Openbravo ERP life cycle management

Juan Pablo Aroztegi

Release Management Team
Openbravo

March 22, 2010

Outline

1. Overview: development, testing, production
   - Problem
   - Definition
   - Goals
2. Effective life cycle management
   - Development
   - Testing
   - Production
   - Tools
3. Summary
4. Q&A

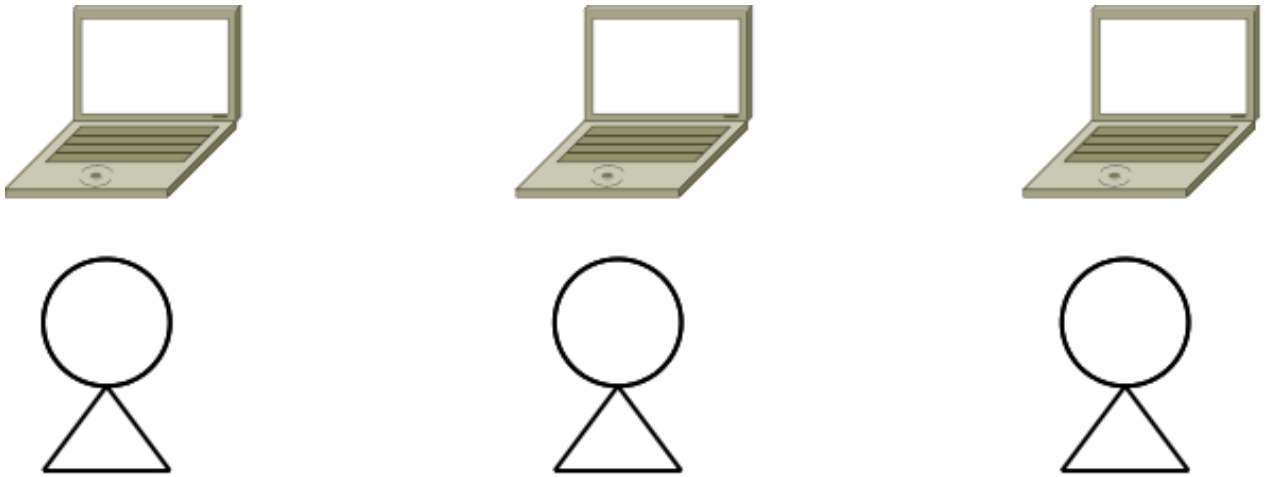
Outline

1. Overview: development, testing, production
   - Problem
   - Definition
   - Goals
2. Effective life cycle management
   - Development
   - Testing
   - Production
   - Tools
3. Summary
4. Q&A

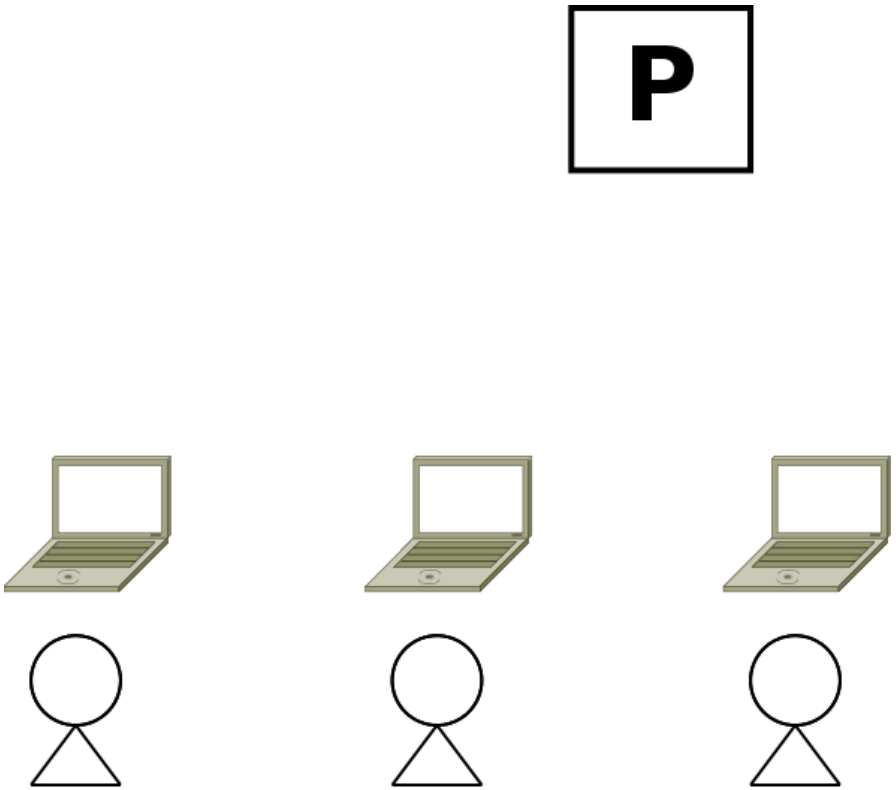
Problem

- 1 project.
- 3 developers.
- 1 production system.

