Corso di Progettazione di Applicazioni Web e Mobile

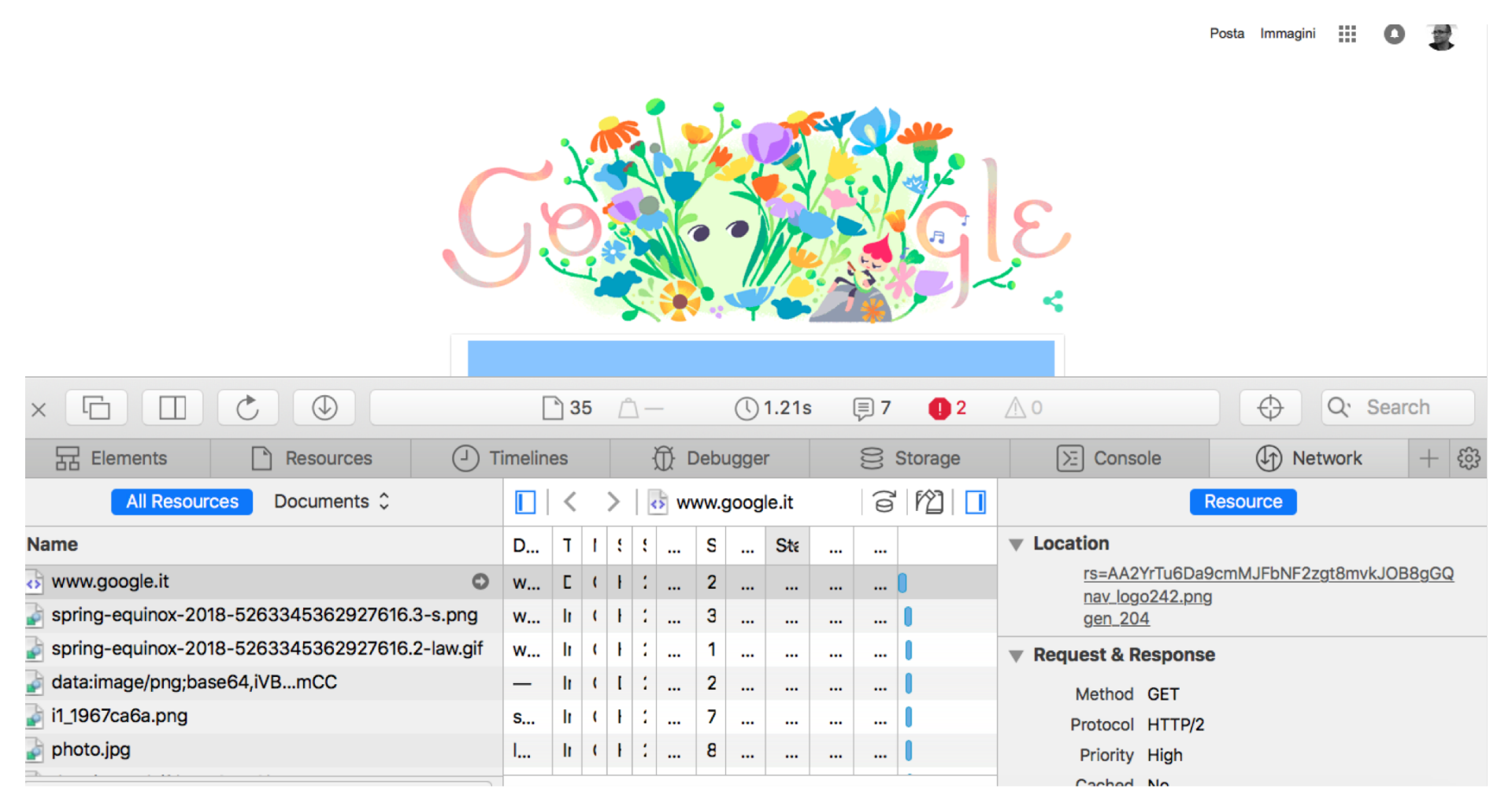
Mirko Calvaresi

What this is about?

How a web appliaction works?

let’s make and example: google.com

What happens when we type on a browser “www.google.com”?


