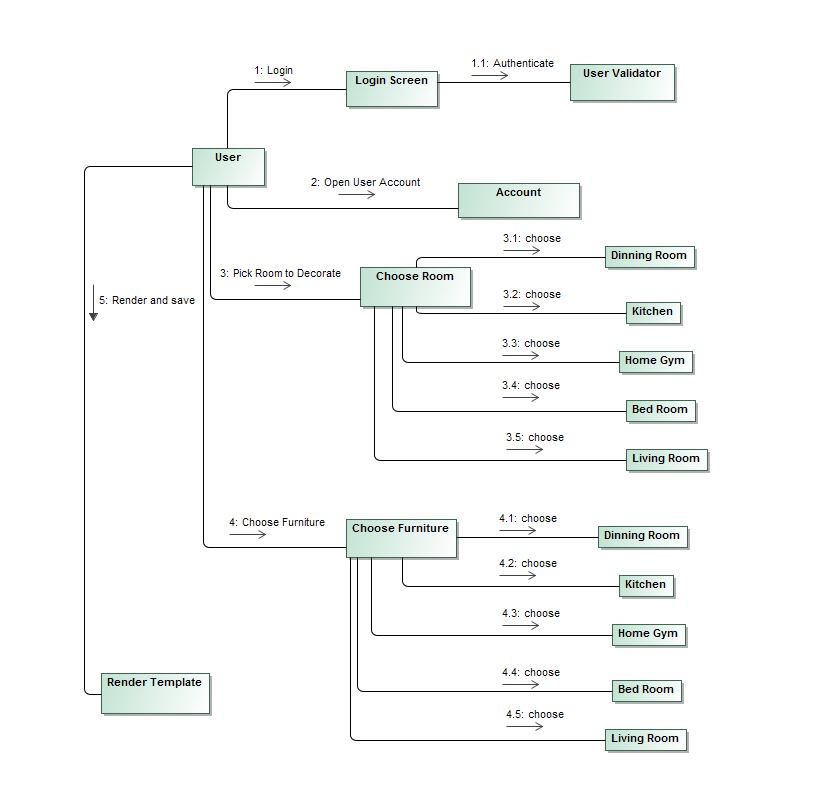
## Testability



## Usability



Collaboration Diagram



# Evaluation

## ATAM

As we face the decision of choosing between two architectures we turn to ATAM and CBAM to make an assessment of both.

As one can imagine a traditional approach will favor security and testability while with an SPA approach we would lean towards performance and usability.

