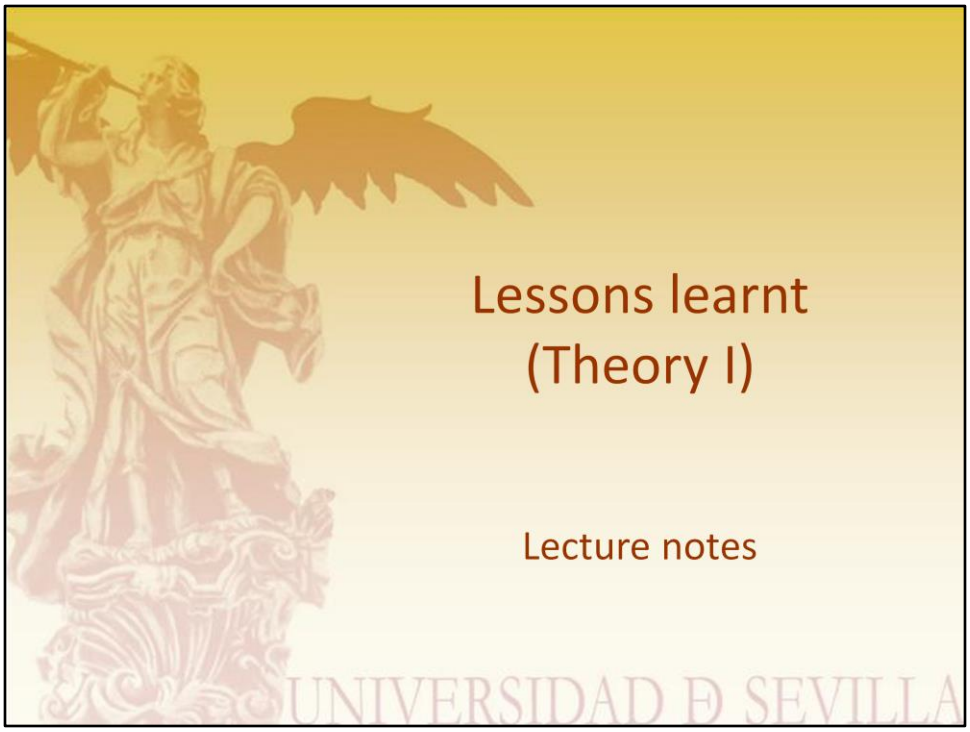



Welcome to this lecture! Today and the next day, we're going to recap on a few lessons that we've learnt during the first semester.

--

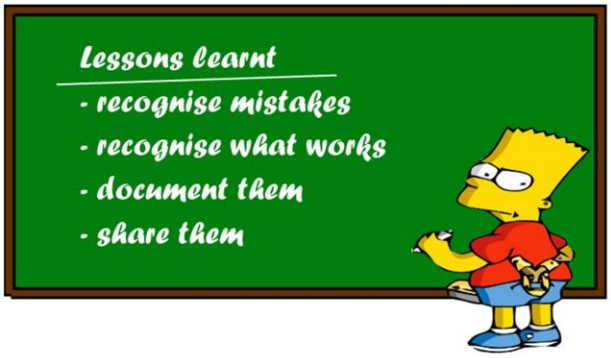
## What's a lesson learnt?

UNIVERSIDAD Ð SEVILLA

2

As usual, we start with a question: what's a lesson learnt? Don't cast a glance at the following slides; please, make a point of producing your own answer.

## This is a good definition



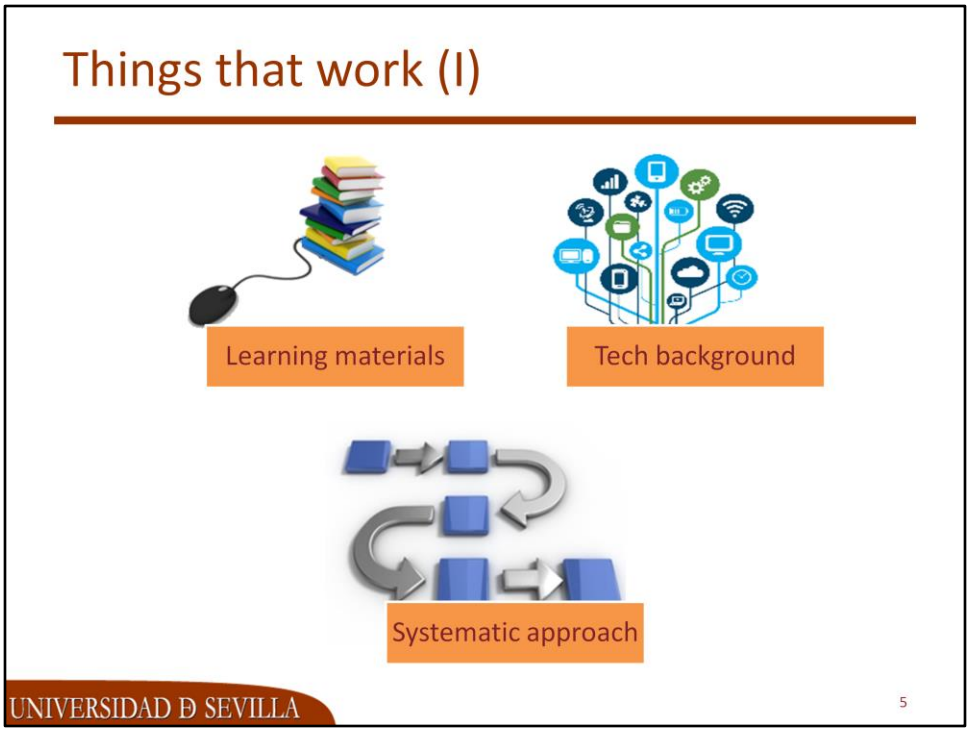

It's a lesson (good or bad) that we've learnt from our experience, that we've documented, and shared with others

UNIVERSIDAD Ð SEVILLA

3

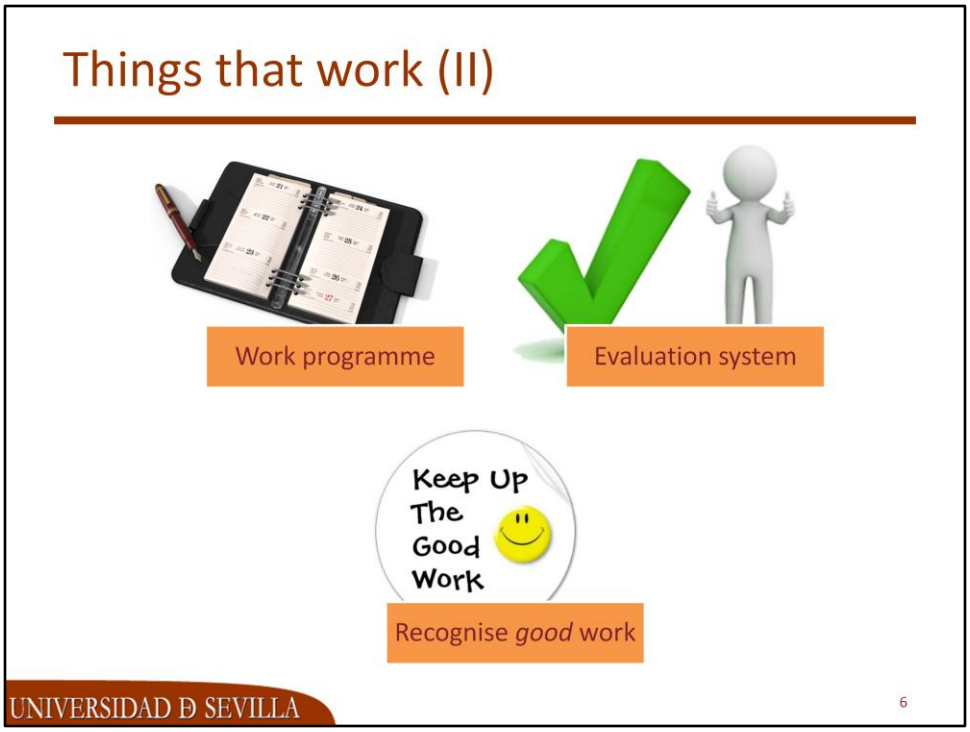
This is a good definition: a lesson learnt is a lesson (good or bad) that we've learnt from our experience, that we've documented, and shared with others. Lessons learnt are important insofar they prevent us and others from making the same mistakes and encourages us to follow proven paths.


It’s time to reflect a little on the lessons we’ve learnt during the first semester. Please, think of them before casting a glance at the following slides.



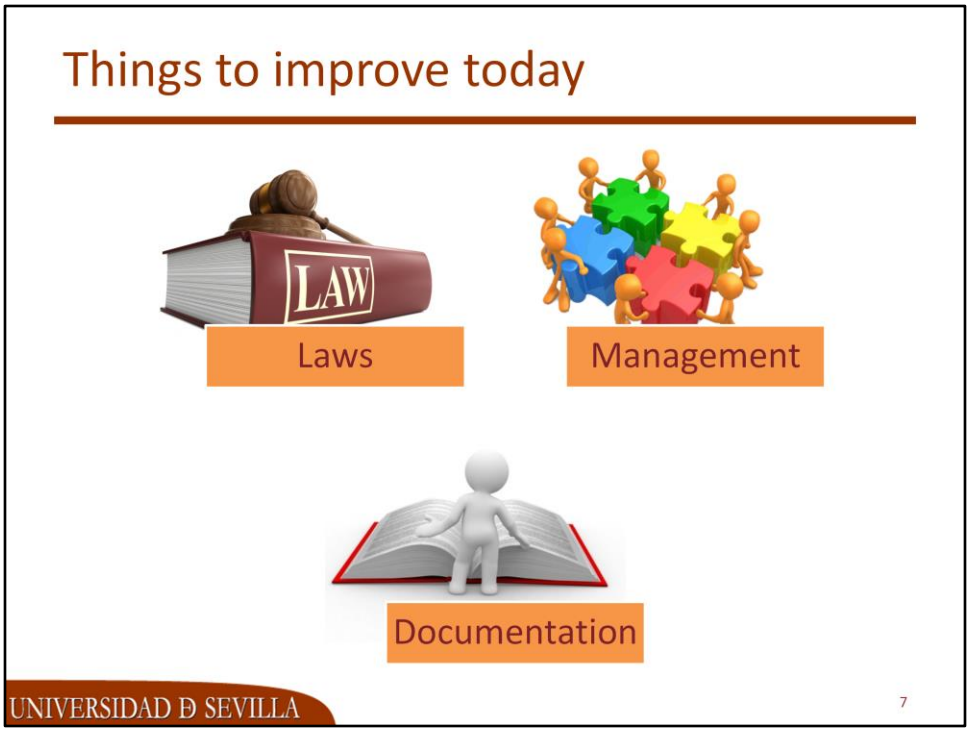

Let's start with the things that work:

- The learning materials are sound, that is: they are correct and complete. The students who have gone through the learning materials have learnt a lot.
- We've provided a good technical background on a number of technologies that are very common in our industry: Java, Spring, JSP, JPA, JQPL, MySQL, Tomcat, and so on.
- We've used quite a systematic approach that's good insofar it helps our students focus on learning, not on finding their way through a gymkhana.




Definitely, it also works:

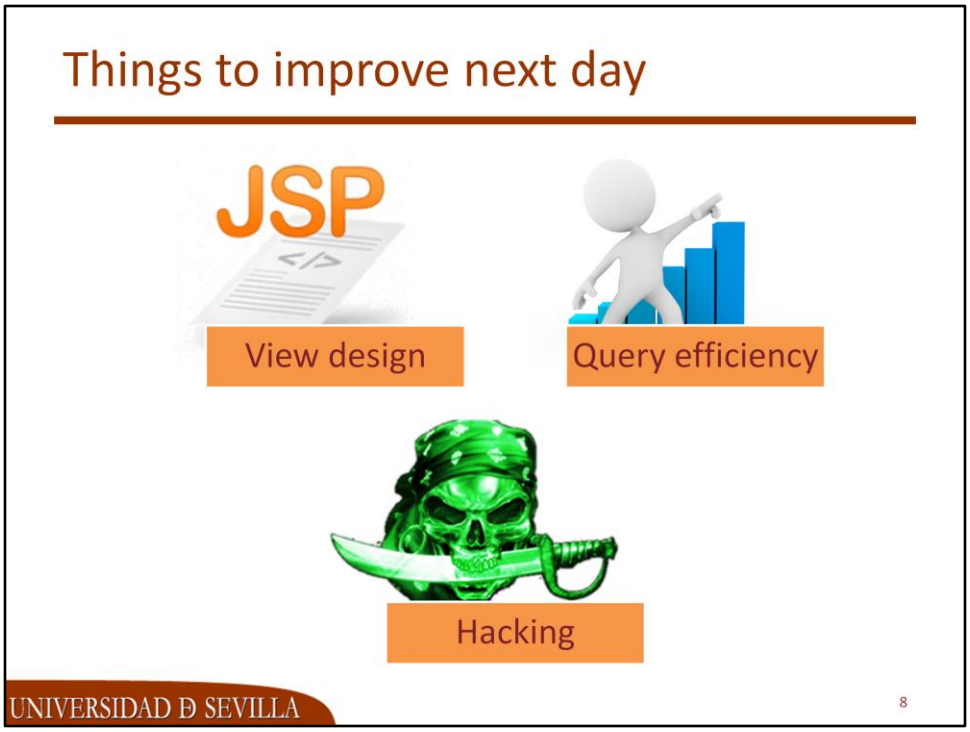
- The work programme. It's well-though; students who engaged our work programme could follow it.
- The evaluation system, which focuses on evaluating your work every now and then, and you know exactly what you have to do to pass your evaluations.
- We recognise good work and cheer A+ students up. We're very sorry if you decided not to be an A+ student. There's only a thing we suggest that you should do: don't trust people how say that you can't be an A+ student (even if that person is you). Many students before you have achieved and A+ in D&T. Will you be an exception?