Posta
Immagini
35
1.21s
7
2
0
Search
Elements
Resources
Timelines
Debugger
Storage
Console
Network
All Resources
Documents
www.google.it
Resource
Name
www.google.it
spring-equinox-2018-5263345362927616.3-s.png
spring-equinox-2018-5263345362927616.2-law.gif
...mCC
i1_1967ca6a.png
photo.jpg
Location
rs=AA2YrTu6Da9cmMJFbNF2zgt8mvkJOB8gGQ
nav_logo242.png
gen_204
Request & Response
Method GET
Protocol HTTP/2
Priority High

HTTP REQUEST AND RESPONSE

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
```

```
HTTP/1.1 200 OK
Content-Length: 243
Content-Type: text/html

...
```

DATA SENT WITH HTTP REQUEST

1. http method
2. url
3. http headers
4. http body
5. accept header
6. (content negotiation)
7. user agent

https://developer.mozilla.org/en-US/docs/Web/HTTP

MANAGE A STATE IN HTTP REQUEST

http://www.ietf.org/rfc/rfc2965.txt

Set-Cookie: Set-Cookie: sessionid=38afes7a8; HttpOnly; Path=/

https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

THE ENTIRE STACK

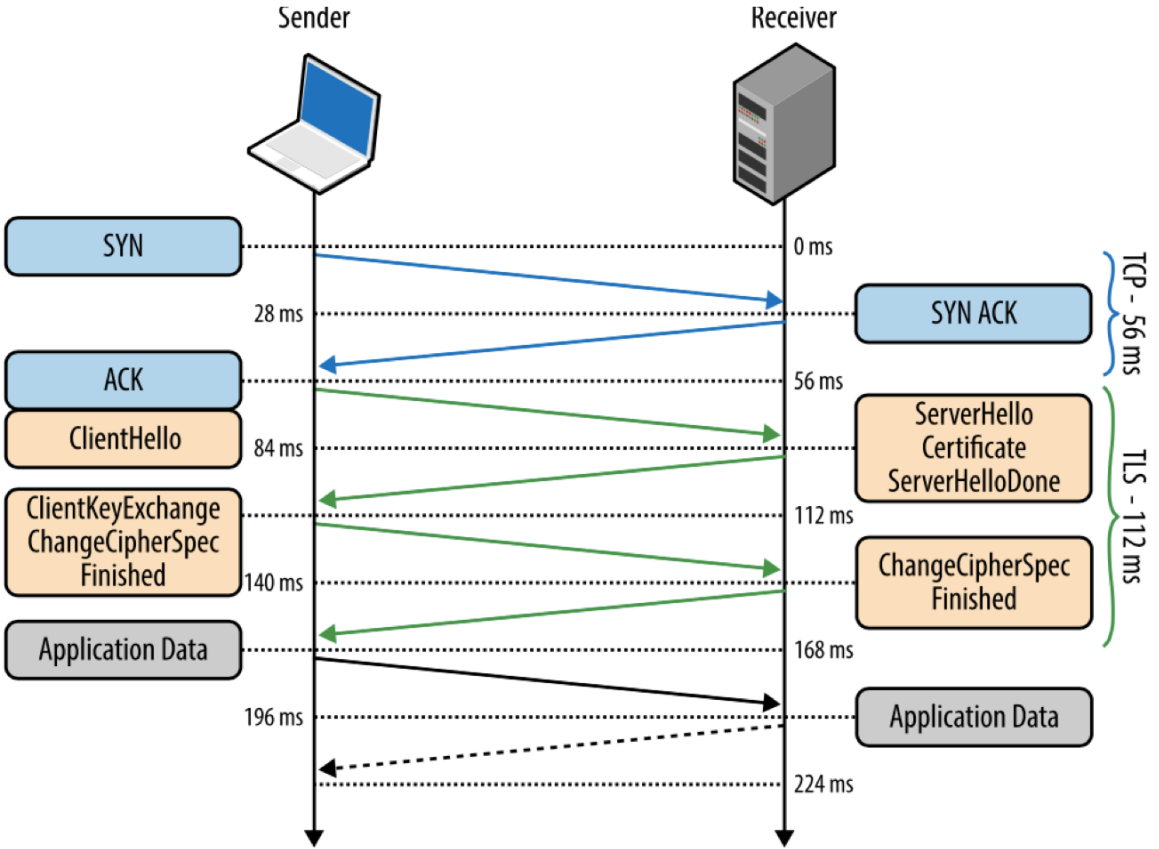
1. Key down, key up, key press events
2. DNS lookup of the url, resolve CNAME after CNAME until you get to an A record
3. open socket to ip
4. ethernet wraps tcp wraps http
5. tcp slow-start handshake
6. https handshake
7. client verifies trust chain
8. http method, headers, accept, hostname
9. server parses request and forms response
10. browser parses response, loads DOM