### 1.3. “Traditional” web application approach

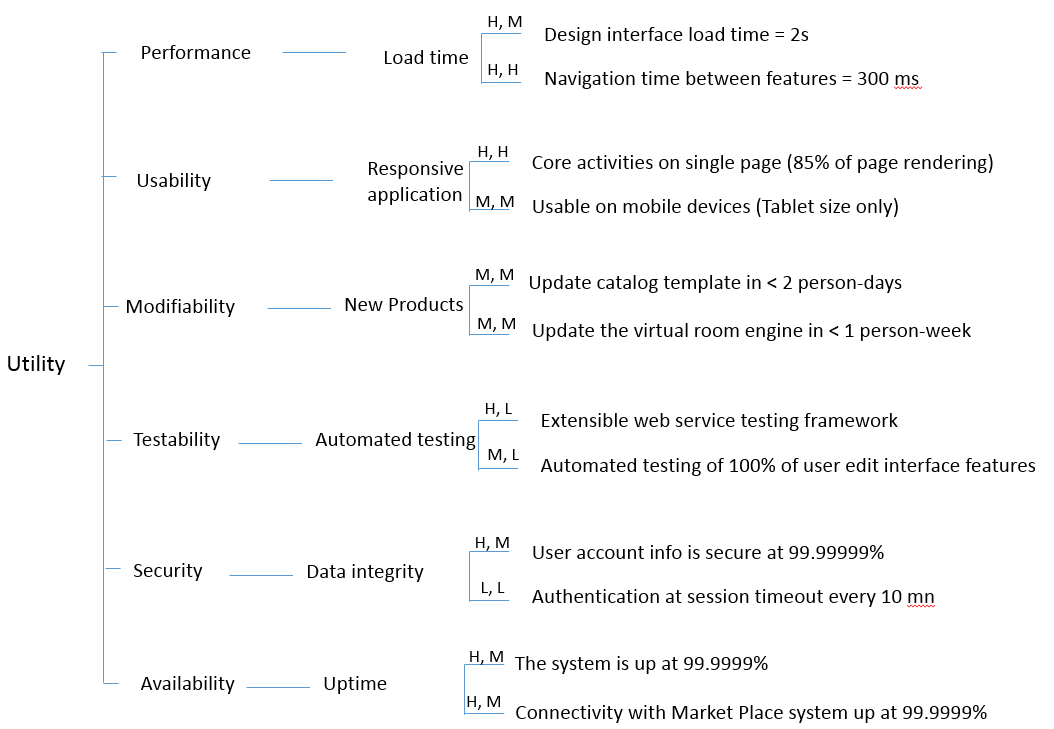


Figure VI.1. Utility tree of the traditional approach

### 1.2. SPA Approach

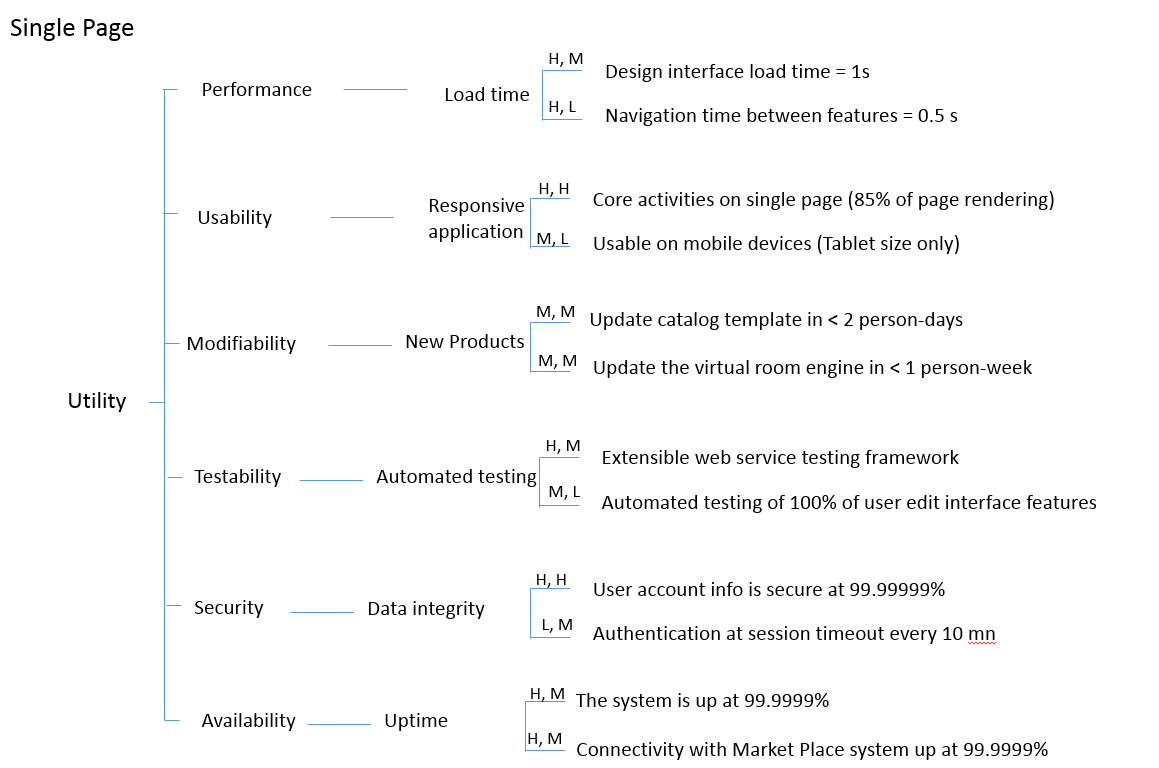


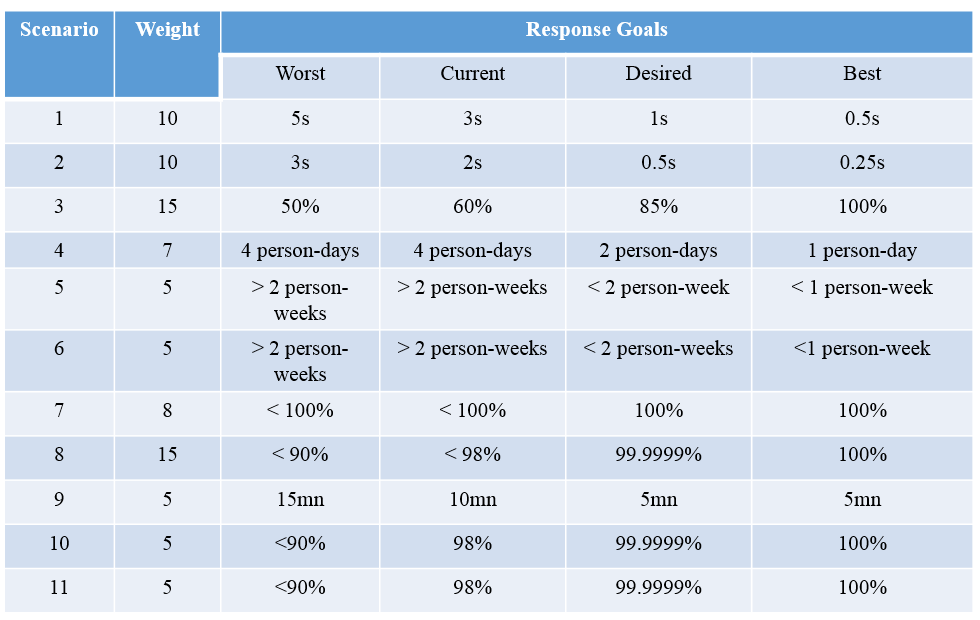
Figure VI.2. Utility tree of the Single Page approach

Mapping Table

|  |  |
| --- | --- |
| Number | Scenario |
| 1 | Load design interface in less than 1s |
| 2 | Navigation time between core features 0.5s |
| 3 | Core features on single navigation page (85% of rendering) |
| 4 | Update template catalog in less than 2 person-day |