But there are a number of things to improve, namely:

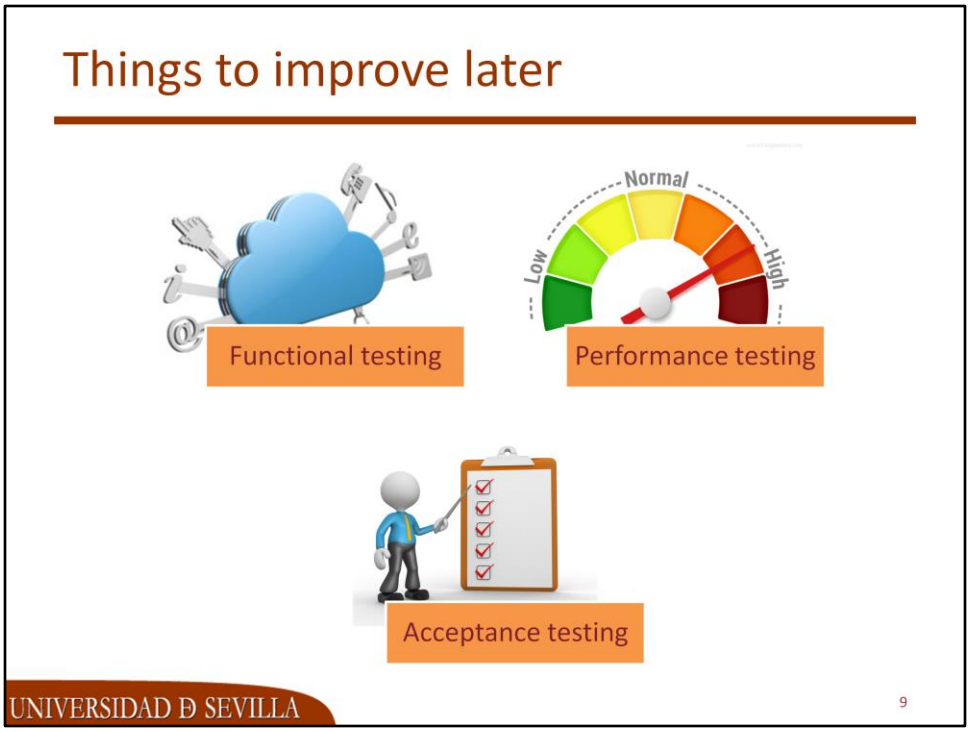
- Laws: laws are very important since web information systems typically handle personal data or offer electronic services and must then comply with current laws. We haven't studied enough about them so far. It's time to learn a little about laws.
- Management: you've been working on the Acme project for one semester, but we didn't put an emphasis on your managing it since our goals were purely technical. It's time to start thinking of managing your projects.
- Documentation: you haven't produced any documentation about your projects, but it's quintessential for maintenance. It's also time to start thinking of documenting your projects.

We'll explore these lessons learnt today.



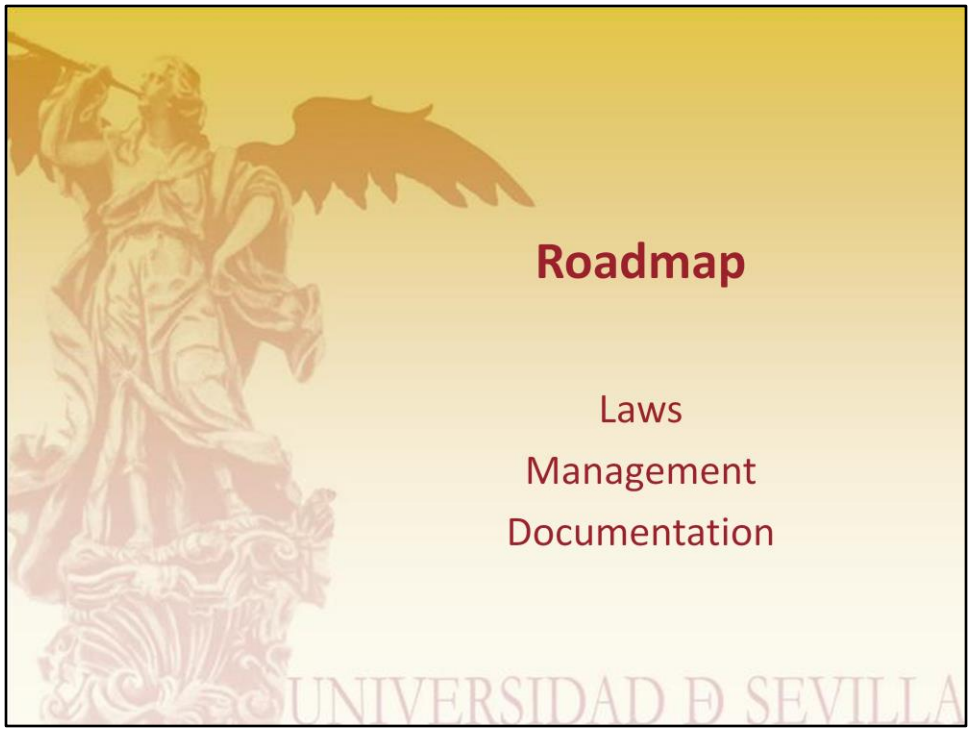

Next day, we'll explore the following lessons learnt:

- View design: we've learnt about the JSP technology, and we've designed wonderful views; but they are quite repetitive and error-prone. It's time to learn on some advanced tricks that will allow us design our views more easily.
- Query efficiency: so far we've designed queries, but we haven't worried about how efficient they are. It's time to study query efficiency issues.
- Hacking: neither have we worried a lot about hacking, but hackers do exist and they may easily become a nightmare. It's time to protect our systems against hackers as much as possible.

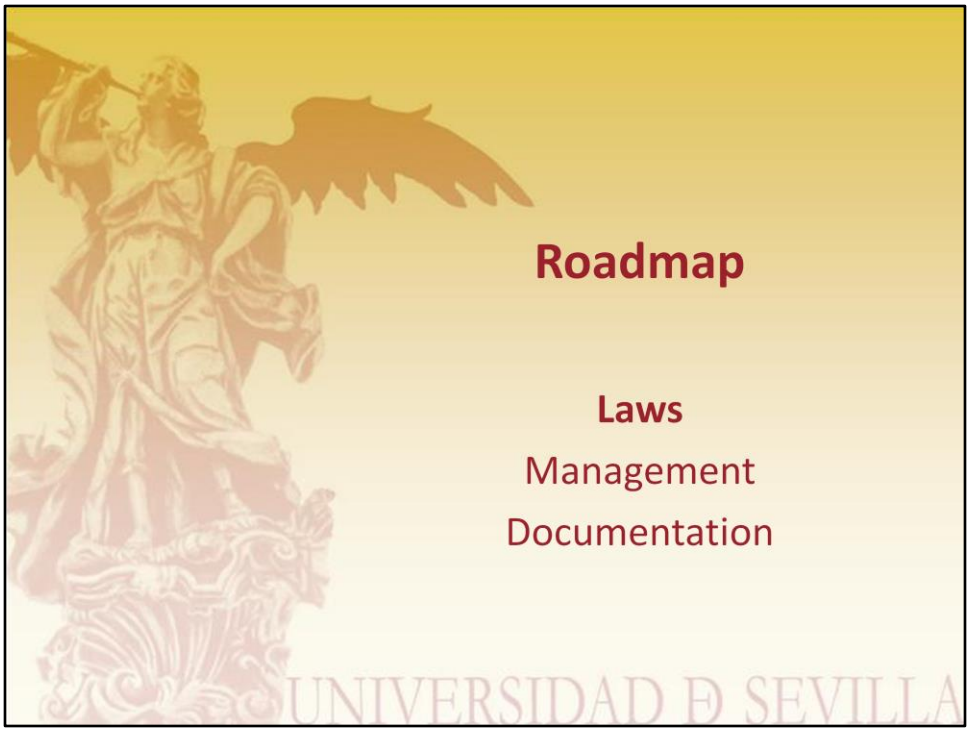



Later this semester, we have to improve regarding testing. Testing our web information systems will definitely help us improve their quality since we can early detect many defects that would otherwise go unnoticed into the production stage. Testing is typically classified as follows:

- Functional testing: it's intended to test methods and their relationships in domain classes, converters, repositories, services, controllers, and other components of our architecture.
- Performance testing: it's intended to test the performance of a system to predict the maximum workload that it can handle gracefully.
- Acceptance testing: it's intended to test things that can't be easily automated, that is, things that require a technician; this includes checking use cases using user interfaces, for instance.




This is our roadmap for today's lecture: we're going to report on laws, on project management, and documentation. In the next lecture, we'll explore view design, query efficiency, and hacking.




Let’s start with an insight into laws.

## What are laws?


The rules that a country uses to regulate the actions of its citizens or organisations

UNIVERSIDAD Ð SEVILLA

12

As usual, we start with a definition. Laws are commonly defined as the rules that a country uses to regulate the actions of its citizens or organisations. By citizen, we mean an individual who lives in a given country and has the corresponding nationality; by organisation, we mean a group of citizens who jointly perform some actions in a given country. Organisations range from student unions to companies.

## Why are they important to us?

They make it clear what a web information system is allowed to do and what it is not allowed to do at all

UNIVERSIDAD Ð SEVILLA

13

Laws are very important to us because they make it clear what a web information system is allowed to do and what it is not allowed to do at all.


In Spain, there are many applicable laws, but the most important are the Organic law 15/1999 (the LOPD), Law 34/2002 (the LSSI), and Royal Decree-Law 13/2012 (the transpositions).


It’s about time to provide a few details on these laws in the following slides.


Let's start with the LOPD.

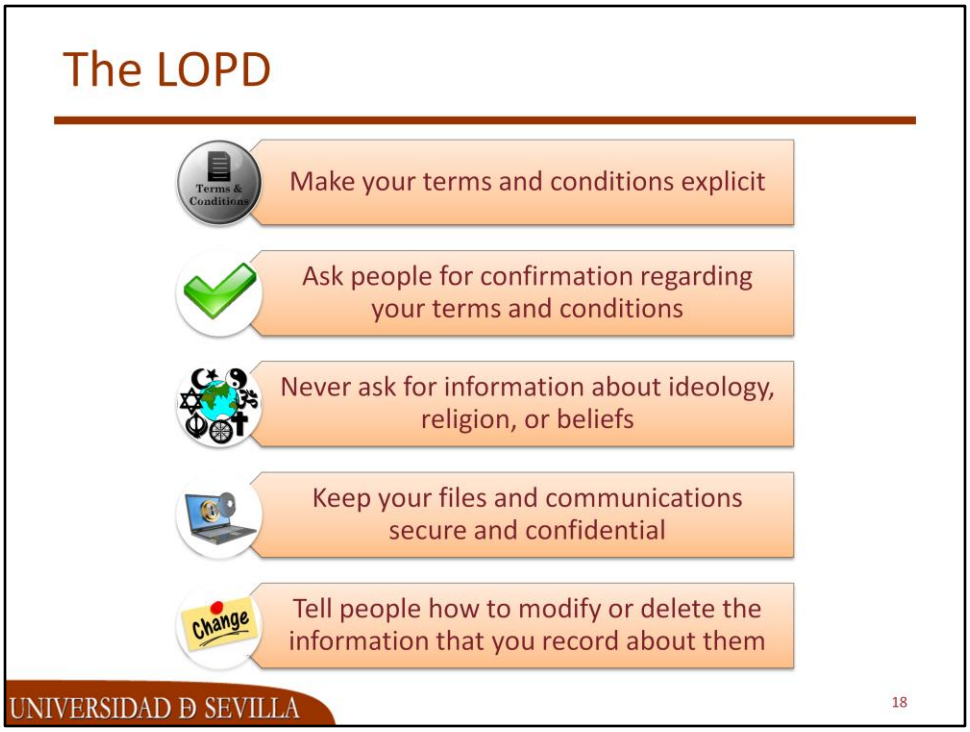
## Organic Law 15/1999: LOPD

It deals with protecting the data that an organisation has about people

UNIVERSIDAD D SEVILLA

17

Organic Law 15/1999 is also known as the LOPD, which is the Spanish acronym for "Ley Orgánica de Protección de Datos". This law deals with protecting the data that an organisation has about people.




