Forms to APEX Project Lifecycle Management

Dmitry Mezhuev, Intrum

intrum

Intro….

- Dmitry Mezhuev
- 20 years of Oracle experience and 15 years with Forms
- Now working as IT architect in Intrum

Intrum – (former Intrum Justitia) credit management service company

intrum

Agenda

- Forms to APEX project intro
- Architecture and Development environment & process
- APEX
- Q&A

intrum

Project

intrum

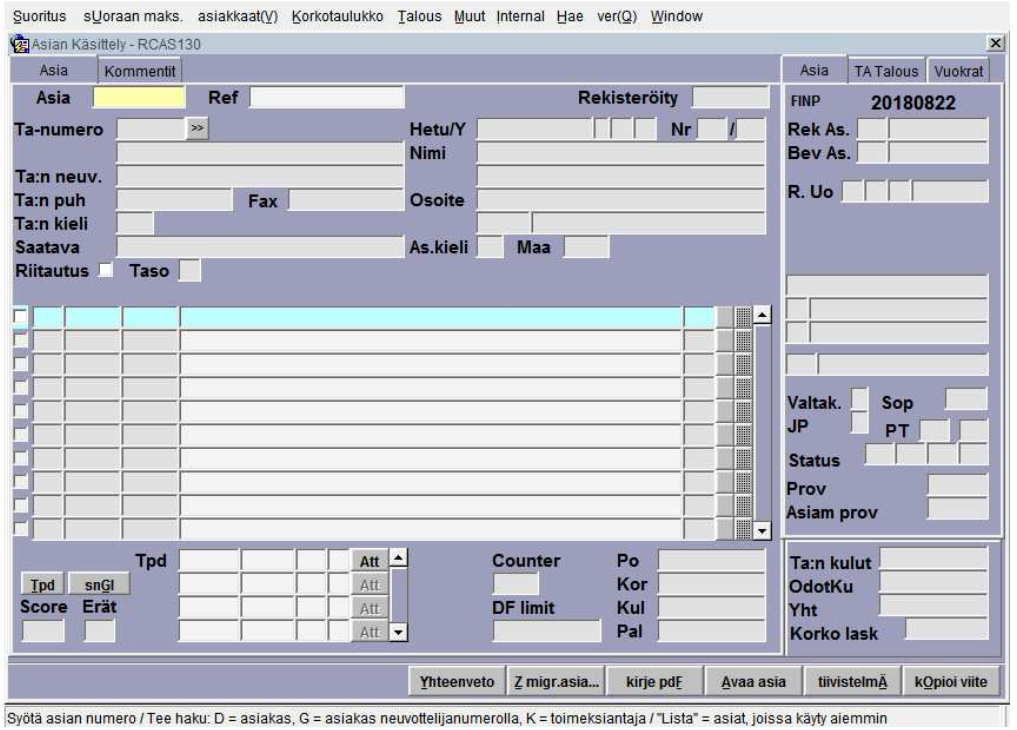
Forms Application

Intrum debt collection system

- Developed and maintained in-house
- Originally was developed in 90s
- In early 2000s was migrated from Forms 3 to Forms 9i
- A lot of legacy Pro*C and SQR code

… and some numbers….

- About 1000 tables
- 350 forms
- C/+SQR
- 2 500 000 cases
- Generate about 50 000 letters daily
- 350 users



intrum

Motivation

- Very difficult to maintain and implement new features
- Steep learning curve – 3-4 weeks for a new employee to start working
- Outdated UI and UX
- Stop Java support in browsers
- Difficult to automate UI testing

Project Goals

- Improve maintainability
- Modern web UI/UX
- Intuitive UI
- Test automation
- Continuous delivery

intrum

Why APEX

- Critical to use the same development team
- Easy to learn for Forms developers
- Web UI/UX
- Standard tools to for UI testing
- Free tools