| 5 | Update the virtual room engine in less than 1 person-day |
| 6 | Extensible web service testing framework |
| 7 | 100% of UI testing |
| 8 | User account info secure at 99.99999% |
| 9 | Authentication at session timeout after 5mn of inactivity |
| 10 | The system is up at 99.9999% |
| 11 | Connectivity with Market Place system up at 99.9999% |

## CBAM

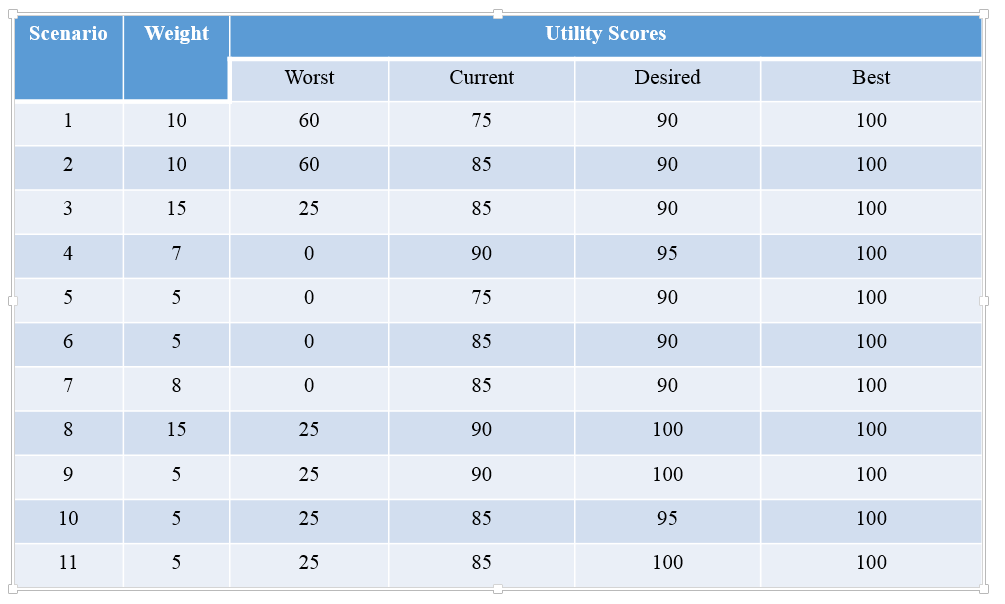
### Evaluate system response goals



At this step:

* Export scenarios from ATAM
* Evaluate quantitatively the system response goals from the worst allowable case to the best possible one.