It basically amounts to the following:

- Your web information systems must provide a page in which you make your terms and conditions explicit.
- You must ask people for confirmation regarding whether they accept your terms and conditions or not.
- Never ask for information about ideology, religion, or beliefs.
- Keep your files and communications secure and confidential.
- Tell people how to modify or delete the information that you record about them.

Apparently, it's not that difficult to comply with this law, is it? Let's do it in our next project!


Please, note that the LOPD is an ancient law and that it's going to be revoked in 2018. It's going to be replaced by a new EU-wide lay called the General Data Protection Regulation (GDPR). The good piece of news is that the GDPR is very similar to the LOPD, but homogeneous across Europe. Please, find more information about the new Lay at https://ec.europa.eu/info/law/law-topic/data-protection_en.


The next law is the LSSI. Let's provide a couple more details.

## Law 34/2002: LSSI

It deals with electronic services, that's why it's also known as "the internet law"

UNIVERSIDAD D SEVILLA

21

Law 34/2002 is also known as the LSSI, which is the Spanish acronym for "Ley de Servicios de la Sociedad de la Información". It deals exclusively with electronic services, which is the reason why it's also known as "the internet law".


Let's now summarise what you have to do to comply with the LSSI:

- You must inform the Chamber of Commerce (Registro Mercantil) about your internet domain.
- You must identify yourself in your web pages, including your legal data (company name, VAT number, and information about your registration in the Chamber of Commerce).
- Adhere to deontological applicable ethic codes, if any. Unfortunately, we don't know of any such code in computing. We're sorry!
- Whenever your system proposes a contract to a customer, ask him or her for confirmation before accepting it and record your customer's answer.


Let's finally delve into the Transpositions law.

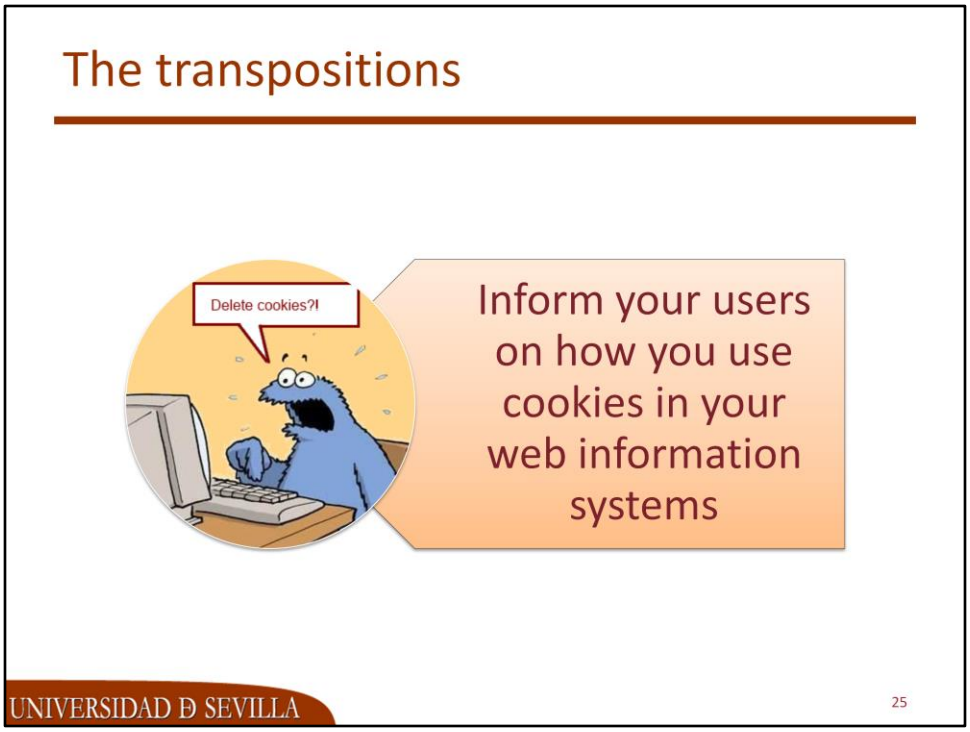
## R. Decree-Law 13/2012: transpositions

These are changes to a variety of older Spanish laws that weren't in accordance with some European Union laws

UNIVERSIDAD D SEVILLA

24

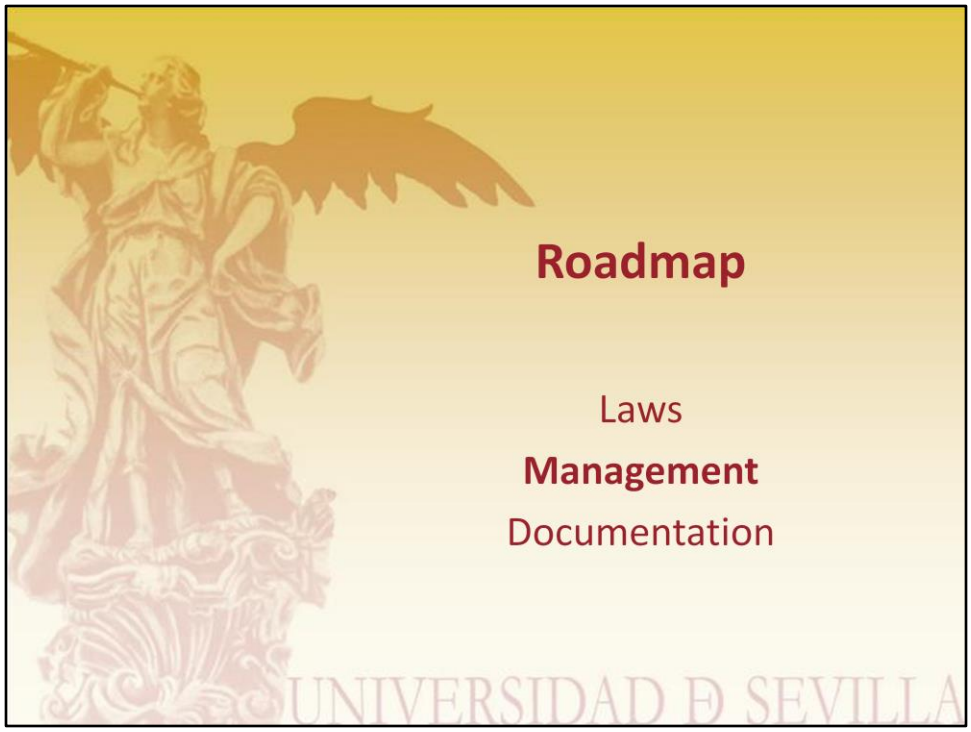
The transpositions law is formally referred to as Royal Decree-Law 13/2012. It deals with a number of changes to a variety of older Spanish laws that weren't in accordance with some European Union laws.




The only interesting thing here is that you must inform your users on how you use cookies in your web information systems. Note that we haven't explored using cookies so far, but our web information systems use two cookies by default, namely:

- "JSESSIONID": this is a cookie that is set automatically by Spring to keep your Apache's session status. (Please, forget this. Consider Apache's session status as something obsolete that you should never use in practice! Spring uses it, but we hope it doesn't use it in forthcoming releases.)
- "language": this cookie is set whenever you change the language in which your system must show its messages.

Don't forget to comment on them and make it clear what they are intended for.




Let's now report on something that is more software-engineering-oriented: let's talk about management.

## What's management?

It's about co-ordinating the work of a number of people to accomplish goals using the available resources efficiently and effectively

UNIVERSIDAD Ð SEVILLA

27

Management is about co-ordinating the work of a number of people to accomplish some goals using the available resources efficiently and effectively.

## Why is it important?

Because it's the key to success: no management implies no success

UNIVERSIDAD Ð SEVILLA

28

Management's very important because it's the key to success. A good management cannot guarantee success, but if there's no management at all, then failure is guaranteed. So far you have worked in small groups that can easily co-ordinate, but this isn't common in a professional setting. Typically, a dozen people have to work on a project, and that requires good management.


In the following sections, we'll provide an insight into the management that you require so that you can face the spring semester successfully. We'll start with a few general hints and then will present a clear guideline and some management tools.

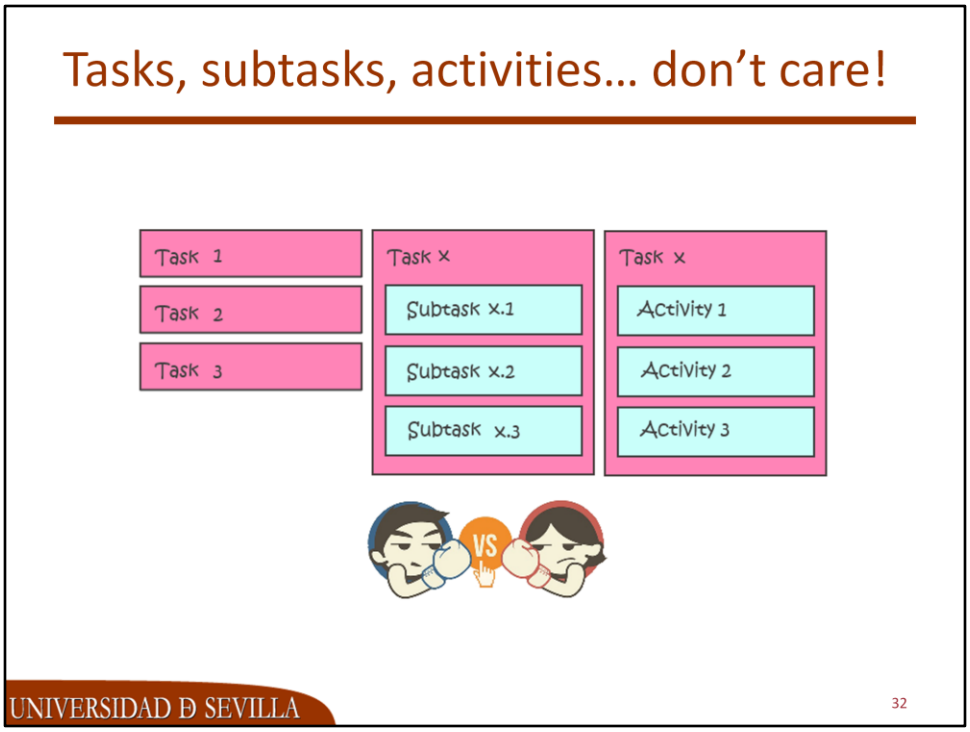

Let's start with some general hints.

## Use check lists extensively

UNIVERSIDAD Ð SEVILLA

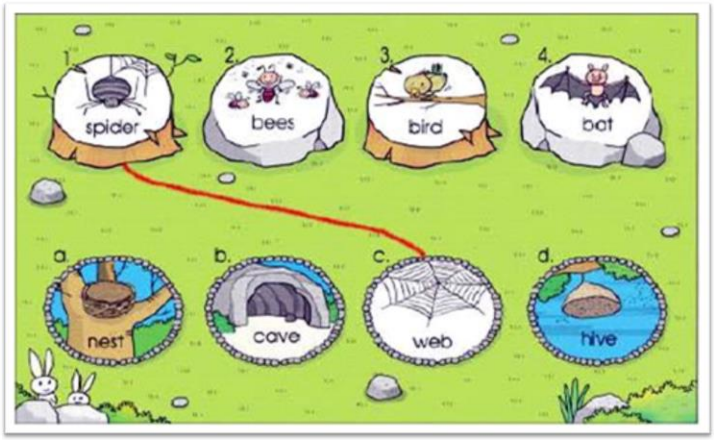
31

Use check lists extensively. Every time you do something for the first time, you have to discover how to do it. It's very important that you keep that knowledge so that you can re-use it in future. Check lists are a good tool to keep that knowledge at hand. Please, get accustomed to using check lists: how to prepare your planning? How to implement a new listing view? How to implement a new edition view? How to configure internationalisation? What to do to package your deliverable? How to check it? The previous questions and many others can be very easily addressed using check lists. Create them, update them, and you soon will have a comprehensive knowledge base that will allow you to work very efficiently.




Should you organise your project into tasks, tasks and subtasks, or tasks and activities? We don't care about that! Get familiar with the different approaches and decide on your own. By now, you should have enough experience to make a good decision.