intrum

Development process

intrum

Project schedule

Go live – beginning 2020

intrum

Project structure

Development team

- 8 developers, including 1 dedicated APEX developer
- 3 testers

Requirements team

- 8 functional owners
- Also responsible for acceptance testing

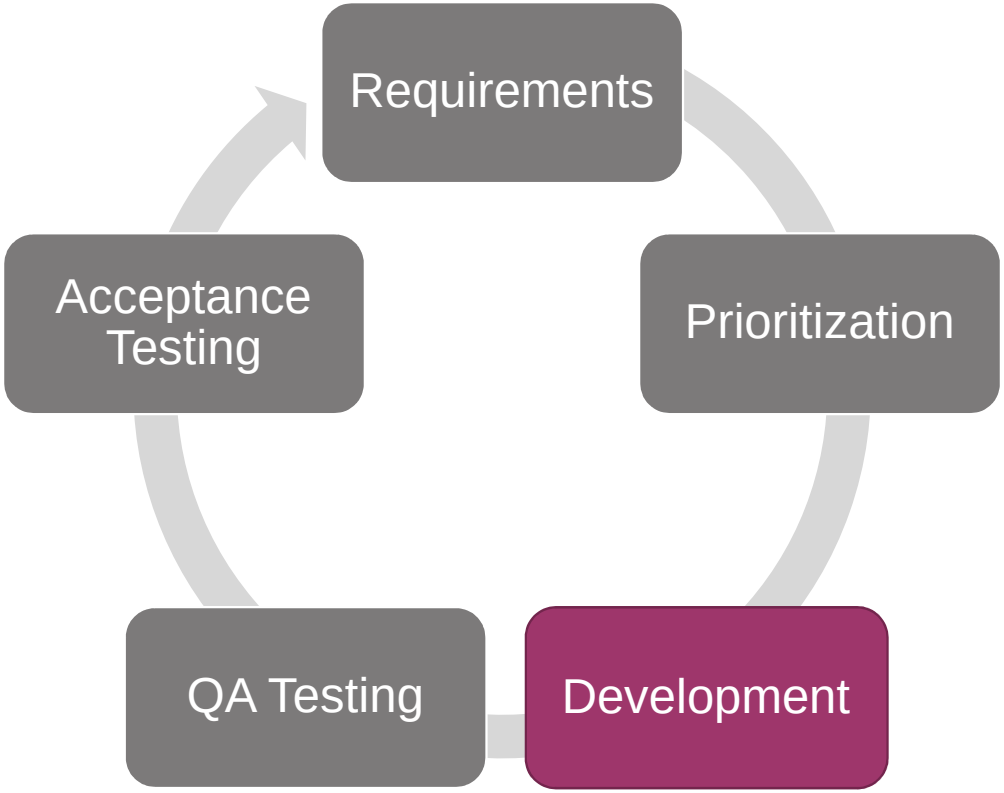
End Users team

- 20 end users
- “Release” testing

intrum

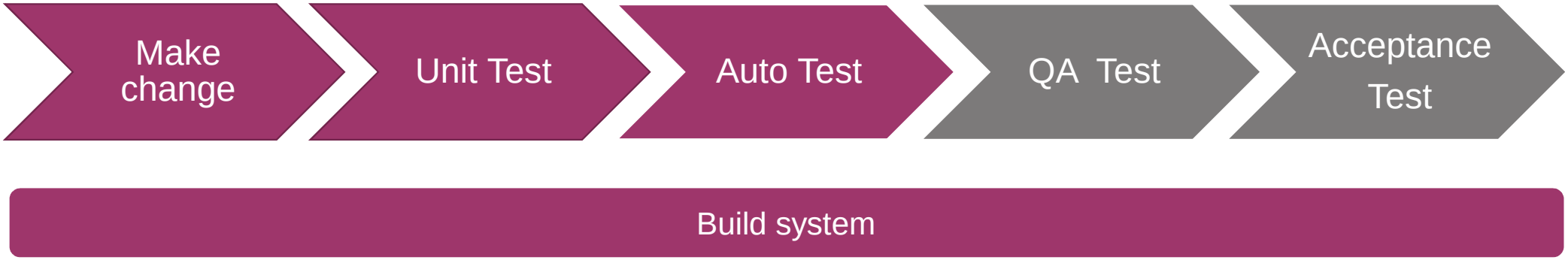
Development Process

- “Continuous Delivery” is one of the project goals
- Agile approach
  - 2 weeks sprint
  - 1 week delivery cycle to acceptance test environment
  - Fix bugs first



