Attn.:	Allevo
From:	Denisa Dinca
Subject:	FinTPc AT - architecture

allevo
THINKING EVOLUTION

Software Architecture
Project: TOSS Automating Testing Tool
Release: FinTPc AT
Version: 3.0

11.03.2020

Contents

11.03.2020

Attn.:	Allevo
From:	Denisa Dinca
Subject:	FinTPc AT - architecture

allevo
THINKING EVOLUTION

Document Control

Title	Software Architecture
Project	TOSS Automating Testing Tool – FinTPc AT
Version	3.0
Creation Date	2 Oct 2017
Author	Denisa Dinca

Version	Date	Short description	Author
1.0	4 Oct 2017	Document structure	Mihai Guiman
1.0	30 Oct 2017	Chapter1	Ioana Guiman
1.0	30 Oct 2017	Chapter2	Sabin Breazu
1.0	30 Oct 2017	Document review	Denisa Dinca
2.0	29 Nov 2017	Chapter2.2 Chapter2.3	Mihai Guiman
2.0	29 Nov 2017	Chapter2.1.4	Denisa Dinca
2.0	29 Nov 2017	Chapter2.1/.1/.2	Ioana Guiman
2.0	29 Nov 2017	Chapter2.1.3	Sabin Breazu
3.0	15 Dec 2017	Document review	Denisa Dinca
3.0	15 Dec 2017	Document review	Sabin Breazu

11.03.2020

1 Introduction

FinTPc AT is a functional extension of FinTPc application aimed to automate testing its functionalities. It assures quick and complete testing and it is useful to be run after configuration changes or major releases and patches.
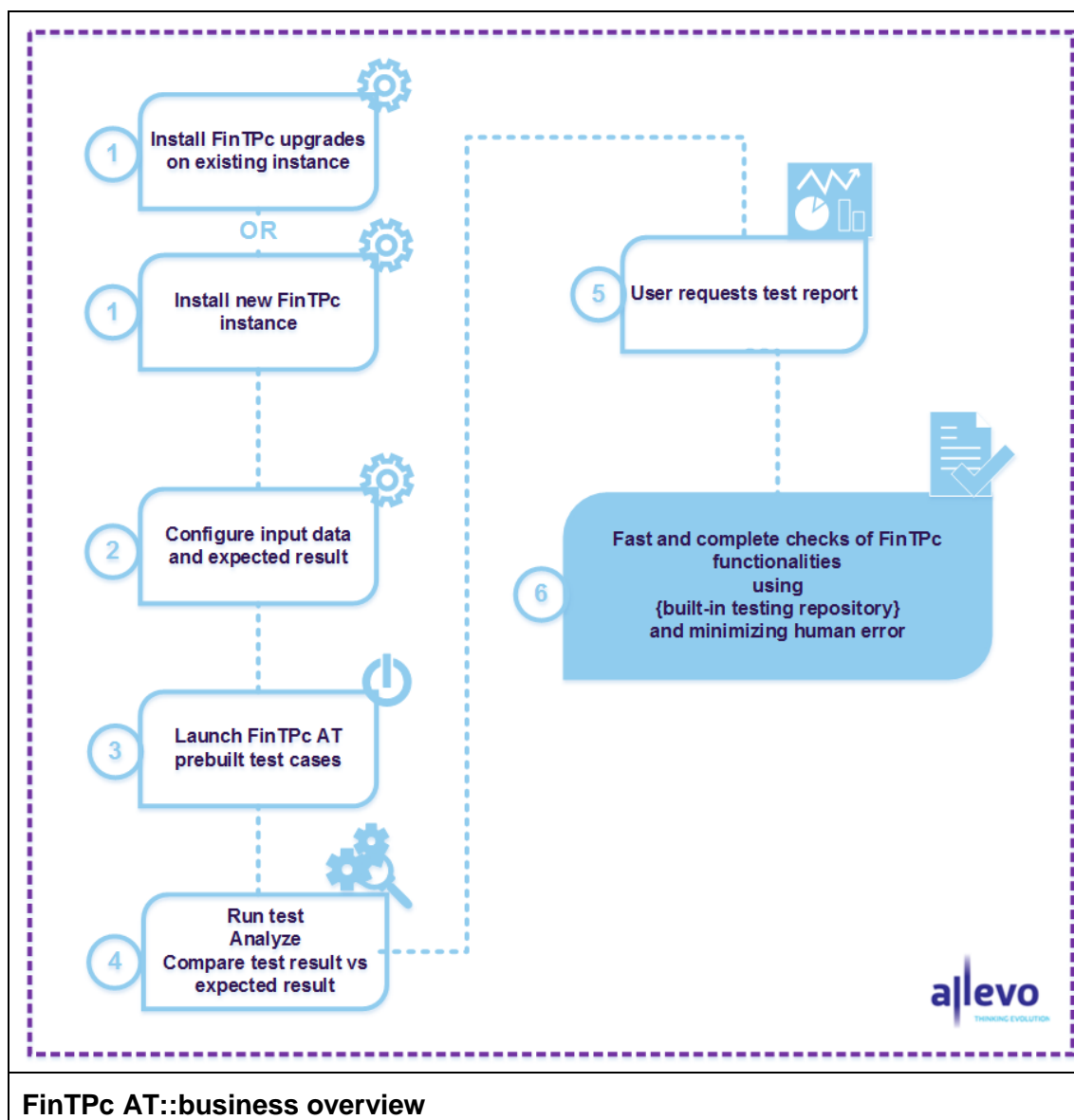
FinTPc AT::business overview