## What matters: one task, one worker




UNIVERSIDAD Ð SEVILLA

33

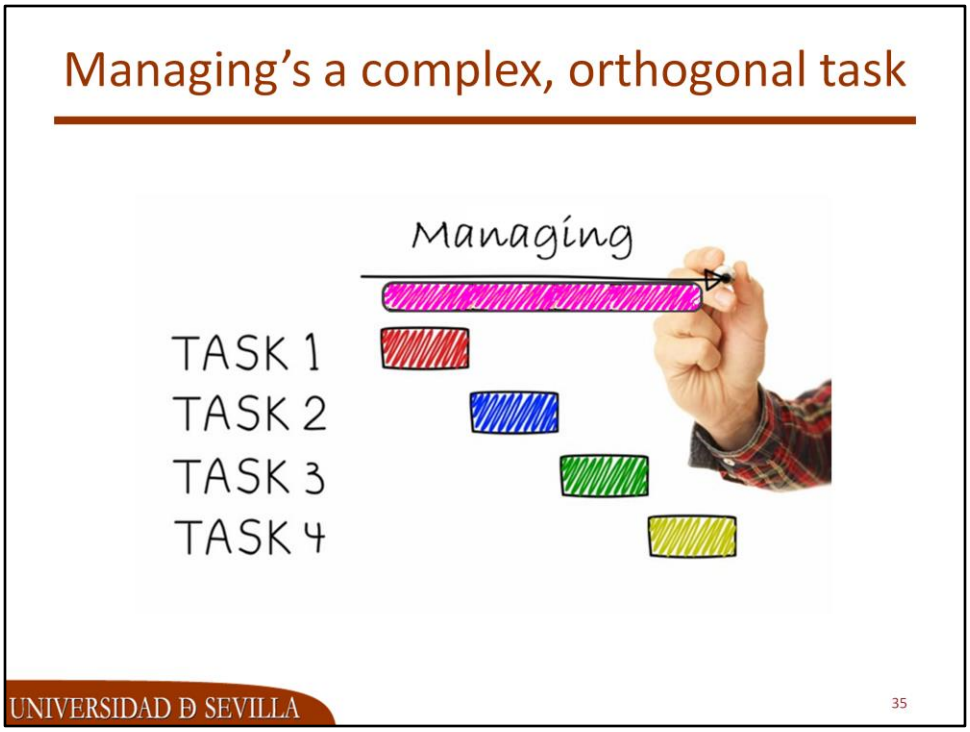
They only thing you must be aware of is that one task must amount to one worker. If a task actually involves more than a person, then split it into subtasks or activities so that every such subtask or activity is assigned to exactly one person. If a task is assigned to two people, then the chances that the task is not finished satisfactorily increase exponentially. You might think that co-ordination meetings are an exception because several people have to attend them. They are *not* such an exception: each person who attends the meeting must have a clear responsibility regarding the meeting and that responsibility must be scheduled as a task within the work plan; otherwise, no-one will know what he or she's expected to do in the meeting and it simply won't work!

## Every project needs a manager

UNIVERSIDAD Ð SEVILLA

34

Every project needs a manager. The manager is responsible for keeping the work under control, which basically amounts to organising the project into tasks, assigning them to workers, following them up, and taking actions to correct deviations. Please, do not undervalue your manager: he or she plays quite a very important role. Note, too, that managing requires a lot of negotiation; the manager is responsible for producing a prospective work plan, but he or she must discuss and polish it with his or her team.




Managing's a complex task. Unfortunately, our students tend to think that it's very easy. You'll soon realise that this is not true, that it's completely wrong. Managing's typically scheduled as an orthogonal task that runs in parallel with the other tasks. Please, allocate time and resources to manage your project and never underestimate the work of a good manager.

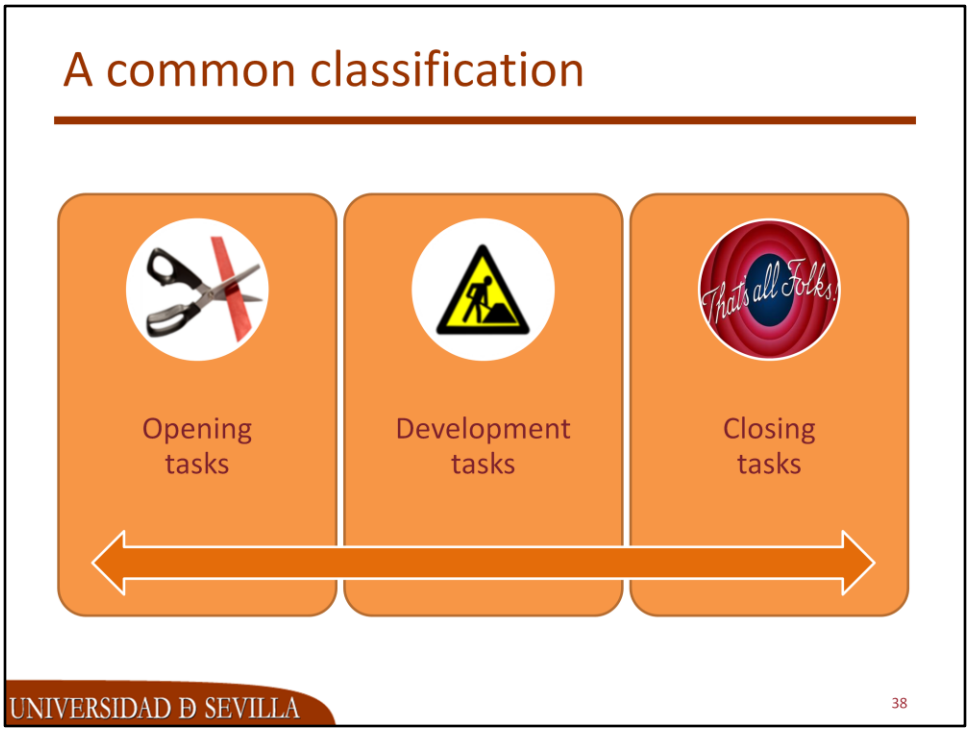
## Be flexible!

UNIVERSIDAD Ð SEVILLA

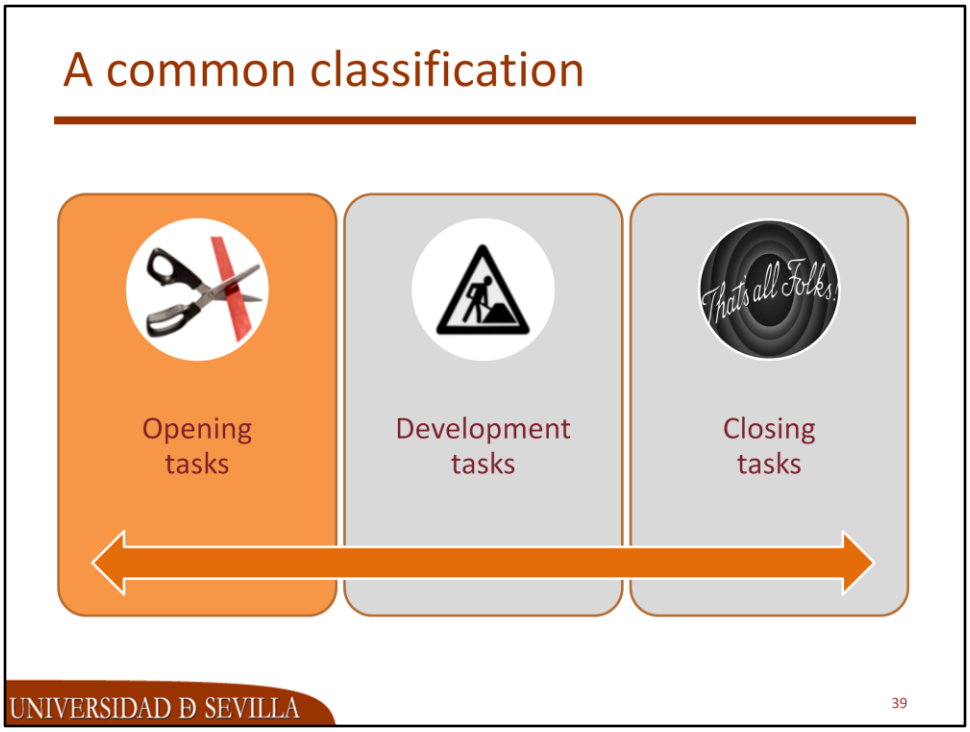
36

Be flexible. In the real world, plans have to change from time to time. What matters is that you meet the deadline with the resources that have been allocated to your project, but it's very common that tasks have to be re-scheduled and re-planned. Don't worry about that, it's a matter of practicing before you can produce accurate plans… And you'll have a lot more opportunities to practice in this subject.


It’s time to provide a clear guideline that you can follow to manage your projects.




The tasks are typically organised into three groups, namely: opening tasks, development tasks, and closing tasks.




Let's start with the opening tasks.

## Attend your lectures

UNIVERSIDAD Ð SEVILLA

40

Your first opening task consists in attending your lectures. They are paramount to understanding the theory that you need to start working on your projects. The lectures must be scheduled in your work plan because they are a part of the work that you have to carry out in order to develop your projects. In an actual job, you will also have occasional lectures because you'll need some training from time to time. You'll also have to attend occasional seminars to get acquainted with a new technology or a new method, and that's time that you spend at work, so it's time that you must allocate to a task in your work plan.

## Study your learning materials

UNIVERSIDAD Ð SEVILLA

41

After attending a lecture, you'll typically have to study the learning materials, and that consumes some time, too. So studying's another task that you must schedule in your work plan.

# Get familiar with your requirements

Document Modification History

| Version | Date | Author | Description |
|---|---|---|---|
| 1.0 | 05/16/2011 | Raggedy Andy | Initial Version |
| 1.1 | 05/18/2011 | Peter Rabbit | Added changes from stakeholders meeting |
| 1.2 | 05/23/2011 | Jane Smith | Added technical documentation details |

**Project Description**

The library will have a public-facing blog that will serve to communicate library news, events and resources, as well as providing the library's user community with the ability to comment on posts.

**Service Need**

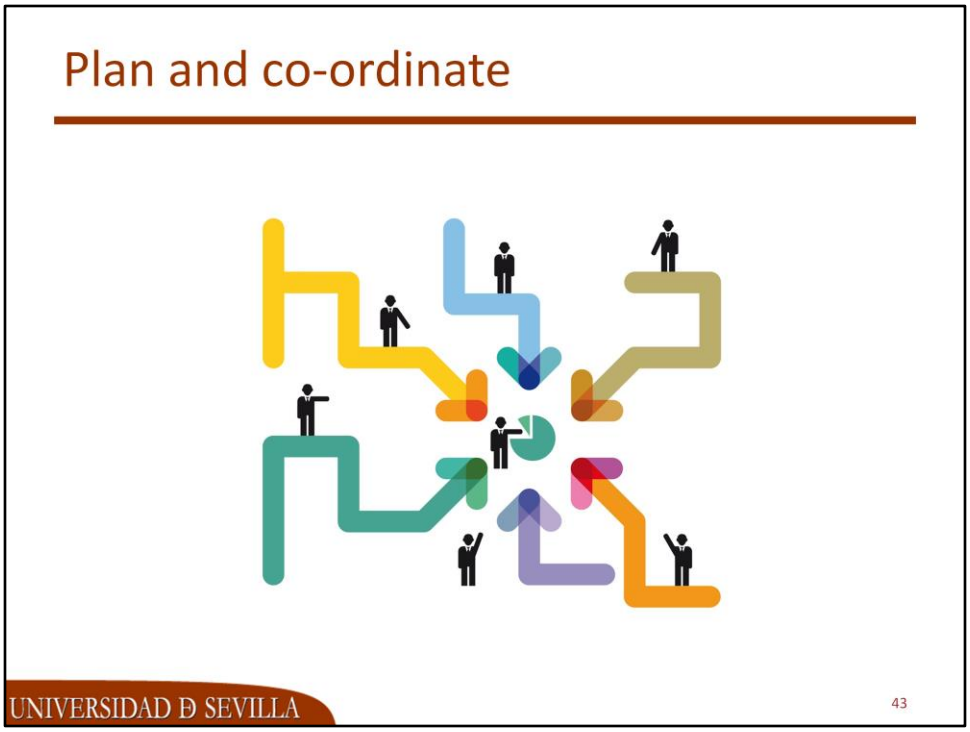
The University of Awesomeness Library has a need for communicating changes to policies, procedures, and resources to its users. Library blogs have been shown to be an effective method of communicating with library users. [1] We currently have a home-grown blog in place, but it lacks many of the functionalities we require.