11. browser stops at <script>, <link>, <style>
12. onload, render
13. writes win32 messages to paint the screen

HTTP/2

http://httpwg.org/specs/rfc7540.html

HTTP/2 provides an optimized transport for HTTP semantics. HTTP/2 supports all of the core features of HTTP/1.1 but aims to be more efficient in several ways.
The basic protocol unit in HTTP/2 is a frame (Section 4.1). Each frame type serves a different purpose. For example, HEADERS and DATA frames form the basis of HTTP requests and responses (Section 8.1); other frame types like SETTINGS, WINDOW_UPDATE, and PUSH_PROMISE are used in support of other HTTP/2 features.
Multiplexing of requests is achieved by having each HTTP request/response exchange associated with its own stream. Streams are largely independent of each other, so a blocked or stalled request or response does not prevent progress on other streams.

HTTP vs HTTPS



HTTP vs HTTPS

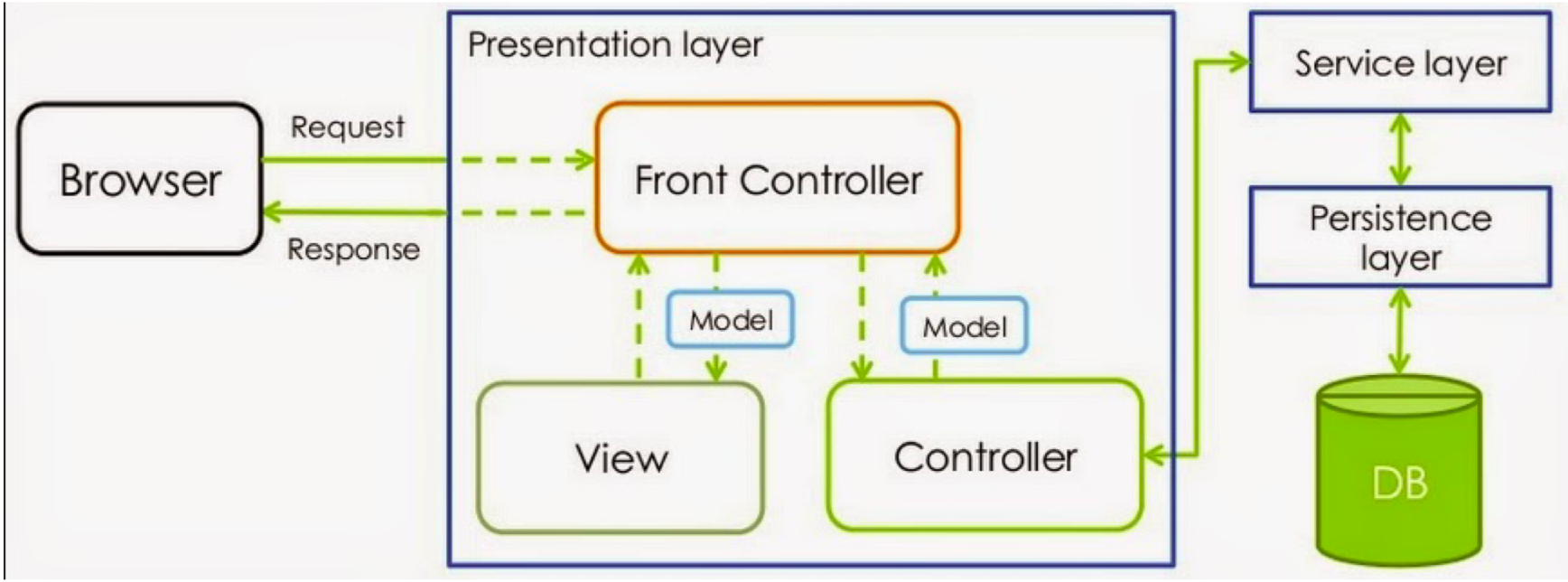
HTTPS protects the integrity of the website
Encryption prevents intruders from tampering with exchanged data—e.g. rewriting content, injecting unwanted and malicious content, and so on.

HTTPS protects the privacy and security of the user
Encryption prevents intruders from listening in on the exchanged data. Each unprotected request can reveal sensitive information about the user, and when such data is aggregated across many sessions, can be used to de-anonymize their identities and reveal other sensitive information. All browsing activity, as far as the user is concerned, should be considered private and sensitive.