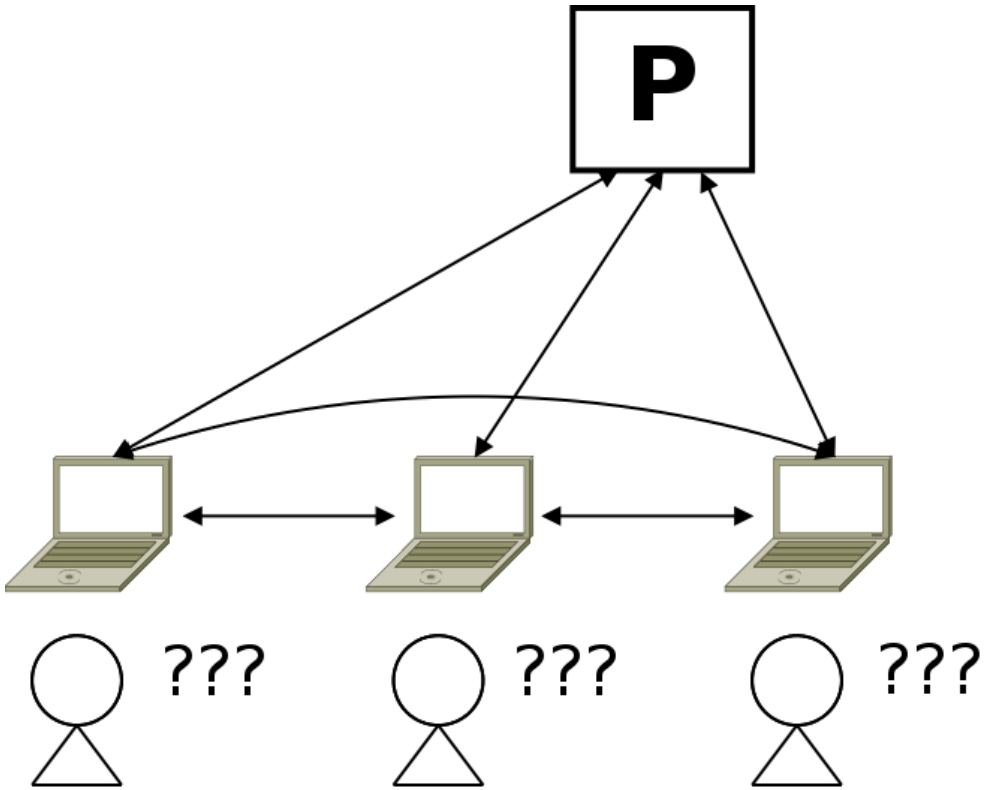
What is this all about?



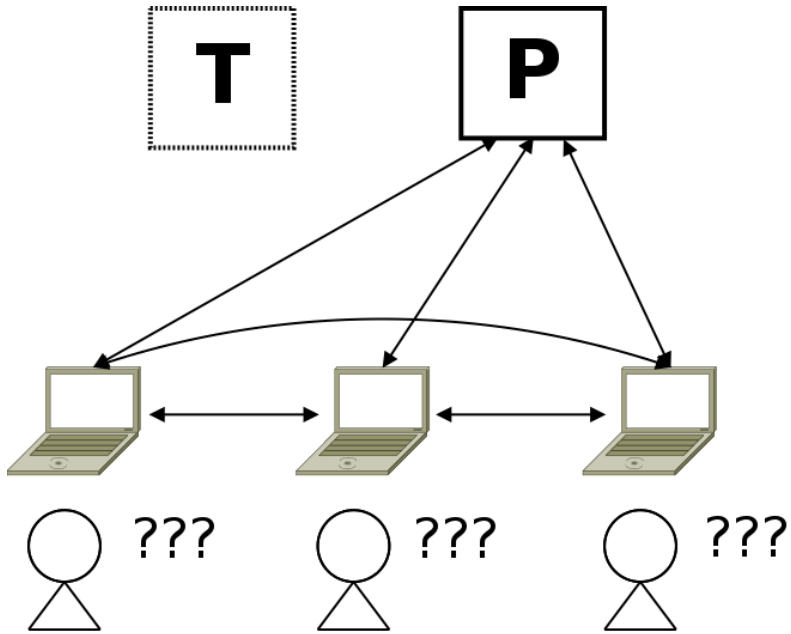
What is this all about?