**Project Purpose & Scope**

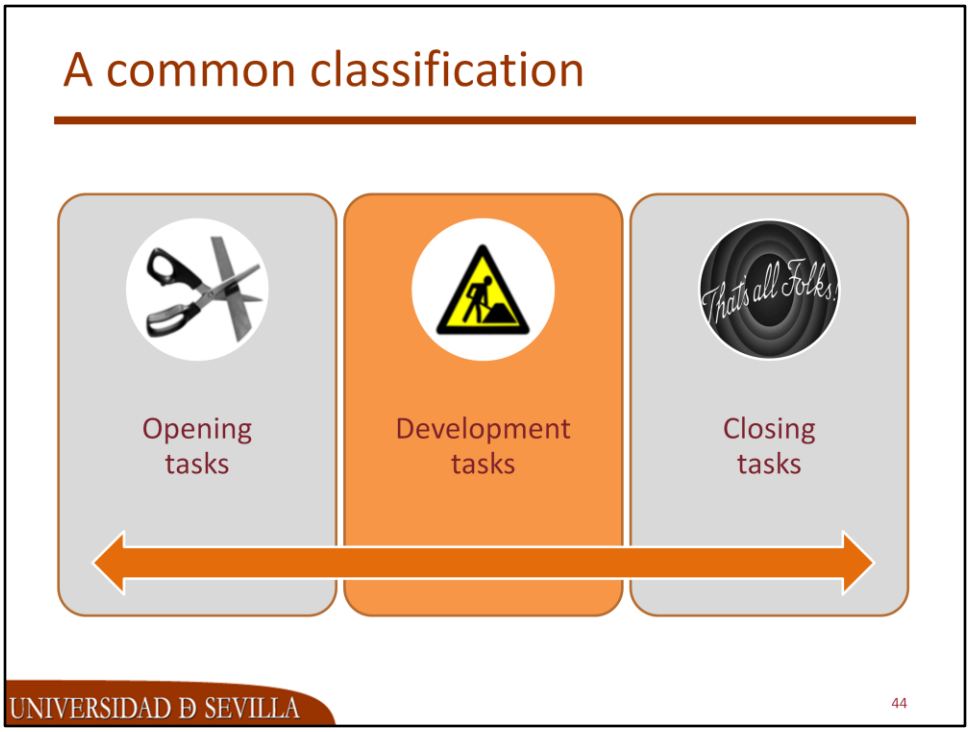
The purpose of this project is to provide library users with a forum for learning about and commenting



Getting familiar with your customer's requirements will also require some time: you need to go through the requirements elicitation document, you need to understand them, and very often you need to talk to your requirements elicitation engineer in order to shed some light on your doubts. So, please, don't forget to schedule a task to get familiar with your requirements.




And never forget that you work in a group, which means that you have to work according to a work plan and that you need to co-ordinate with other people. It's strongly recommended that you meet regularly in order to report on your progress, highlight your findings, ask for feedback, and align your tasks with your partners' tasks. Note that meetings don't have to be very long; typically, a 15-minute meeting every morning suffices to co-ordinate well.




Pretty simple, right? It's now time to delve into the development tasks.

## This is a good guideline (I)

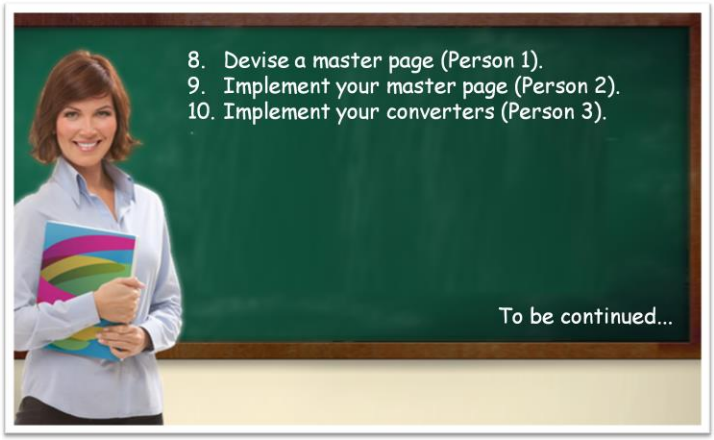



This slide shows a simple guideline that works pretty well in practice. It deals with a set of tasks whose goal is to set up a domain and a persistence model on top of which we'll implement the functional requirements. Please, read on:

- First, create a conceptual model from your requirements.
- Then, create a domain model from your conceptual model.
- After that, implement your domain model.
- When the domain model is ready, create a PopulateDatabase file that provides enough sample objects, i.e., make sure that some optional attributes are null, make sure that some collection attributes are empty collections, and the like. Summing up: make sure that the sample objects provide as much variability as possible.
- Then, implement your persistence model and polish it using the previous PopulateDatabase file.
- Now, check the models.
- And, finally, spread the models around so that everyone who is involved in the project can study them.

Note that we strongly recommend that the tasks be assigned as shown in this slide. This maximises the chances to find bugs, inconsistencies, errata, and other common problems.

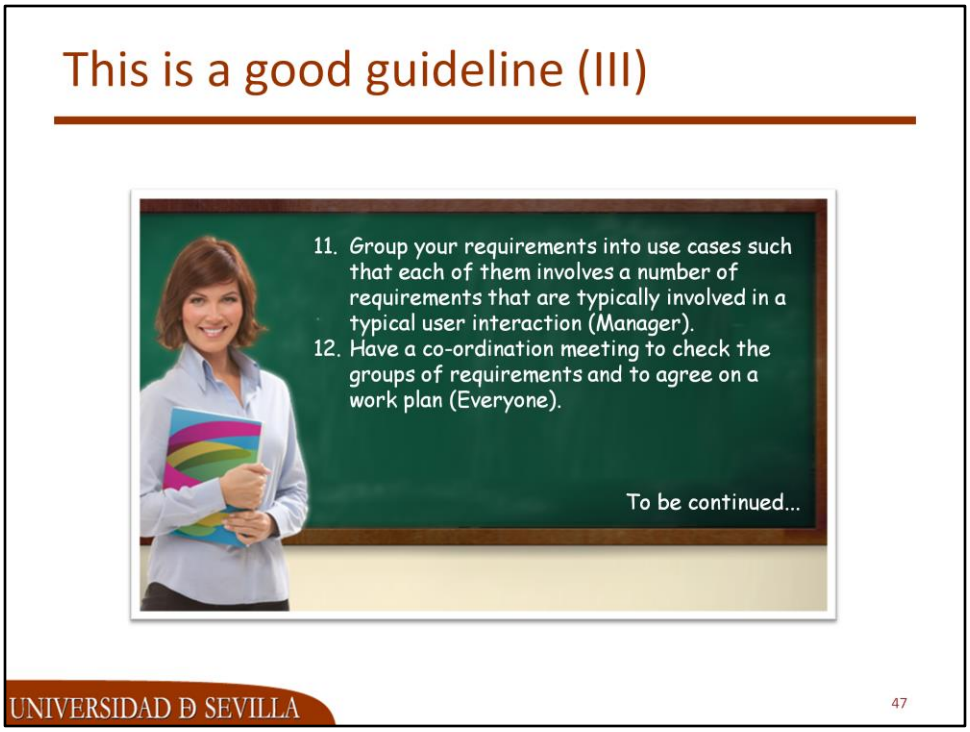
## This is a good guideline (II)






Our recommendation is that you should then schedule the following tasks;

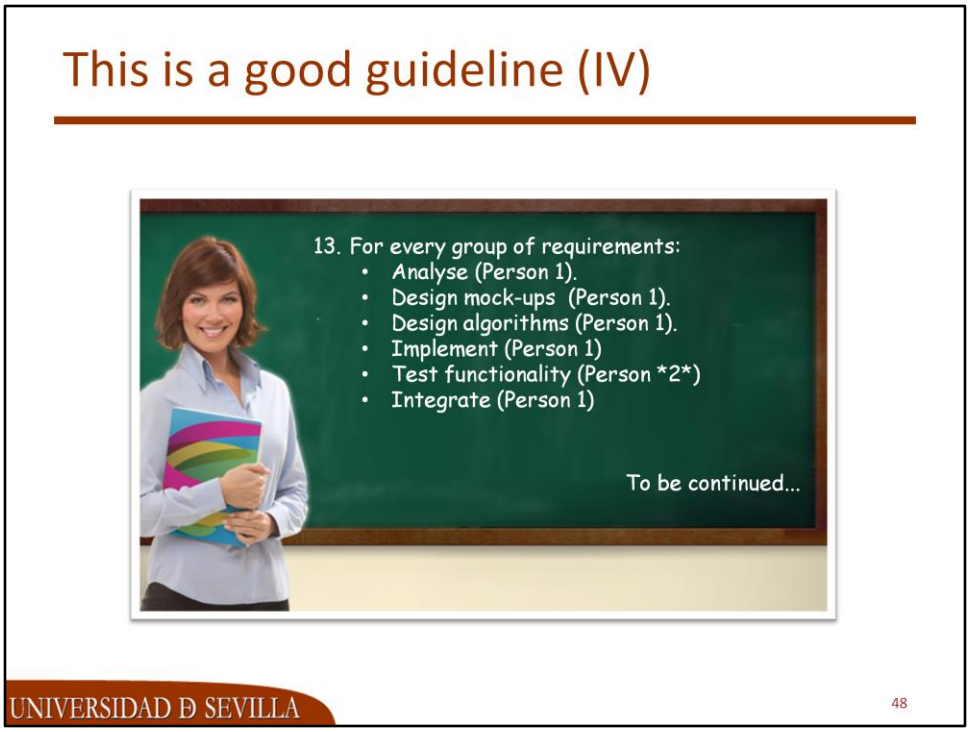
- Devise and implement a master page for your project. This will typically require you to collaborate with a web designer. You must make sure that the master page includes a menu that is complete enough and the web designer will focus on rendering that menu in as a friendly manner as possible.
- Implement your master page, which involves creating the corresponding views and I18n&l10n bundles.
- Implement your converters. You know that you need a string-to-entity and an entity-to-string converter per entity in your domain model plus a few datatype converters. So, the sooner you implement them, the better.




Now we have to work on the functional requirements. Our guideline's as follows:

- First, group your requirements into use cases such that each of them involves a number of requirements that are typically involved in a typical user interaction. Please, think of use cases as typical actions that a user performs. For instance, a customer keys in some key words in a search box, then presses the search button, and then browses the list of certifications that Acme Certification offers. This is a typical use case that involves a number of related functional requirements. Please, note that it's not easy at all to group functional requirements into use cases, so we don't expect you to do it perfectly... now. Perfection requires practice and this is the perfect subject to start practicing. Please, note, that we recommend that this task should be performed by a single person, the manager of the group.
- It is then time to have a co-ordination meeting in order to check that the groups of requirements make sense and to agree on a work plan. As a result, the groups might change; don't worry, that's pretty usual when the manager doesn't have enough experience, but you have to train a lot to gain that experience.

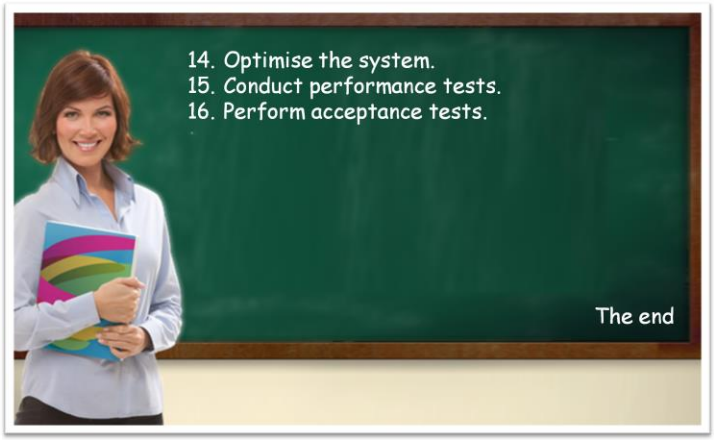
Note that it's very important that the groups into which you cluster the requirements must help the people who are assigned to implementing them work as uncoordinatedly as possible. We'll return too this issue later.




Now, for every group of requirements we recommend that you should plan on the following tasks:

- Analyse the group of requirements. You got familiar with them previously, but it's now time to revisit them and answer the last questions.
- Design mock-ups. You have to design mock-ups that provide good user interfaces from a functional point of view. Ask a web designer for help regarding how to make them friendly, easy to use, responsive, and so on.
- Design algorithms. Note that most requirements build on simple algorithms, but it's usually a good idea to sketch a pseudo code. Note, too, that most of the design effort is already done, since you have a reference architecture and a number of templates to implement listing and edition use cases.
- Implement repositories, views, services, and controllers for your requirements.
- Test your requirements from a functional point of view. But, please, bear in mind that this task must be assigned to a different person since this maximises the chances to find bugs that otherwise might go unnoticed. We'll work a lot more on functional testing in the forthcoming lessons.
- When the requirements are correctly implemented, integrate the corresponding artefacts into your master project.