HTTPS enables features on the web
The security and integrity guarantees provided by HTTPS are critical components for delivering a secure user permission workflow and protecting their preferences.

https://hpbn.co/transport-layer-security-tls/

ANATOMY OF WEB APPLICATION



PRESENTATION LAYER

The presentation layer is where the data is formatted and presented to the user .
In a classic scenario this corresponds to the page flow, including the controller activity, form validation, and user feedback.

Involves the HTML but also the backend component that controls the flow.

SERVICE LAYER

The service layer is where
the business logic of the application is implemented
data is analyzed
business rules applied
integration with other system invoked.

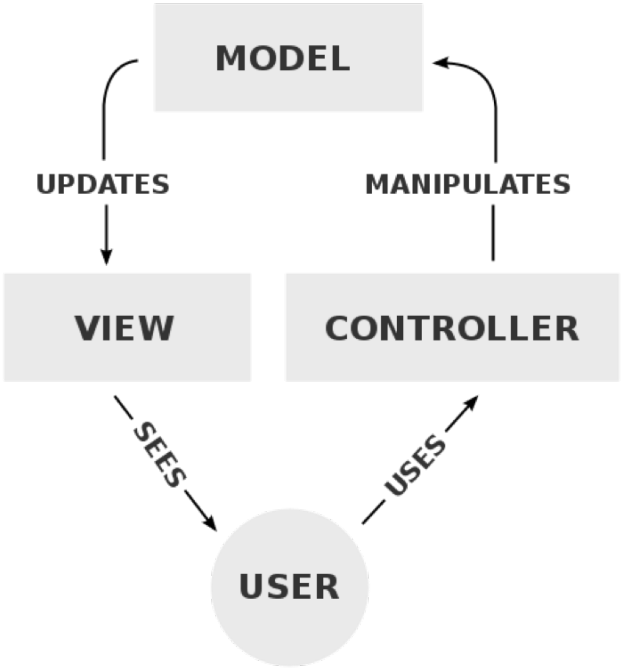
PERSISTENCE LAYER

The persistence layer is where the data is simply saved or retrieved from:

relational database
noSQL database
File System

MODEL VIEW CONTROLLER

Model–view–controller (MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.[1][2] The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development (Wikipedia)



AN EXAMPLE OF MVC CONTROLLER APPLICATION

https://github.com/spring-projects/spring-petclinic