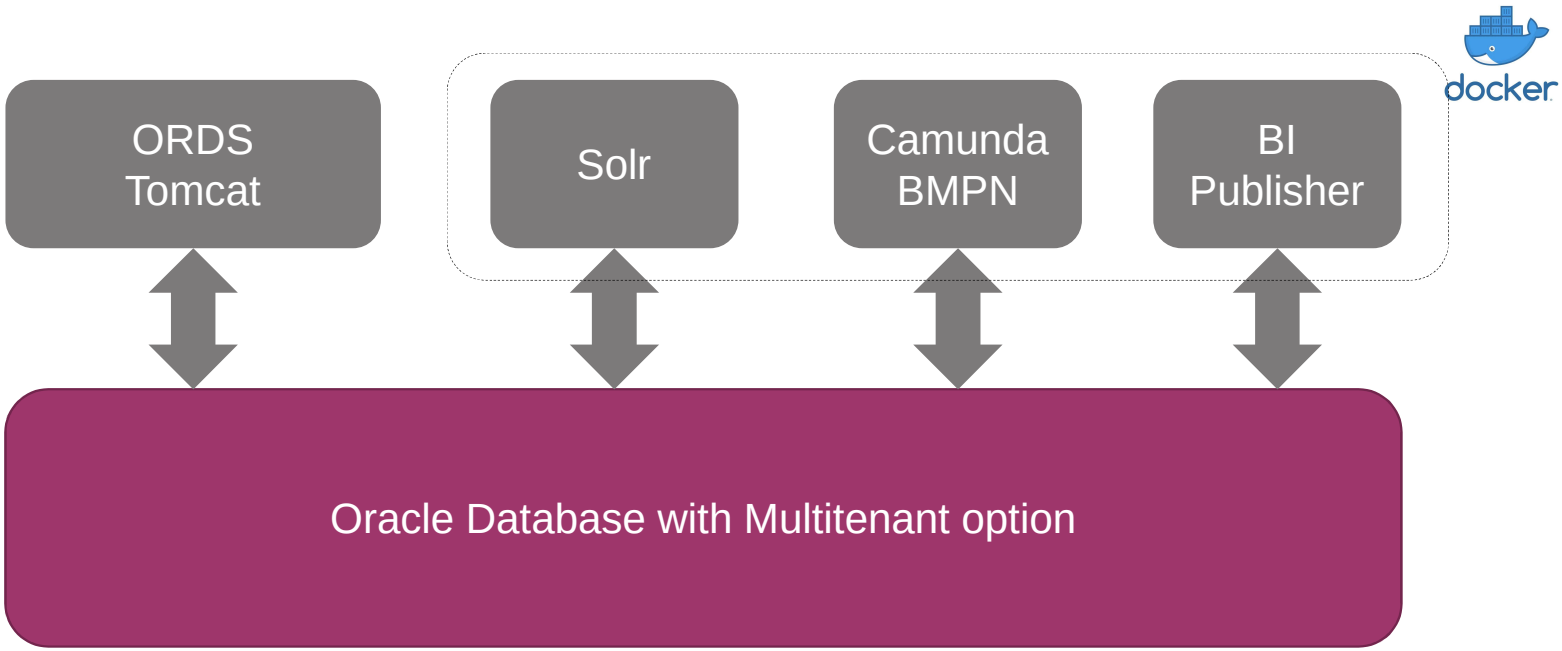
intrum

Development Workflow



intrum

Architecture



intrum

Environments

	Title	Description	Persistance/Builds
DEV	Development	Development environment physical container**. Environment is recreated daily.	Created as a copy of DEV_KNIGHT.
DEV_AT	Regression Testing	Environment for regression testing.	Created as a snapshot from DEV after night build.
DEV_KNIGHT	Build	Build is performed on this environment. Later this environment is used for copying latest build. Environment is built from latest master.	Has initial DB setup (clean DB).
DEV_KNIGHT_UT	Build Unit Tests	Unit testing environment after build.	Created as a snapshot of DEV_KNIGHT after build.
DEV_APEX	Apex Development Sandbox	Development environment for UI changes in Apex. All changes from this environment are automatically exported to version control trunk before night build.	Created as a copy of DEV after night build.
DEV_NSU_A*	Automated Developer's Sandbox	Auto-refreshable developer's sandbox environment, used to produce code changes and unit-tests. Typically used for quick fixes.	Created as a snapshot from DEV after night build.
DEV_NSU_M*	Manual Developer's Sandbox	Manually-refreshable developer's sandbox environment, used to produce code changes and unit-tests. Typically used for longer developments.	Created as a copy from DEV manually.
TEST	Testing	Testing environment physical container**.	Copied from DEV environment nightly, after build and auto-tests succeeded.
TEST_1..N	Testing sandboxes	Test environment for running automated integration/system tests. All integration and system tests executed automatically after environment is created.	Created as a snapshot from TEST after it is created. Can be rebuilt on request with release candidate tag.
QA	QA	QA environment physical container**.	Copied from DEV environment on request.
QA_1..N	QA Sandboxes	Environments to execute functional (manual) tests and fix verification by Test Engineers.	Created as a snapshot from QA manually. Can be rebuilt on request with release candidate tag.
UAT	User Acceptance Testing	UAT environment physical container**.	Copied from DEV environment on request.
UAT_1..N	UAT Sandboxes	Environments to execute acceptance tests of release candidates by Acceptance Testers (Business development, Super users). This environment is also used for external testing with clients and partners.	Created as a snapshot from UAT manually. Can be rebuilt on request with release candidate tag.

intrum

Build/Tools

Jenkins

Gradle (custom scripts)