## This is a good guideline (V)

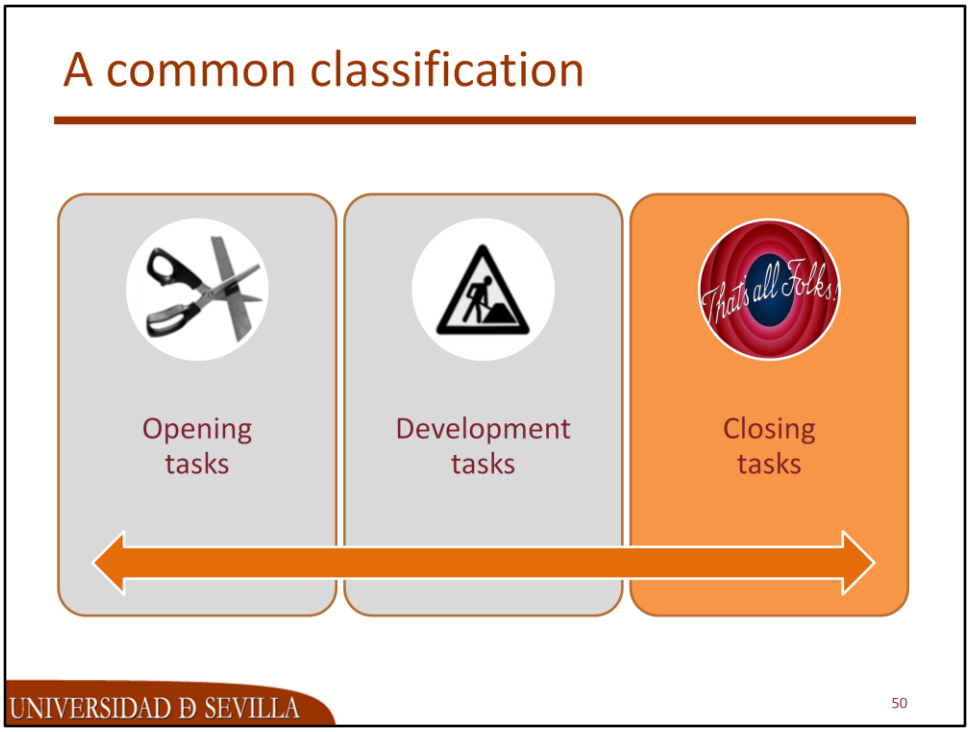



UNIVERSIDAD D SEVILLA

49

Finally, when everything's ready from a functional point of view, it's time to perform the following tasks:

- Optimise the system. Typically, this involves a review of the system in which the goal's to improve its performance. We'll learn a little more about this in the forthcoming theory session.
- Conduct performance tests. The goal is to find out the workload that our system can dispatch. We'll explore this topic in a forthcoming lesson.
- Perform acceptance tests. The goal is to show the system to our customer and get an approval from him or her. We'll also explore this topic in another lesson.




The guideline regarding the development tasks makes sense, doesn't it? It's now time to report on the closing tasks.

## A demo, please

UNIVERSIDAD Ð SEVILLA

51

We strongly suggest that you should plan on a demo task before concluding your project. A person of your team should present an interactive tour of your system. This very commonly helps detect a lot of minor problems that might otherwise go completely unnoticed.

## Package your deliverable


UNIVERSIDAD Ð SEVILLA

52

The next task consists in packaging your deliverable according to your customer's delivery instructions. Please, recall that there's a document entitled "On your deliverables" that provides detailed instructions regarding how to package your deliverable. Follow them very carefully, please.


Then upload your deliverable to the appropriate delivery system, which is the USE's e-learning platform in D&T.

## Download and verify it again


UNIVERSIDAD D SEVILLA

54

Next, someone else must download and verify it again. It's the most important task in your project. It absolutely makes sense that someone re-verifies your deliverable. Please, note that we emphasise that someone who has not been involved in packaging the deliverable must perform this task since this maximises the chances to find problems.


Finally, after the project's closed and delivered, it's strongly recommended that you should have a recap meeting, that is: a meeting in which you all meet, review your work, identify weak and strong points, make a decision regarding how to improve on (some of) your weak points, and agree on your strong points. The recap meeting will surely help you improve in your next project.

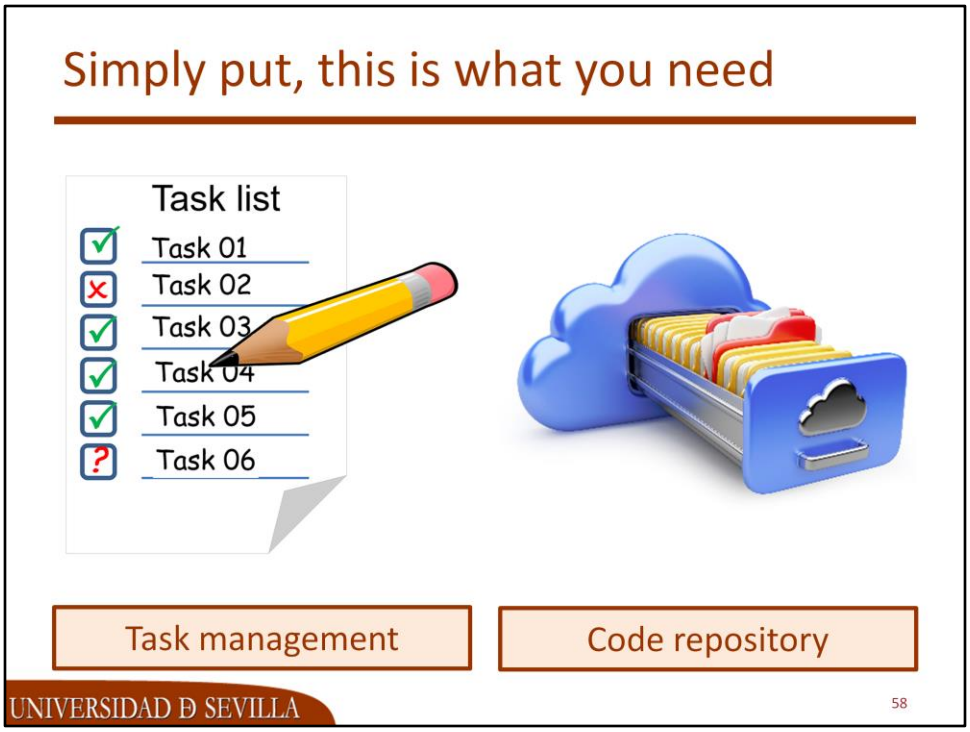
## And have some fun together!

UNIVERSIDAD Ð SEVILLA

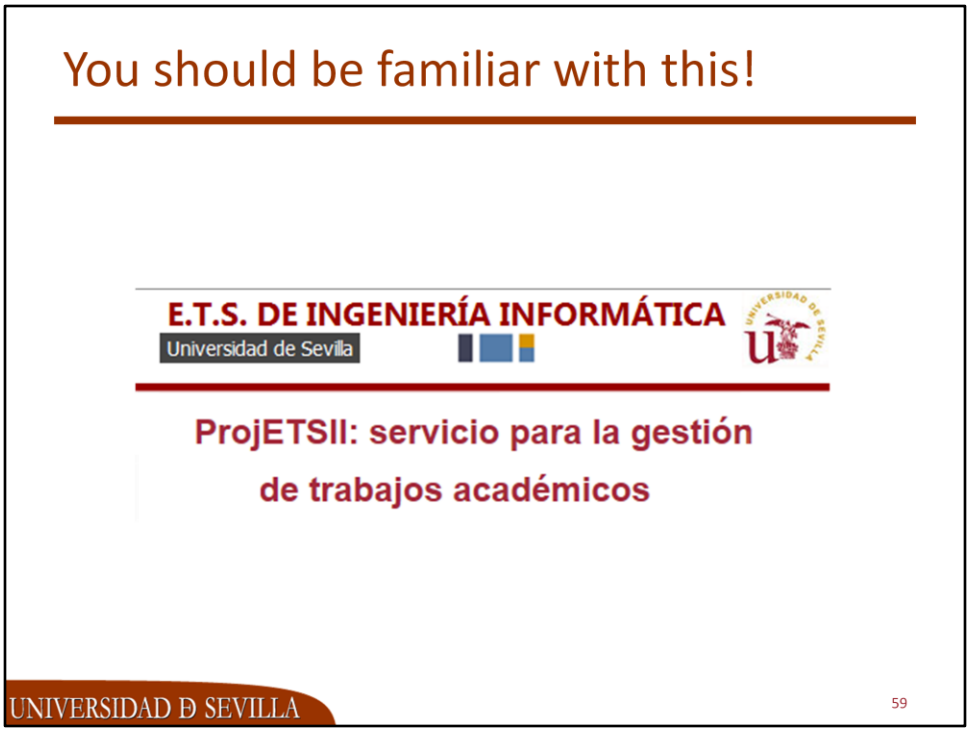
56

And last, but not least: seize the opportunity to have some fun together.

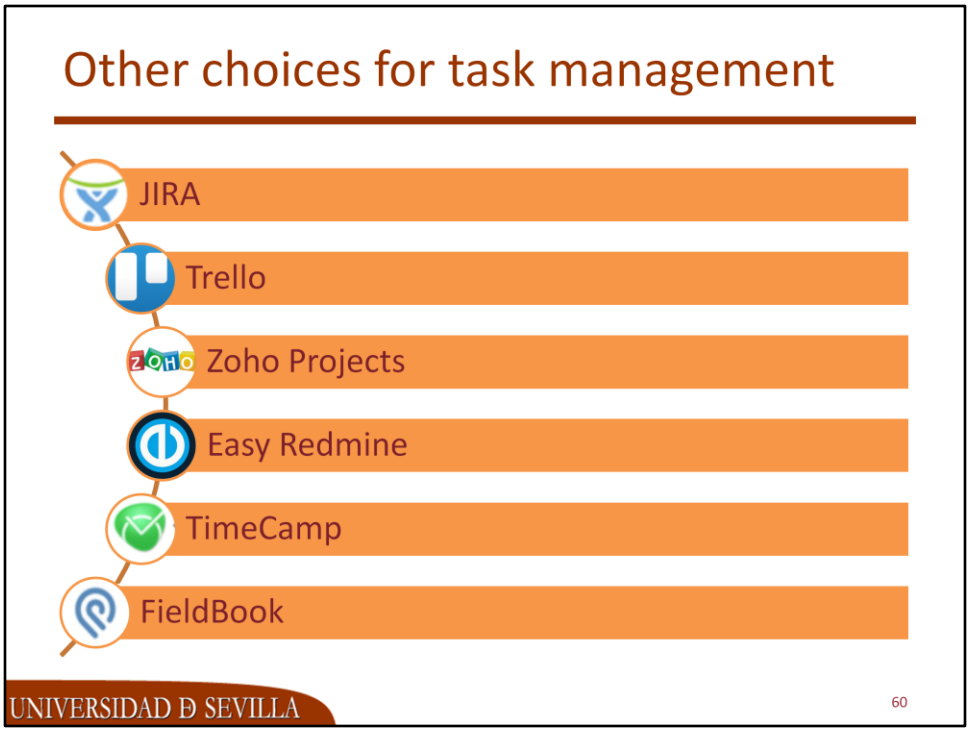

It’s now about time to report on the tools that you need to manage your projects.



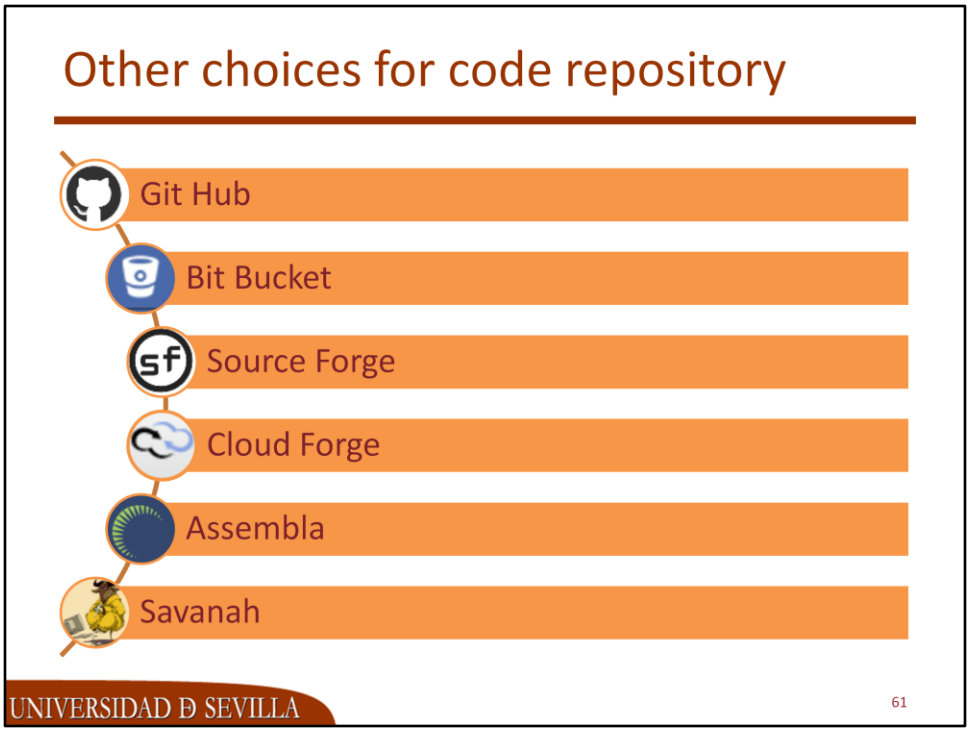

Basically, you need a task manager and a code repository. If you wish, you might also try a continuous integration system; but we can't provide you with such a server, sorry.