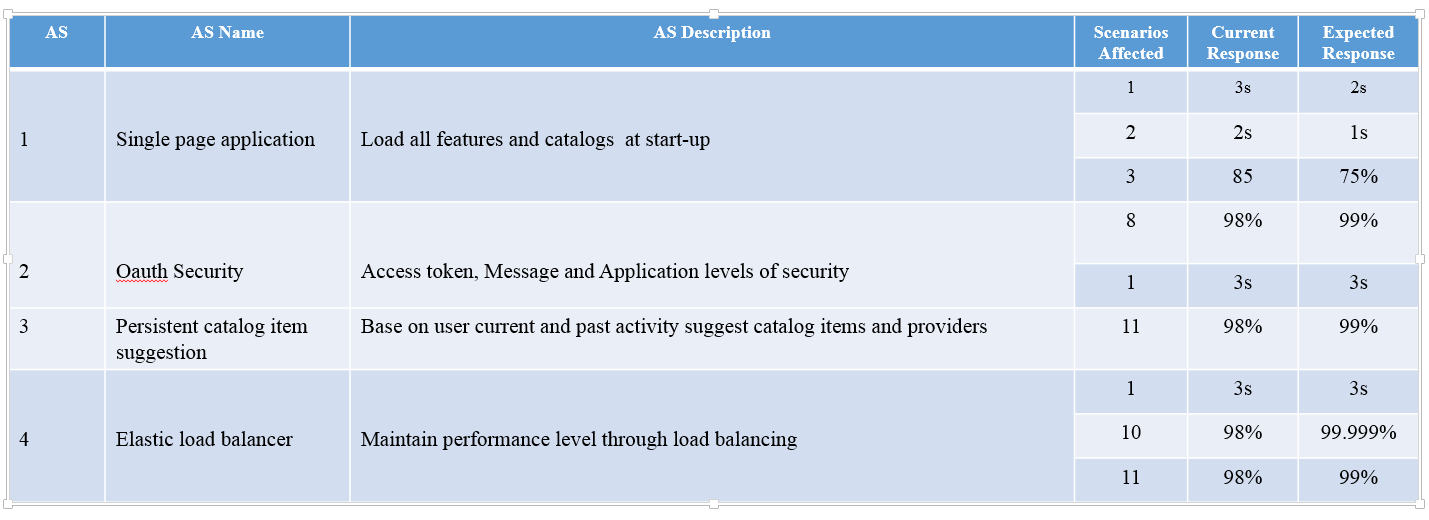
### Assign utility values to quality scenarios



At this step:

* Assign utility to each response goal according to what business value it has to a potential user
* Assign weight (importance) to each scenario.