Git

SQLcl
APEX exp/imp

Docker

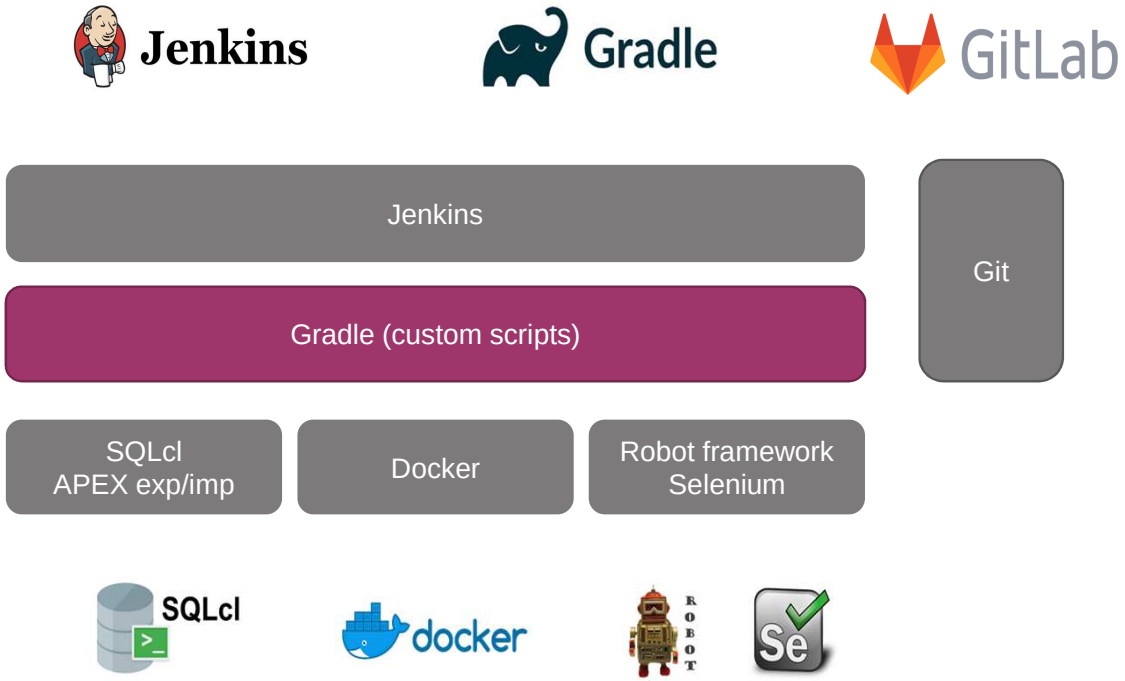
Robot framework
Selenium

intrum

Build/Tools

Convention over configuration

- All automation is done by Gradle tasks
- Developers can run all tasks locally
- Jenkins automates build&deploy processes
- Automate as much as possible. Manual work must be minimized
- Use standards and conventions. Minimized configuration file usage
- Standards must be validated automatically