Pag 4 / 15

Finanțat în cadrul POC, Axa prioritară 2, Acțiunea 2.1, Prioritate de investiții 2b,
Cod MySMIS: 115724, Nr. Contract Finanțare: 101/16.08.2017

11.03.2020

1.1   Intended audience

Architects

Developers

Testers

Implementers

Sales and Marketing

1.2 Project background

FinTPc AT is a project that brings together automated predefined test cases to be used by testers (internal or external) and also unit tests to be used by developers.

FinTPc AT allows:

- Testing configuration for:
  - All financial instructions supported by FinTPc application;
  - Specific business flows;
  - Data sets corresponding to each interfacing external system;
  - Test scenarios;
- Monitors and controls the testing execution
- Reports and records testing results
- Offers support in investigating faults and diagnosis;
- Test results may be exported along with technical context;
- Helps validation fresh installations or monitoring confidence testing of existing productive installation;

Notes

- Must be periodically updated for compliance with the new regulations and the new applicable financial standards;
- Depending on the architecture of each corporation, must allow easy installation and configuration on any FinTPc platform - production, backup, pre-production or testing;
- FinTPc AT is an open source application published on GitHub under the GPLV3 license as well as FinTPc;

11.03.2020

2   Architecture

The IEEE recommendation defines an architecture as the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution. Architectures represent the abstraction used to understand any system and also form the basis for a shared understanding to all its stakeholders.

Application architecture seeks to build a bridge between business requirements and technical requirements by understanding use cases, and then finding ways to implement those use cases in the software.

An architectural overview is aimed at providing a shared understanding of the architecture across a broad range of people including the developers, marketing, management and possibly potential end-users. An architectural overview is ideally produced early in the development lifecycle and serves as the starting point for the development. An architectural overview should be at a high level of abstraction. All the major functionalities and components of the architecture should be described but the descriptions may lack detail and precision as they often use natural language rather than formal notations.

2.1   Logical architecture

2.1.1   Overview

The main purpose of FinTPc AT is to offer a set of test cases designed to fulfil a particular set of requirements and specific business implementations that need to be run repeatedly on request in order to give quality assurance confidence on implemented business rules in FinTPc application. FinTPc AT is deployed as a module of FinTPc application.

FinTPc AT handles functional tests, integration tests and unit tests. All these are predefined, any change to the FinTPc application has impact on FinTPc AT test repository and generates updates.

Functional tests are built based on the specification document that identified those test cases that make sense being automated, following a black box testing method. These are customizable and follow specific already implemented business flows. Each scenario is testing an individual part of the processing flow (it may involve one or more FinTPc components). AT allows defining input data (also referred as test data generation) and expected results based on a predefined list in its repository for each of these functional tests that have well defined points for input and output data. It all ends with a validation phase that analyzes the output data of the test and the configured expected result summarized in a log (that may also include technical errors occurred).