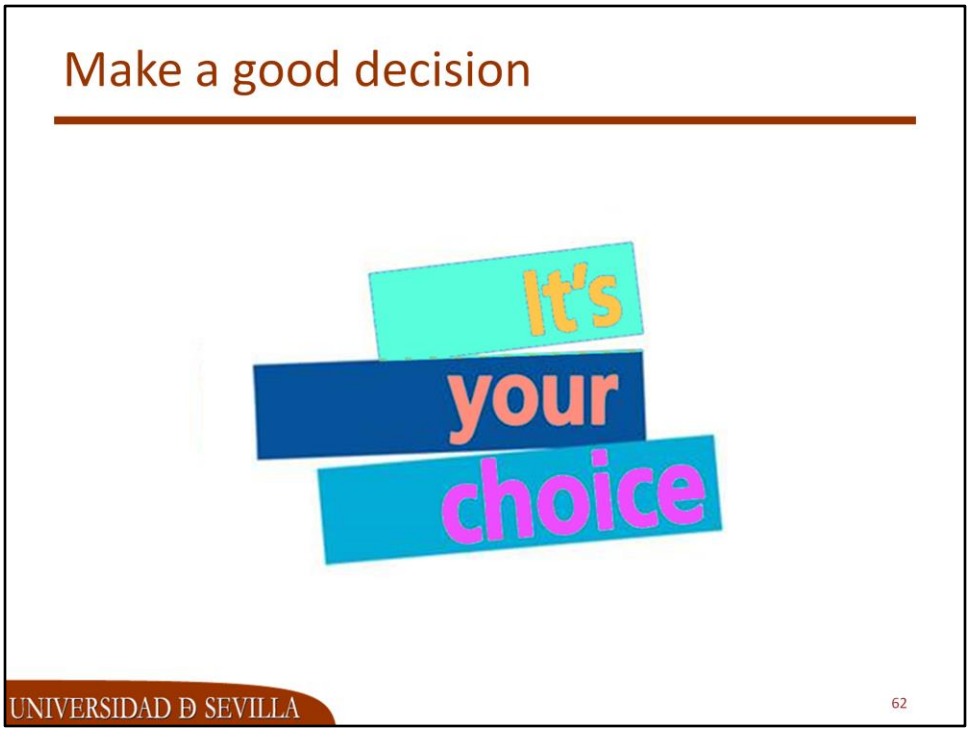

We strongly recommend that you should use ProjETSII, but you might give a try to some other tools during the Spring semester. ProjETSII's been great to get accustomed with a project management tool, but it's time to explore the many alternatives that are available out there.



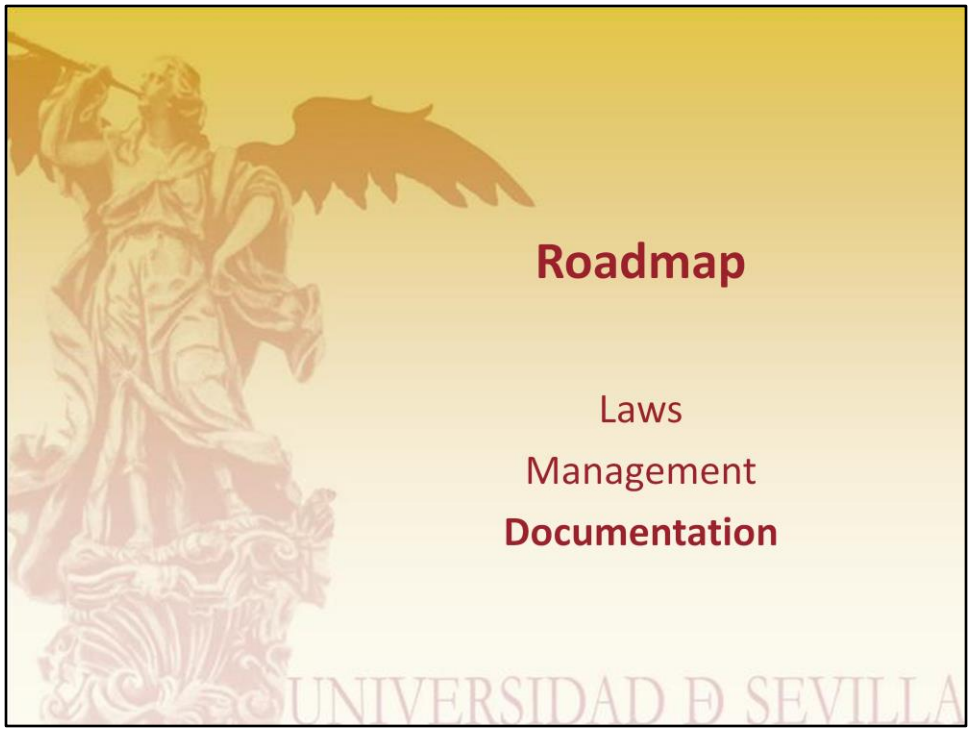

Here you have a listing with some popular task management tools. Please, browse https://en.wikipedia.org/wiki/Comparison_of_project_management_software for a complete list. We have sorted them according to our real-world experience: the first one and the second one are the most common in our local context, but the others are also popular enough to deserve a try.




And here you have a few suggestions regarding the code repository. The complete list is available at https://en.wikipedia.org/wiki/Comparison_of_source_code_hosting_facilities. We have also sorted them according to our real-world experience: the first one's the most common in our local context and the others are also used in many companies.

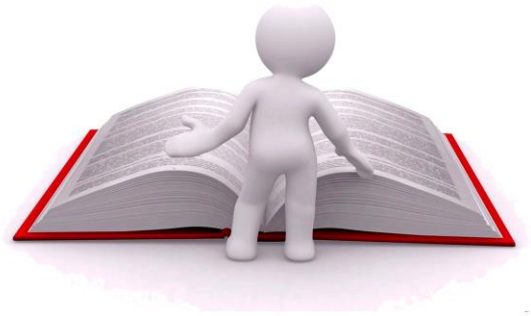



Please, select the tools that you prefer and learn... or keep using ProjETSII. Might it be a good idea to explore them all? Definitely. There are several projects ahead before we conclude this semester and you might well seize the opportunity to get familiar with the previous tools. The decision is up to you!




Managing the projects that we're going to develop during the spring semester shouldn't be very difficult. Let's now report on a project's documentation.

## What's documentation?



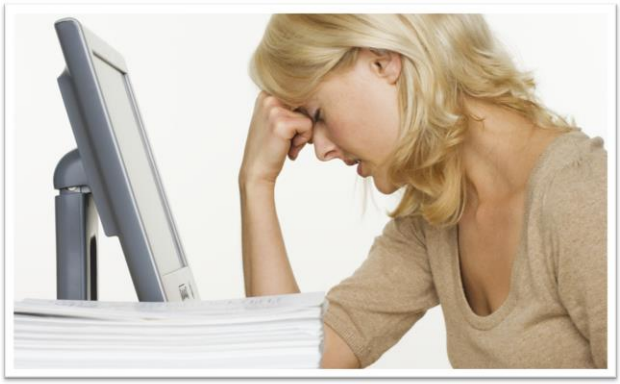
It's a collection of artefacts that help other people understand your project

UNIVERSIDAD D SEVILLA

64

The documentation of a project consists in a number of artefacts that help other people understand the internals of your project.

## Why is it important?



Because a bad documentation amounts to trouble during maintenance

UNIVERSIDAD Ð SEVILLA

65

The documentation's very important in every project because if you don't have a good documentation, then you're very likely to get in trouble during maintenance.

## It’s not about tons of paper!

UNIVERSIDAD Ð SEVILLA

66

But, please, note that producing a good documentation doesn’t amount to producing tons of paper. We’ll make our best effort to be agile regarding documentation. Keep reading.


Next, we're going to talk about documenting planning, development, and deployment.


Let’s start with documenting planning.

# Typical documentation

Activity reports

Calendars

Gantt diagrams

Project costs

UNIVERSIDAD D SEVILLA

69

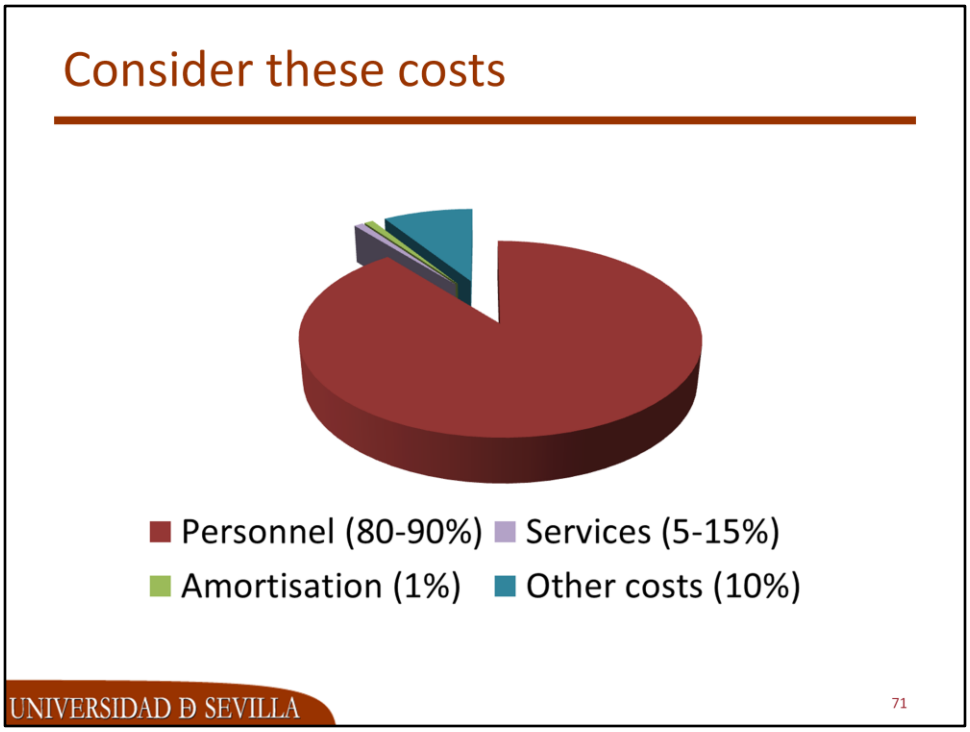
The typical documentation regarding planning consists of activity reports, calendars, Gantt diagrams, and project costs. Your customers will be mostly interested in the costs.

## How am I suppose to manage costs?


UNIVERSIDAD Ð SEVILLA

70

You can easily have access to activity reports, calendars, or Gantt diagrams with project management tools. But what about project costs? We're pretty sure that this is the first subject in which you're required to manage a budget, so you need some tips before going ahead.




A typical project involves many costs, but we recommend that you should focus on the following ones: personnel, services, amortisation, and other costs.

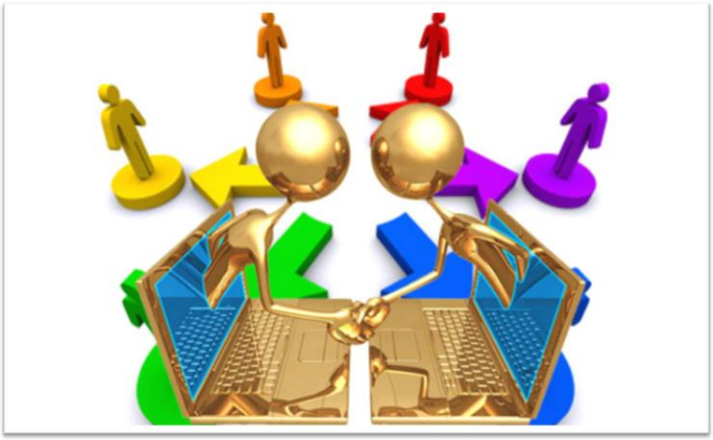
## Personnel costs (80-90%)


UNIVERSIDAD Ð SEVILLA

72

Personnel costs are the most important in a software project. Typically, they amount to 80-90% of the budget, if not more. Nowadays, the cost of a junior software engineer's roughly 12-14€, which after taxes and other stuff means that he or she'll get 4-5€ in his or her pocket. That's the average income of an average developer who didn't pass D&T or got a C or a B in D&T. Fortunately, those who prove to be above the average and earn an A or an A+ can easily earn a little more money per hour.