```
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
./mvnw spring-boot:run
```

AN EXAMPLE OF MVC CONTROLLER APPLICATION

With the evolution of the WEB Browser and the introduction of Javascript Framework more and more application logic has been moved to the front end

At the base of this approach is the possibility to execute web request from the browser using Javascript – AJAX

SINGLE PAGE APPLICATION

A web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.
Wikipedia

Single page:
– interaction without page reloading
– data dynamically loaded from Server Side
– fluid transitions
– intensive use of javascript
They typically do not have
– support for crawlers
– support for legacy browsers

SINGLE PAGE APPLICATION

The single page application philosopy was made possible for the introduction to at least 3 major component:

1 JSON based SERVICES (https://tools.ietf.org/html/rfc8259)
2 HTML5
3 Evolution of javascript framework

```
var invocation = new XMLHttpRequest();
var url = 'http://bar.other/resources/public-data/';

function callOtherDomain() {
  if(invocation) {
    invocation.open('GET', url, true);
    invocation.onreadystatechange = handler;
    invocation.send();
  }
}
```

SINGLE PAGE APPLICATION Example

https://github.com/intojs/architecting-single-page-applications

https://codepen.io/Big_Brosh/pen/GObWow?page=1&

https://codepen.io/tag/reactjs/

REQUIREMENTS

It is unreasonable to expect that business stakeholders in a potential solution can articulate a set of complete, fully- developed consistent requirements through part-time involvement in a few requirements gathering exercises.

- Ignoring some of the components of a complete solution will not make them go away or reduce their needs
- **Complete solution view allows expectations to be managed**
- Requirements never capture the detail of the complete solution
- Requirements are just one set of constraints imposed on the solution design

REQUIREMENTS

Any Complete Solution Consists of:

- Zero or more of {Changes to Existing Systems}
- Zero or more of {New Custom Developed Applications}
- Zero or more of {Information Storage Facilities}
- Zero or more of {Acquired and Customized Software Products
- Zero or more of {System Integrations/Data Transfers/Exchanges}
- Zero or more of {Changes to Existing Business Processes}
- Zero or more of {New Business Processes}
- Zero or more of {Organizational Changes}
- Zero or more of {Reporting and Analysis Facilities}
- Zero or more of {Existing Data Conversions/Migrations}
- …

REQUIREMENTS

Functional requirements
Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

Non-functional requirements
constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

Domain requirements
Requirements that come from the application domain of the system and that reflect characteristics of that domain

REQUIREMENTS

Functional requirements
Describe functionality or system services

Depend on the type of software, expected users and the type of system where the software is used

Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail

Examples of functional requirements
The user shall be able to search either all of the initial set of databases or select a subset from it.

The system shall provide appropriate viewers for the user to read documents in the document store.

REQUIREMENT

Non Functional requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

Non functional requirements are very specific to the architecture and play an important role for the effective result and success of the entire application.

NON FUNCTIONAL REQUIREMENT

Accessibility
Availability (see service level agreement)
Backup
Capacity, current and forecast
Certification
Compliance
Configuration management
Dependency on other parties
Documentation
Disaster recovery
Efficiency (resource consumption for given load)
Effectiveness (resulting performance in relation to effort)
Emotional factors (like fun or absorbing or has "Wow! Factor")
Environmental protection
Escrow
Exploitability
Extensibility (Failure management
Fault tolerance (e.g. Operational System Monitoring, Measuring, and Management)
Legal and licensing issues or patent-infringement-avoidability
Interoperability
Maintainability
Modifiability
Network topology
Operability
Open source
Performance / response time (performance engineering)
Platform compatibility
Price
Privacy
Portability
Quality
Recovery / recoverability (e.g. mean time to recovery - MTTR)
Reliability (e.g. mean time between failures - MTBF, or availability
Reporting
Resilience
Resource constraints (processor speed, memory, disk space, network bandwidth, etc.)
Response time
Reusability
Robustness
Safety or Factor of safety
Scalability (horizontal, vertical)
Security
Software, tools, standards etc. Compatibility
Stability
Supportability
Testability
Transparency
Usability by target user community

REQUIREMENT DELIVERABLES

ISO/IEC/IEEE 29148:2011 Systems and Software Engineering – Life Cycle Processes – Requirements Engineering
Proposes five key deliverables:

1. Stakeholder Requirements Specification (StRS) Document
2. System Requirements Specification (SyRS) Document
3. Software Requirements Specification (SRS) Document
4. System Operational Concept (OpsCon) Document
5. Concept of Operations (ConOps) Document

http://www.iso.org/iso/catalogue_detail.htm?csnumber=45171

REQUIREMENT DELIVERABLES

1. Stakeholder Requirements Specification (StRS) Document

- Business Purpose
- Business Scope
- Business Overview
- Stakeholders
- Business Environment
- Goal And Objective
- Business Model
- Information Environment
- Business Processes

REQUIREMENT DELIVERABLES

2.System Requirements Specification (SyRS) Document

- System Scope
- System Overview
- System Context
- System Functions
- User Characteristics
- Functional Requirements
- Usability Requirements
- Performance Requirements
- System Interfaces
- System Operations
- **...**

REQUIREMENT DELIVERABLES

3. Software Requirements Specification (SRS) Document

- Product Perspective
- System Interfaces
- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communications Interfaces
- Memory Constraints – Operations
- Site Adaptation Requirements
- Product Functions
- Product Functions

REQUIREMENT DELIVERABLES

4. System Operational Concept (OpsCon) Document

- System Overview
- Referenced Documents
- Current System Or Situation
- Background, Objectives, And Scope
- Profiles Of User Classes
- Interactions Among User Classes
- Concepts For The Proposed System – Background, Objectives, And Scope – Operational Policies And Constraints – Description Of The Proposed System – Modes Of Operation – User Classes And Other Involved Personnel
- Organisational Structure

REQUIREMENT DELIVERABLES

5. Concept of Operations (ConOps) Document

- Strategic Plan
- Effectiveness
- Overall Operation – Context – Systems – Organisational Unit
- Governance – Governance Policies – Organisation – Investment Plan – Information Asset Management – Security – Business Continuity Plan

REQUIREMENT TYPES

User requirements
Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers

System requirements
A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor

Software specification
A detailed software description which can serve as a basis for a design or implementation. Written for developers

REQUIREMENT GATHERING

Requirements can be gathered in different ways:

- Structured brainstorming workshops
- Interviews and questionnaires
- Technical, operational, and/or strategy documentation review
- Simulations and visualizations
- Prototyping
- Modeling
- Quality function deployment (QFD)
- Use case diagrams (OMG 2010)
- Activity diagrams (OMG 2010)
- Functional flow