Integration tests represent a logical chaining of more functional tests that validate a whole business flow. This validation ends at first error encountered, continuing the test would not assure any consistency.

Unit tests are predefined for each logical component of FinTPc and are addressed to the developers of the application.

FinTPc AT consists of a repository of predefined sample test data and test templates and unit tests that are to be configured by the user before running them. This test automation framework must be highly modular (using a black box approach) and maintainable.
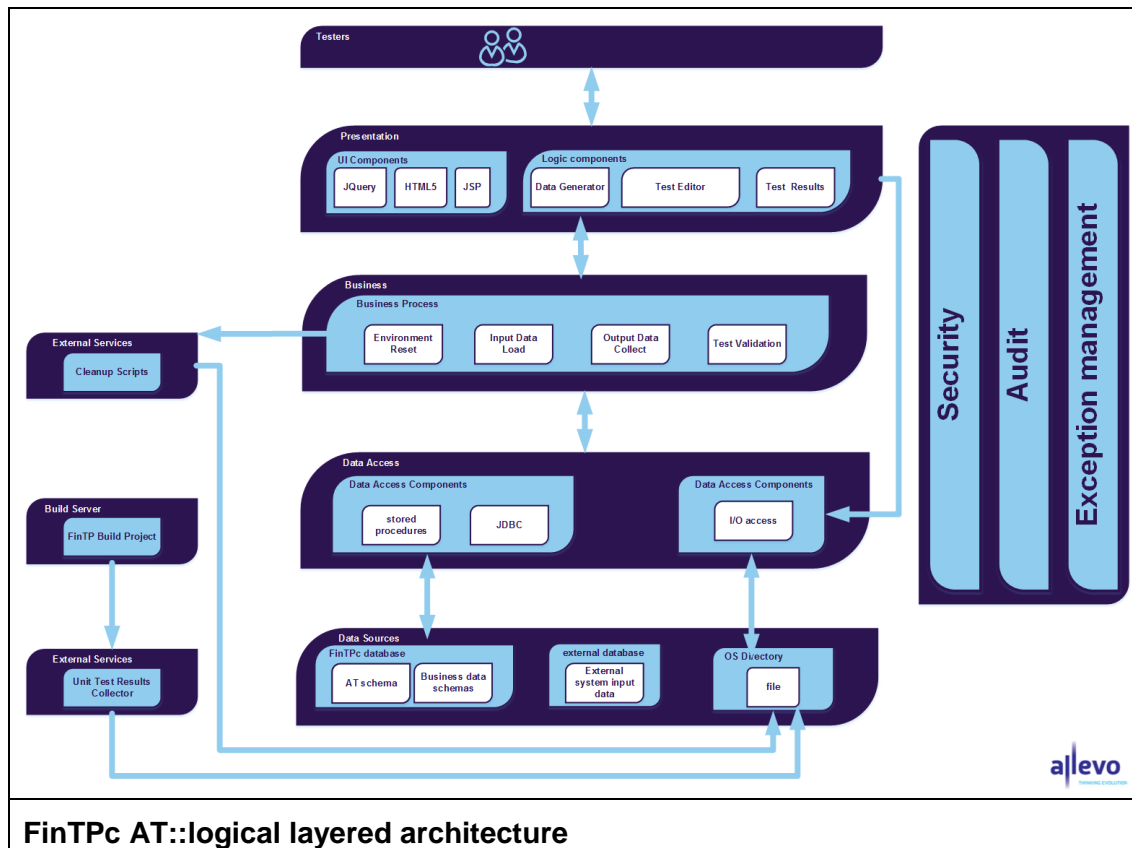
11.03.2020

2.1.2 Logical layered architecture

FinTPc AT::logical layered architecture

The upper diagram describes a layered architecture style of the project.

The components of the *Presentation* layer access the information via the *Data Access* layer in order to be available on user request. The mentioned UI components are used to display information and also accept user input. Users may configure test data generation, edit test cases and view test results – all these implemented by logical components Data Generator, Test Editor and Test Results.