What is this all about?



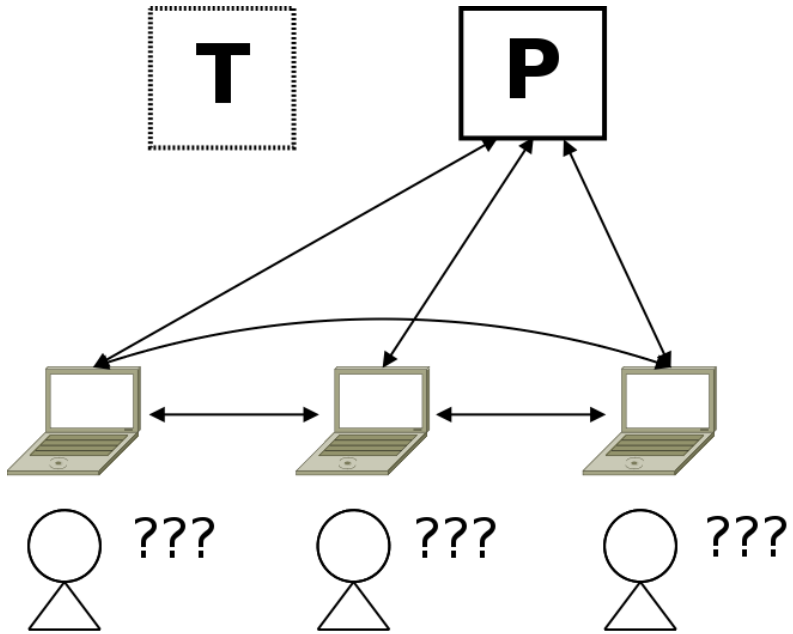
What is this all about?



Life cycle management

How do I glue these components effectively?

What is this all about?



Life cycle management

How do I glue these components effectively?

Goals

- **Secure production**: test anything before going into production.
- **Customize** the ERP easily.
- **Control and share** changes among developers.
- **Demos**: show latest features to the customer **before** going to production.

Goals

- **Secure production**: test anything before going into production.
- **Customize** the ERP easily.
- **Control and share** changes among developers.
- **Demos**: show latest features to the customer **before** going to production.

Goals

- **Secure production**: test anything before going into production.
- **Customize** the ERP easily.
- **Control and share** changes among developers.
- **Demos**: show latest features to the customer **before** going to production.

Goals

- **Secure production**: test anything before going into production.
- **Customize** the ERP easily.
- **Control and share** changes among developers.
- **Demos**: show latest features to the customer **before** going to production.

Goals

- **Secure production**: test anything before going into production.
- **Customize** the ERP easily.
- **Control and share** changes among developers.
- **Demos**: show latest features to the customer **before** going to production.

Outline

1. Overview: development, testing, production
   - Problem
   - Definition
   - Goals
2. Effective life cycle management
   - Development
   - Testing
   - Production
   - Tools
3. Summary
4. Q&A

Customize the ERP: how?

What's development?

Any code change you do, at any point in time.

Options:

1. I'll change Core directly . . . DON'T!
2. I'll use modules.

Customize the ERP: how?

What's development?

Any code change you do, at any point in time.

Options:

1. I'll change Core directly . . . DON'T!
2. I'll use modules.

Customize the ERP: how?

What's development?

Any code change you do, at any point in time.

Options:

1. I'll change Core directly . . . **DON'T!**
2. I'll use modules.

Customize the ERP: SCM

Why does a SCM matter?

- **Control** my own changes: how did I solve that issue?
- **Share** changes easily. Comfortable distribution.
- **Concurrent** development.
- **Revert** and tweak changes.

Useful even if you work alone!

Customize the ERP: SCM

Why does a SCM matter?

- **Control** my own changes: how did I solve that issue?
- **Share** changes easily. Comfortable distribution.
- **Concurrent** development.
- **Revert** and tweak changes.