intrum

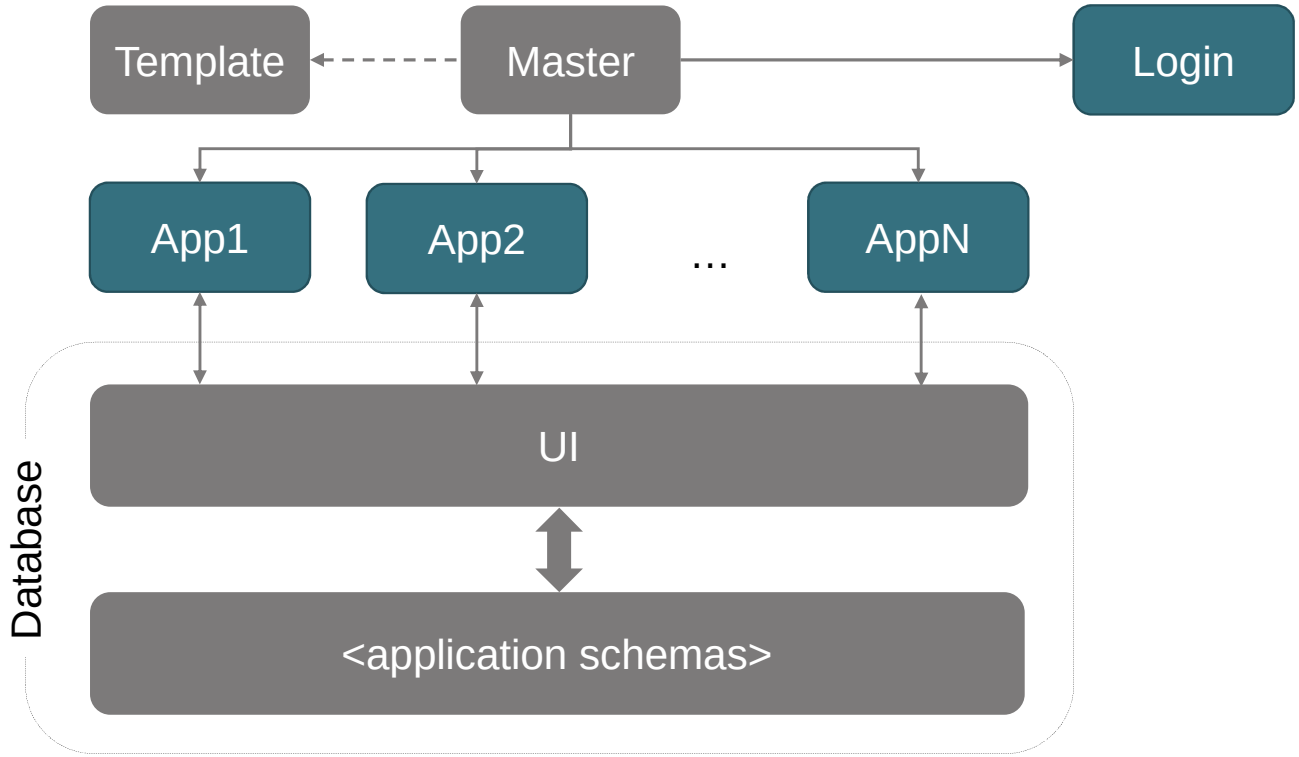
APEX

APEX

- APEX application architecture
- Version control and build process
- Testing
- Standards
- Issues