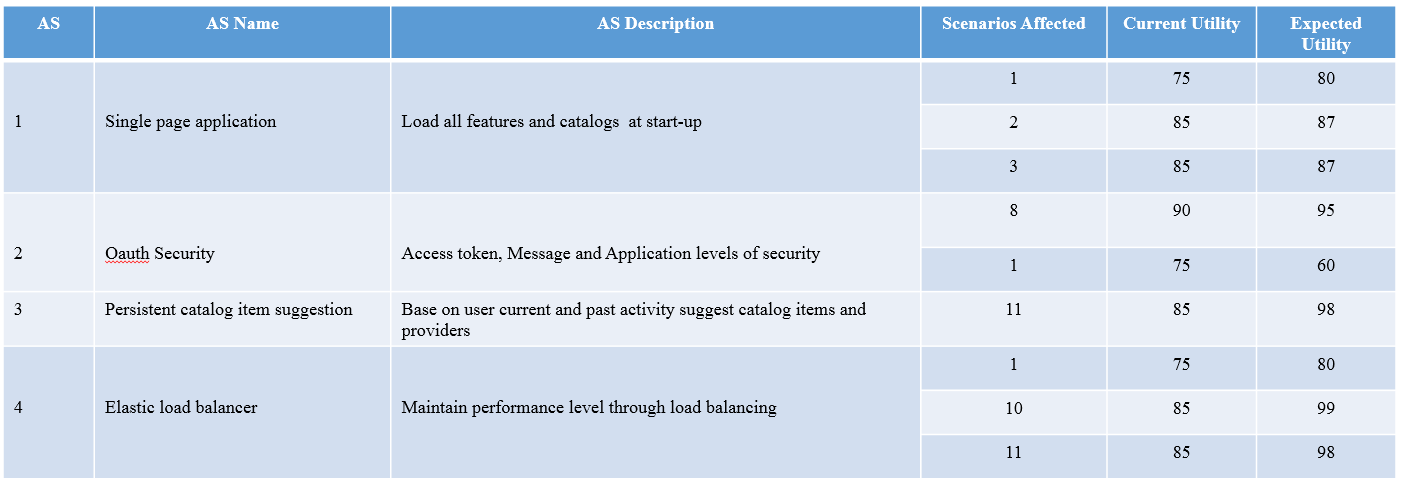
### Develop Architectural Strategies for Scenarios and determine expected response level



At this step:

* Determine architectural strategies and evaluate which scenario each of them affects
* In our example, the current column represents is the “traditional” approach and the expected column represents the SPA approach.

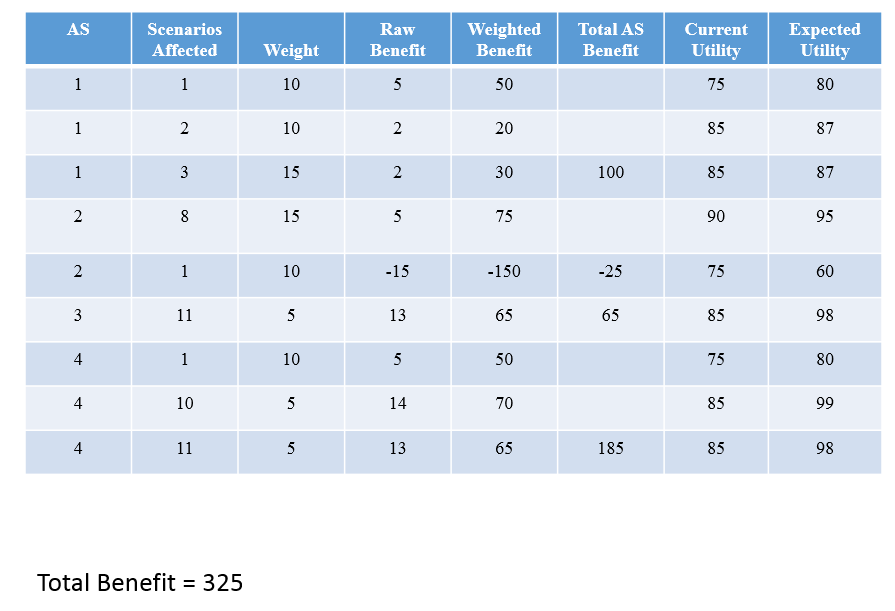
### Determine utility levels by interpolation



At this step:

* Because the expected response might not reach the desired level, we need to extrapolate to find how much utility the expected response brings us to.

### Determine total AS Benefits



To calculate the total benefit

Sum (weight\*(Expected utility – Current utility)).

There is a last step that involves calculating the cost of implementing the strategies and thus compute the ROI to evaluate the most beneficial strategies, but within the scope of this project we will not present that step.

# Conclusion

In this document we have presented the steps and methodologies necessary the architecture definition of a Home Furniture Application. We analyzed applying a SPA strategy to it and validated it with ATAM and CBAM over a more “traditional” way of designing web applications.

The limitations of this architecture definition are due to our lack of domain knowledge in web application development in general and in SPA’s in particular.