Useful even if you work alone!

Customize the ERP: SCM

1 repository pre module **vs** 1 repository for all the modules

- Mercurial.
- Git.
- Bazaar.
- Subversion.

Customize the ERP: SCM

1 repository pre module **vs** 1 repository for all the modules

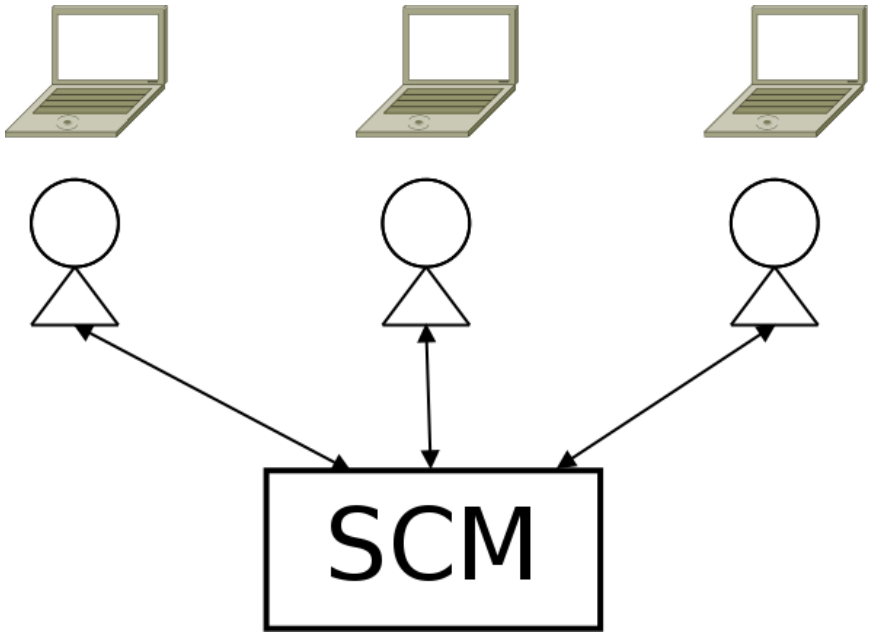
- Mercurial.
- Git.
- Bazaar.
- Subversion.

Customize the ERP: SCM

1 repository pre module **vs** 1 repository for all the modules

- Mercurial.
- Git.
- Bazaar.
- Subversion.

Customize the ERP: SCM



Customize the ERP: issue tracker

Why does an issue tracker matter?

- Traceability of bugs and their resolutions.
- Increase visibility of the development process.
- And why not, involve your customer.

Useful even for a single module!

Customize the ERP: issue tracker

Why does an issue tracker matter?

- Traceability of bugs and their resolutions.
- Increase visibility of the development process.
- And why not, involve your customer.

Useful even for a single module!

Customize the ERP: issue tracker + SCM

A SCM and an issue tracker; I don't have time to set up that

Use a Forge! Even for private projects.

Plenty of options:

- Openbravo Forge.
- Bitbucket.
- Github.
- Launchpad.

Customize the ERP: issue tracker + SCM

A SCM and an issue tracker; I don't have time to set up that

Use a Forge! Even for private projects.

Plenty of options:

- Openbravo Forge.
- Bitbucket.
- Github.
- Launchpad.

Customize the ERP: issue tracker + SCM

A SCM and an issue tracker; I don't have time to set up that

Use a Forge! Even for private projects.

Plenty of options:

- Openbravo Forge.
- Bitbucket.
- Github.
- Launchpad.