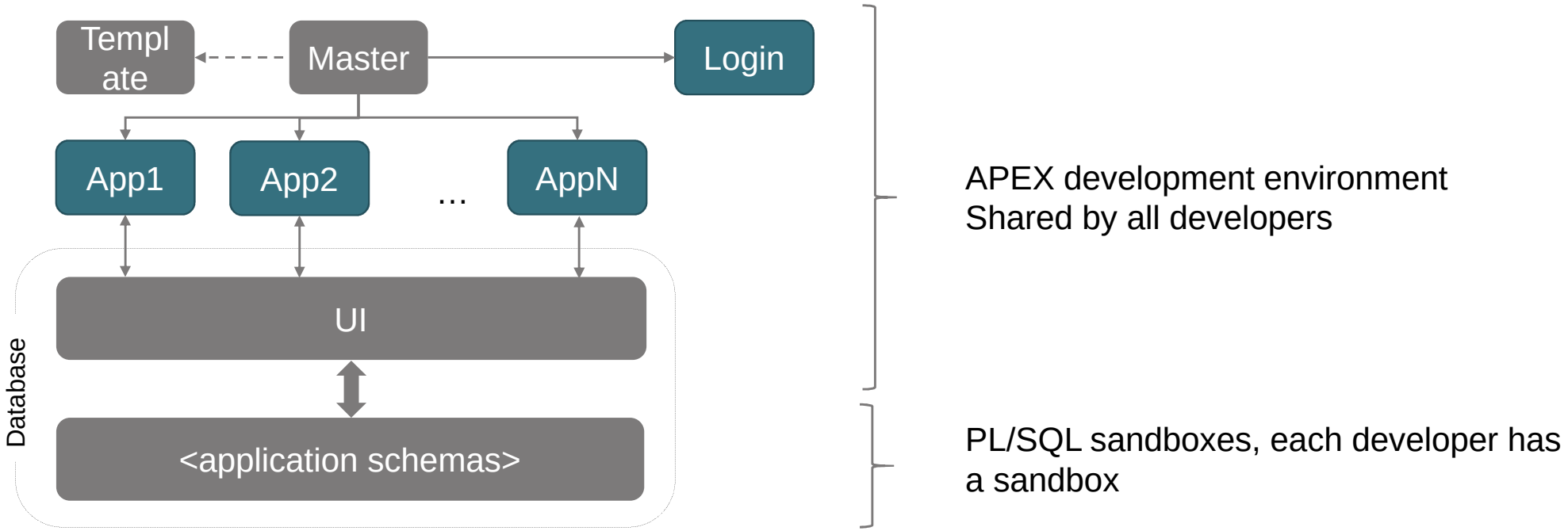
intrum

APEX Architecture



intrum

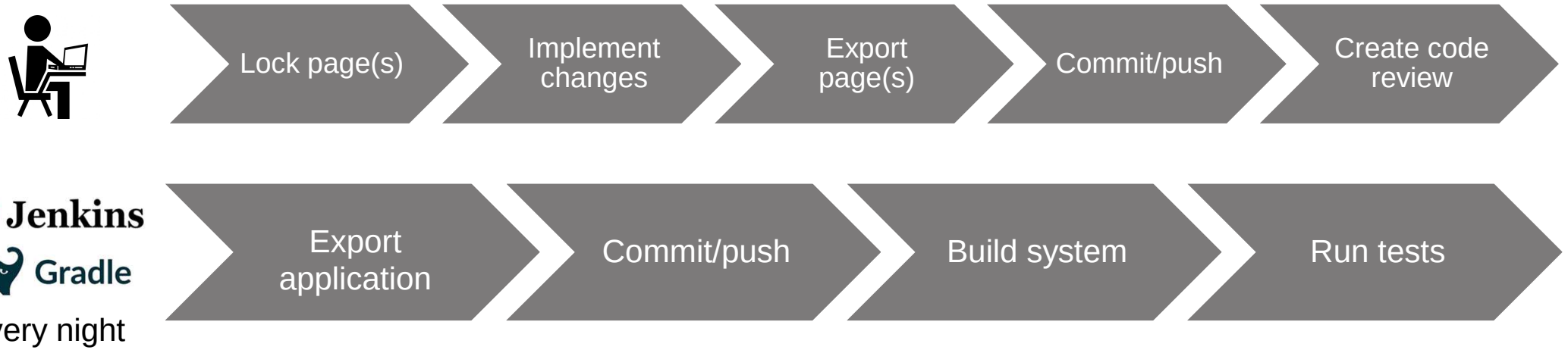
APEX Architecture



intrum

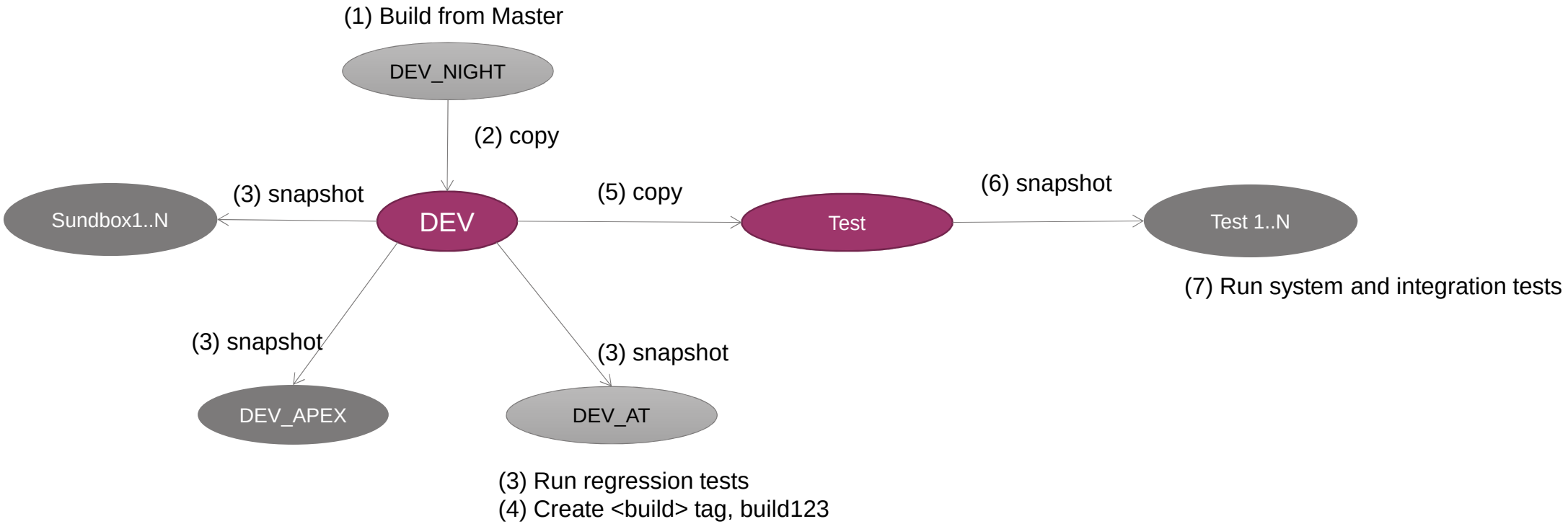
Version control and build process

All developers are working in the same database



intrum

Environments and build process



intrum

Testing

Robot framework with Selenium

- Unit – automatic/manual
- Regression – automatic
- System/Integration – automatic
- QA – manual
- Acceptance – manual