## Services (5-15%)



UNIVERSIDAD Ð SEVILLA

73

Services include everything you need from another company. Typical services include training (to get familiar with a new technology or a new method), consulting (to learn how to solve a complex problem), renting (computers and software licenses), or subcontracting (when a part of the project's subcontracted to another company). Typical service costs range in interval 5-15%.

## Amortisation (1%)


UNIVERSIDAD Ð SEVILLA

74

Amortisation is quite a simple concept: you need a computer and some software licences to develop your projects. A typical company does not own the computers or the software licenses with which they work; they typically rent them, which means that there's no amortisation involved, but a service cost. But it is very unlikely that you (our students) are renting your computers or your software licenses; very likely, you bought them some time ago and you're now amortising them, that is, you use a bit of your computer and your software licenses in every subject. As an example, assume that you've bought a computer or a software license and you have to amortise it in *N* years (*N* is typically three or four years), which amounts to roughly N * 1980 work hours; if a project requires roughly *W* work hours (for instance, *W* = 30 hours in this lesson), then the amortisation of your computer and your software is roughly (100 * *W* * *C* ) / (N * 1980), where *C* denotes the cost of your computer or your software licence. Pretty simple, isn't it? Make your browser for http://tinyurl.com/Amortizacion to find the latest information about amortisation in Spain.

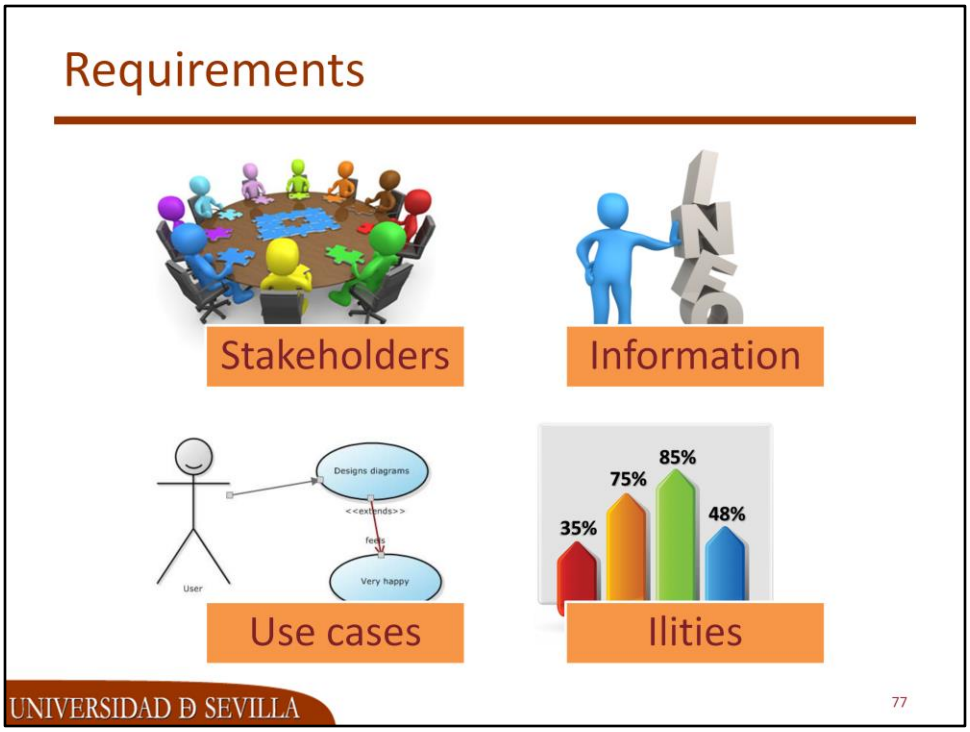
## Other costs (10%)

UNIVERSIDAD Ð SEVILLA

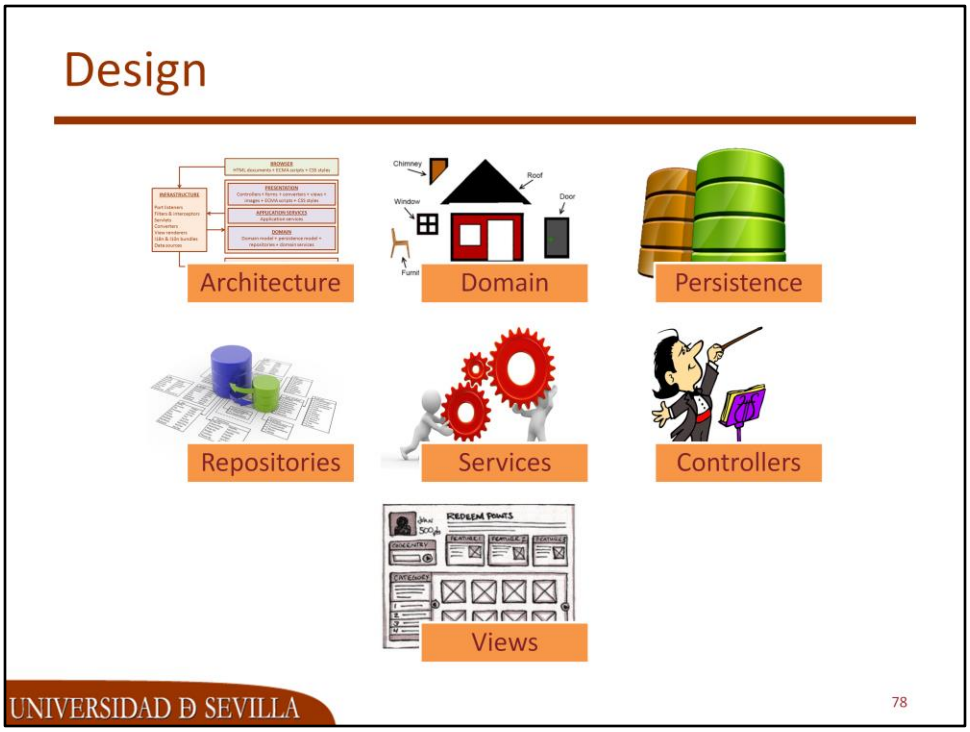
75

Finally, there are other costs that are typically calculated as 10% the remaining cost. These costs account for deviations to the original planning; for instance, a task that requires more hours than were initially planned or a service that must be contracted but wasn't initially planned.


Let's now analyse the development documentation.

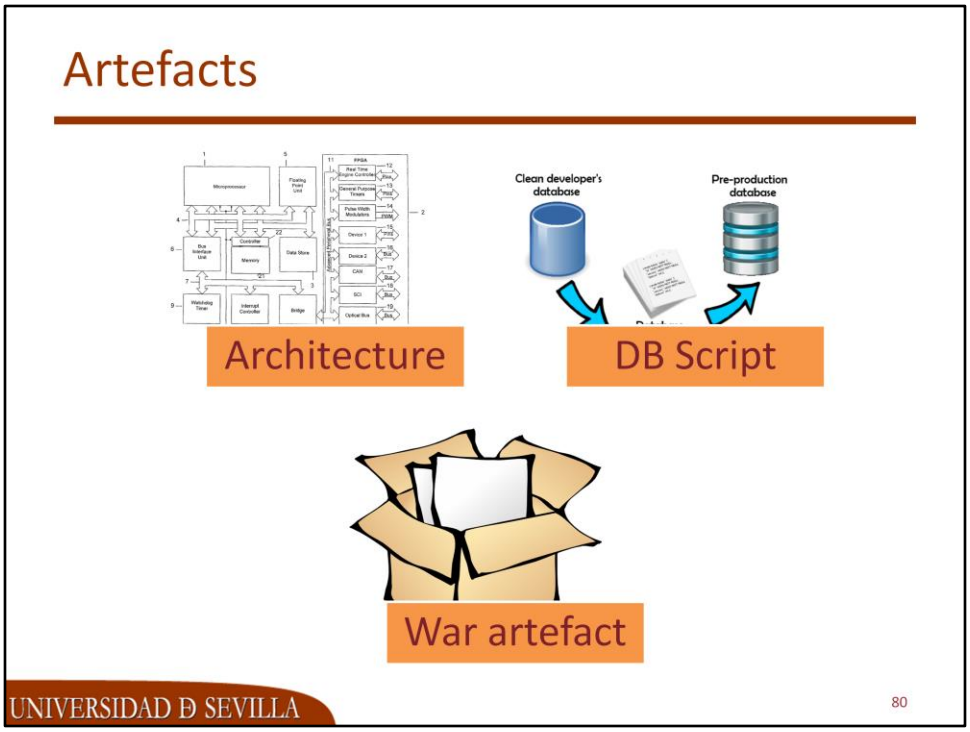



You basically have to document your requirements, which amounts to documenting the stakeholders, the information requirements, the use cases (including functional requirements and mock-ups), and your non-functional requirements (ilities). Note that this information's usually provided by a requirements engineer, but it's very likely that it evolves along the course of your project. Please, note that the lectures provide the majority of these items in their project statements. Please, don't forget this: be agile! Don't re-phrase the documentation that we provide to you.

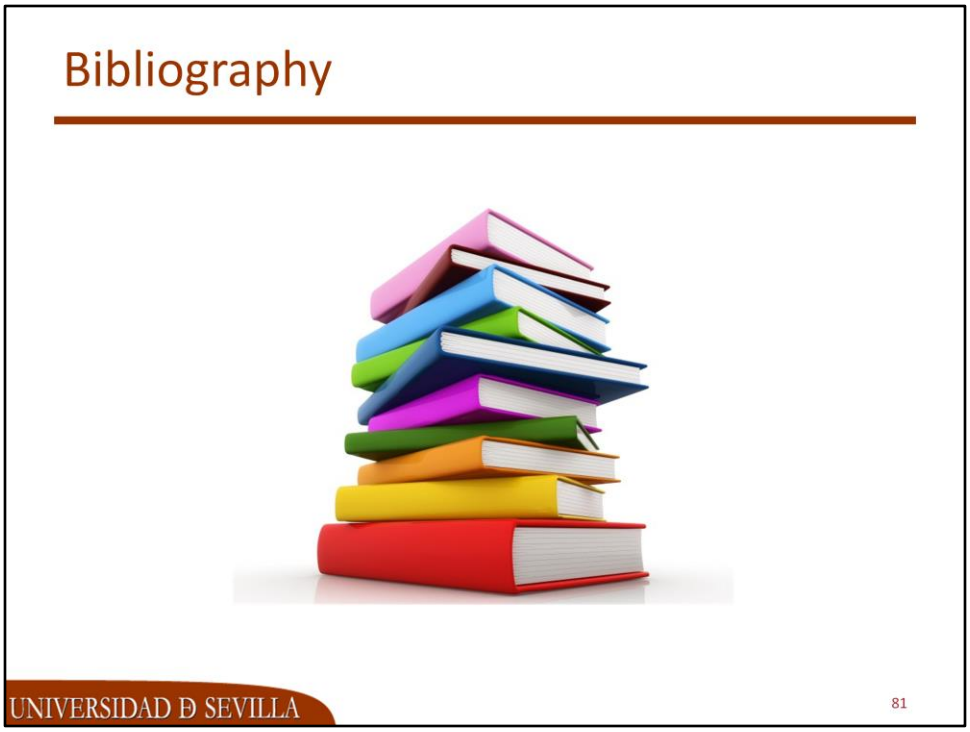



Regarding the design, you have to document the architecture, your domain model, your persistence model, your repositories, services, controllers, and views. That's all. Please, note that you don't have to document your architecture, since the lecturers have explained it in the previous lectures. Neither have you to produce tons of paper to document your domain classes or your persistence model: a UML domain model's enough. Regarding your repositories, it's strongly recommended that non-trivial queries have a comment to explain them. There's little to document regarding services or controllers; document them using appropriate comments if they deviate from the patterns that we've taught. Regarding the views, it's strongly recommended that you should document your model variables. Please, don't forget this: be agile!


The development documentation's simple, isn't it. So is the deployment documentation.




You basically need to document the architecture of your deployment, the script to be used to create the database in the pre- or the production configuration, and the war artefact that you've created. Little else is necessary to document your deployments. In the kind of projects that we're developing, the architecture is very simple, and it was explained in a previous lecture. You needn't provide any details about it.




It's not easy to recommend good bibliography for this lecture. We think that you should consider these notes as your primary source of information.

There are some textbooks that provide a good introduction to laws, but we honestly think that they fall outside the scope of this subject. The accompanying materials include a copy of the laws that we have explored in this lesson. This should be enough for you.

We suggest that you might take a look at the following book on project management:

Fundamentals of project management, 5th edition
Joseph Heagney
Amacom Publisher, 2016
ISBN: 0814437362

Regarding documentation, we recommend taking a look at the 32nd chapter of the following book:

Code Complete, 2nd edition
Steve McConnell
Microsoft Press, 2004
ISBN: 0735619670

## Time for questions, please




Time for questions, please. Note that we won't update the slides with the questions that are posed in the lectures. Please, attend them and take notes.


Thanks for attending this lecture!  See you next day!