Customize the ERP: issue tracker + SCM

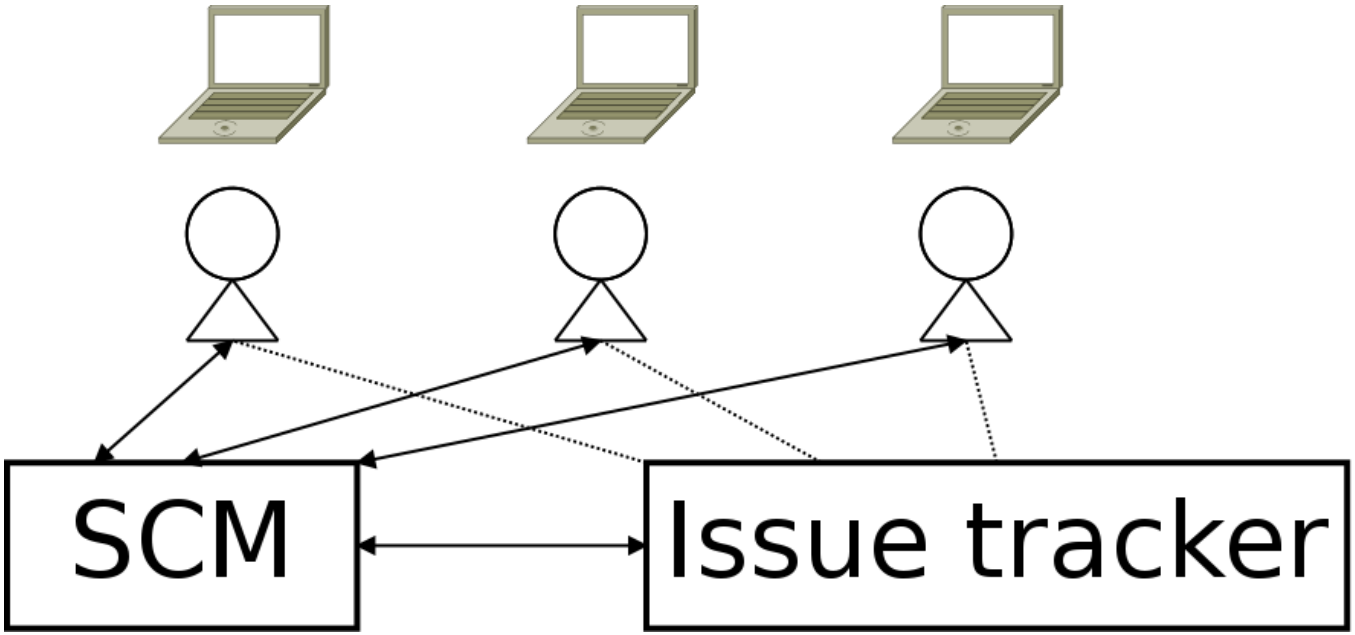
A SCM and an issue tracker; I don't have time to set up that

Use a Forge! Even for private projects.

Plenty of options:

- Openbravo Forge.
- Bitbucket.
- Github.
- Launchpad.

Customize the ERP: issue tracker



Testing: why it matters

- Test your developers in a real environment.
- Let the customer test it.
- Test your upgrades: modules, Core.

Production and Testing: 1 machine **vs** 2 machines

Testing must be a clone of production. Specially in software.
Recommendations:

- 2 x Cloud Appliance
- 2 x Ubuntu

Testing: why it matters

- Test your developers in a real environment.
- Let the customer test it.
- Test your upgrades: modules, Core.

Production and Testing: 1 machine vs 2 machines

Testing must be a clone of production. Specially in software.
Recommendations:

- 2 x Cloud Appliance
- 2 x Ubuntu

Testing: why it matters

- Test your developers in a real environment.
- Let the customer test it.
- Test your upgrades: modules, Core.

Production and Testing: 1 machine vs 2 machines

Testing must be a clone of production. Specially in software.
Recommendations:

- 2 x Cloud Appliance
- 2 x Ubuntu

Testing: why it matters

- Test your developers in a real environment.
- Let the customer test it.
- Test your upgrades: modules, Core.

Production and Testing: 1 machine vs 2 machines

Testing must be a clone of production. Specially in software.
Recommendations:

- 2 x Cloud Appliance
- 2 x Ubuntu

Development to testing

Developers send customizations to Testing through OBX files

1. Replicate production to testing. How?
2. Apply your OBX files, upgrade Core, etc. using the Module Management Console.
3. Check that everything is correct. Let the customer test it.

Development to testing

Developers send customizations to Testing through OBX files

1. Replicate production to testing. How?
2. Apply your OBX files, upgrade Core, etc. using the Module Management Console.
3. Check that everything is correct. Let the customer test it.

Development to testing

Developers send customizations to Testing through OBX files

1. Replicate production to testing. How?
2. Apply your OBX files, upgrade Core, etc. using the Module Management Console.
3. Check that everything is correct. Let the customer test it.