intrum

Standards

Database Naming Standards

Database Design Standards Standards

PL/SQL Standards

APEX Naming Standards

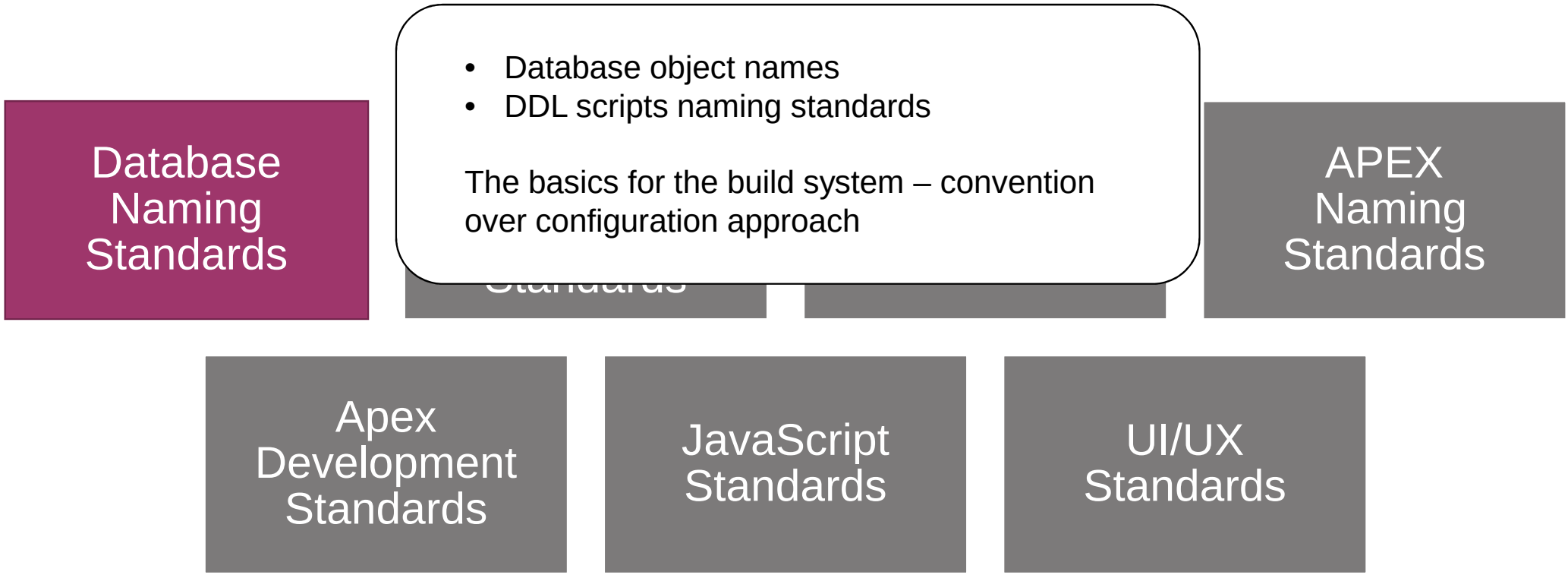
Apex Development Standards

JavaScript Standards

UI/UX Standards

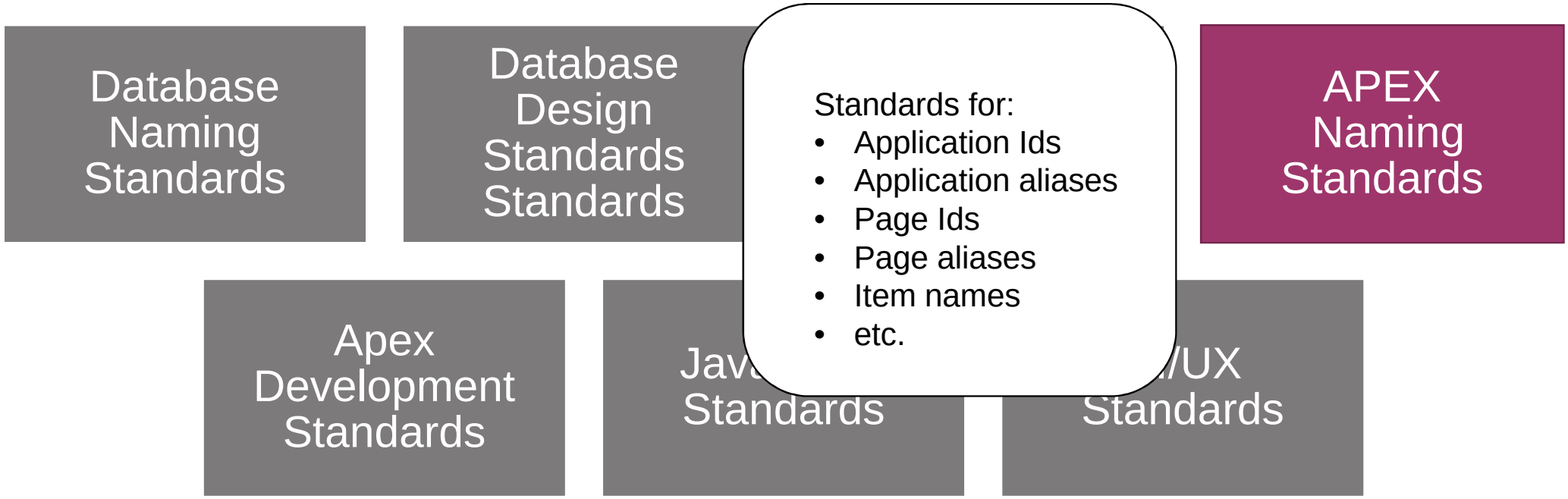
intrum

Standards



intrum

Standards



intrum

Standards

Database Naming Standards

Database Design Standards

PL/SQL Standards

APEX Naming

Apex Development Standards

- Put PL/SQL code to the packages, create a “page” package for PL/SQL code
- Reference only UI schema objects in the pages
- Put JavaScript code to the external files
- Use static ids from all items – needed for test automation

intrum

Standards

intrum

Standards. Examples

PL/SQL package - <prefix>_<name>[_<postfix>] => cli_client_api

- api – API level, build system generate GRANT EXECUTE TO UI

Page package - pge_<application allias>_<page_id>

JavaScript - <application allias>_<page_id> and the same namespace

Standards enforced by:

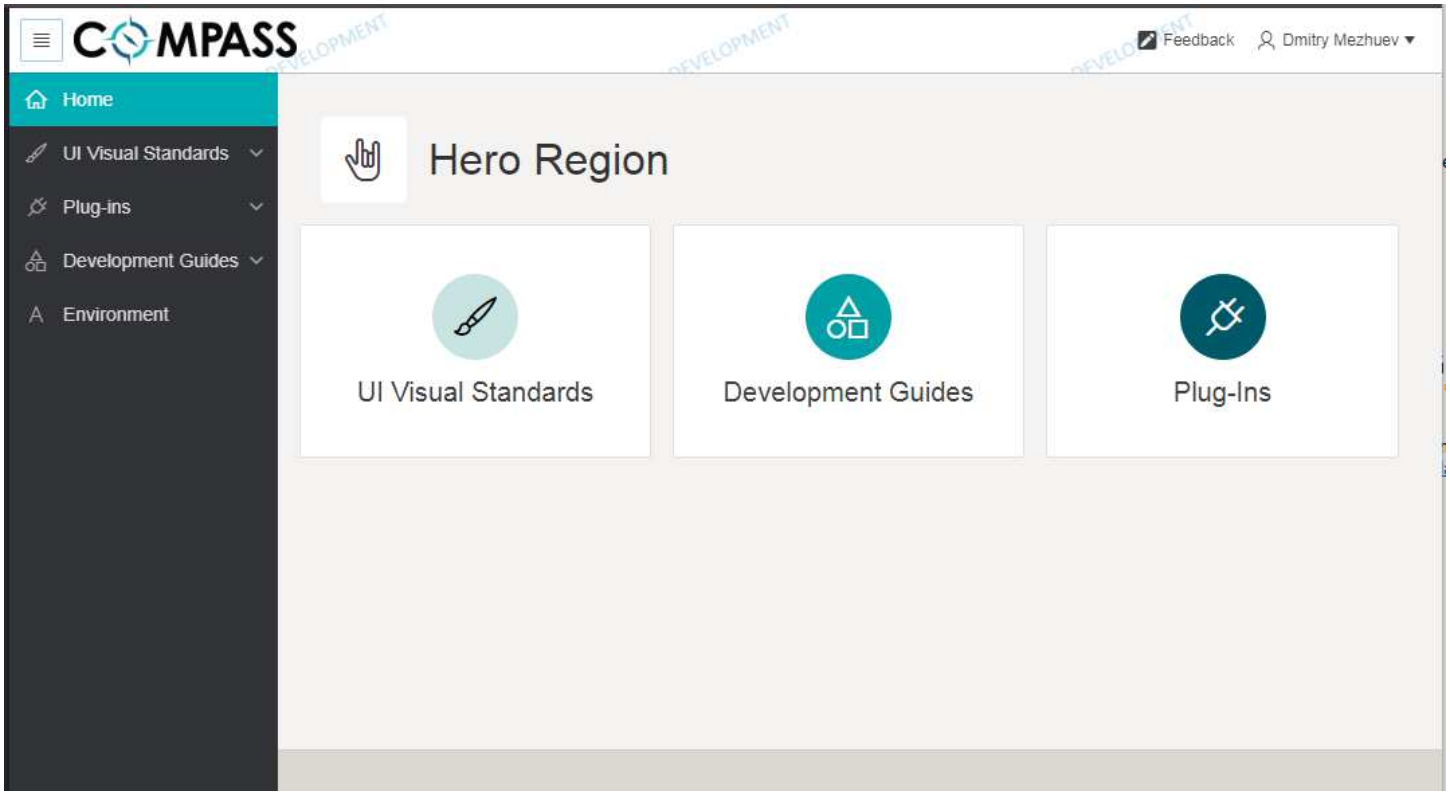
- DLL triggers on the database level
- Git hook scripts
- Code reviews

intrum

UI/UX standards

- Visual standards – custom CSS
- UX standards
  - Navigation
  - Tables
  - View forms
  - Edit forms
  - Master-detail
  - Pop-ups
  - Data formats
  - Error handling
  - Etc.
  - Standard UI scenarios

intrum


COMPASS
Feedback
Dmitry Mezhuev
Home
UI Visual Standards
Plug-ins
Development Guides
Environment
Hero Region
UI Visual Standards
Development Guides
Plug-Ins

intrum

Working with UI/UX experts

- They know nothing about APEX and it limitations
- Forms/Oracle developers do not have experience working with UI/UX experts
- We spent 3-4 month only to start talking the same language

intrum

Other issues

- Printing
- Version control
- API documentation

intrum

Summaty

Good	Issues
Low code	Version control
Easy to learn	APEX 5.1 lack of the API documentation
Rapid development	Printing
Prototyping	Dynamic forms, staff like editing in Report (Interactive greed?), copy regions
Integration wit JS libraries	
REST integrations	

intrum

Q&A


intrum