Development to testing

Developers send customizations to Testing through OBX files

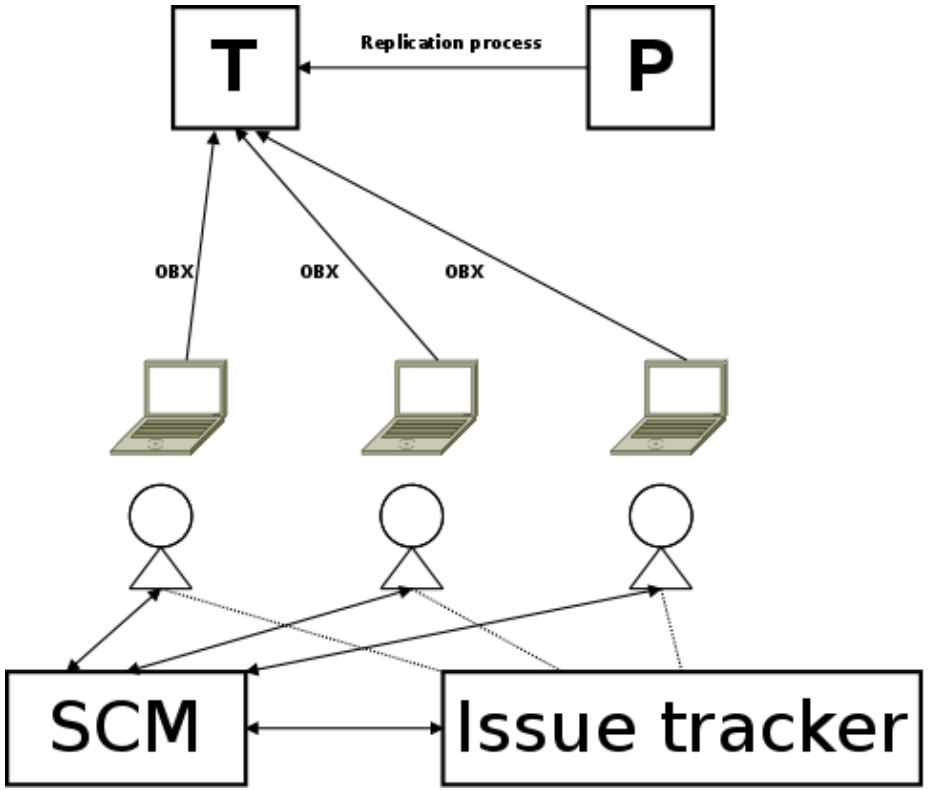
1. Replicate production to testing. How?
2. Apply your OBX files, upgrade Core, etc. using the Module Management Console.
3. Check that everything is correct. Let the customer test it.

Development to testing

Developers send customizations to Testing through OBX files

1. Replicate production to testing. How?
2. Apply your OBX files, upgrade Core, etc. using the Module Management Console.
3. Check that everything is correct. Let the customer test it.

Development to testing



Applying changes

Possible kind of changes:

1. Core update.
2. External module updates.
3. Self developed module updates.

Howto:

1. Do it in testing first.
   - Replicate production to testing.
   - ...
2. Rerun those steps in the production system.

Applying changes

Possible kind of changes:

1. Core update.
2. External module updates.
3. Self developed module updates.

Howto:

1. Do it in testing first.
   - Replicate production to testing.
   - ...
2. Rerun those steps in the production system.

Applying changes

Possible kind of changes:

1. Core update.
2. External module updates.
3. Self developed module updates.

Howto:

1. Do it in testing first.
   - Replicate production to testing.
   - ...
2. Rerun those steps in the production system.

Replicating production to testing

- Manual, repetitive, tedious.
- We've developed a command line tool that does exactly this in an easy manner.

Demo!

Replicating production to testing

- Manual, repetitive, tedious.
- We've developed a command line tool that does exactly this in an easy manner.

Demo!

Outline

1. Overview: development, testing, production
   - Problem
   - Definition
   - Goals
2. Effective life cycle management
   - Development
   - Testing
   - Production
   - Tools
3. Summary
4. Q&A

Summary

- Development collaboration: modules + SCM + issue tracker
- Testing: use OBX files and the MMC. Replicate *Production* to *Testing* first.
- Production: do it in testing first. Repeat the steps in production later.

Outline

1. Overview: development, testing, production
   - Problem
   - Definition
   - Goals
2. Effective life cycle management
   - Development
   - Testing
   - Production
   - Tools
3. Summary
4. Q&A

Thank you.

Questions?

juanpablo.aroztegi@openbravo.com
http://jpabloae.wordpress.com
IRC:iarwain(Freenode)