The *Business* layer components represent the core functionalities of the system and encapsulate business logic. The business process is built base on functionalities like Environment Reset, Input Data Load, Output Data Collect or Test Validation. These functionalities depend on defined test cases that may include a financial instruction processing flow or User Interface business configuration functionality. This layer has access to information and saves information communicating to the *Data Access* layer.

The *Data Access* layer provides different ways of retrieving and sending information from and to the data sources. All the configurations are saved into specific database schema. Tests are run against data from business data schemas, so the

11.03.2020

Business layer communicates to both sides of database. Also Input Data Load and Output Data Collect may push or request data to / from external (to FinTPc) data sources, those that interface with FinTPc.

External Services layer includes one Unit test collector – grouping component specific unit tests that are run by developers on a build server and also includes a collection of Cleanup scripts called by Environment reset functionality.

The cross cutting concerns identified are the following: Security, Audit and Exception Management. These will be addressed on most layers.

2.1.3 Test cases design

2.1.3.1 Sample test data

FinTPc AT module offers a repository of relevant sample test data specific to given implementation. This data is designed in order to achieve relevant test data that makes possible the testing process of each scenario, given its divers content and also its volume.

Sample test data may be prepared into specific sets, given its purpose and content, so that the process of choosing the right set of data when defining a test case becomes easier.

The sample test data repository contains following categories of inputs:

* Financial transaction data files for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (payments, invoices), formatted according to corporate internal format specifications; These shall simulate corporate applications input to FinTPc;

* Database script files that generate financial transaction data for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (payments, statements, invoices), according to corporate internal storage format specifications; These shall simulate corporate applications input to FinTPc;

* Financial transaction data files for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (statements), formatted according to bank standards format specifications; These shall simulate banking applications input to FinTPc;

For each of the inputs described above, there is a related expected output in the repository:

- ✳ Financial transaction data files for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (payments, invoices), formatted according to banking format standards; These shall serve to validate FinTPc's output for banking applications;

- ✳ Financial transaction data files for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (statements) formatted according to corporate internal format specifications; These shall serve to validate FinTPc's output for internal corporate applications;

- ✳ Database script files that generate financial transaction data for each type of transaction identified in the business analysis phase of FinTPc and specific to client's implementation (statements), according to corporate internal storage format specifications; These shall serve to validate FinTPc's output for internal corporate applications;

2.1.3.2 Test templates

FinTPc AT provides a set of predefined tests based on the ones identified in the analysis phase to be relevant to the overall application testing process and that need to be automated. From an architectural perspective, these tests are divided into two categories and further implemented differently:

- ✳ *FinTPc User Interface tests category* - those that aim to test and access the FinTPc User Interface functionalities and need to simulate user interaction;

These test are to be implemented by building separated Unit Test modules available in the FinTPc AT testing repository.

- ✳ *Transaction processing tests category* - those that aim to test the transaction processing mechanism of FinTPc, without any access to the User Interface;

These tests are to be implemented as a collection of test templates accessible and configurable from the FinTPc AT module's User Interface. They will test the financial

11.03.2020

transaction processing mechanism of FinTPc and this way also the well-functioning of all the components in this flow. Every test is designed to be launched manually and shall run independently of others, one at a time, on a clean environment, the simultaneously running of other tests being restricted. This constraint ensures a non-altered test result and reduces additional checks.

Acceptance tests are to be configured and run using the same collection of test templates as the ones above.

The implementation of test templates is based on first mapping the tests described in the specification document on a model that makes it easier and certainly to identify what the user may customize in a test template, what are the constraints, what other additional checks have to be made. Doing this preliminary step also ensure an atomic design by eliminating or dividing test templates into individual non repeating ones.

The result of a given test is listed in a log file.

Test name	Test description		Configurations	Implementation
Given	[context checks] *specific test initializations; *up and running FinTPc tested component checks; *input / output is configured; *data is loaded or the test loads it; *processing flow constraints are met;	*sources and destinations *sample data – input and expected result *result waiting timeout	*cleanup scripts; *input data load; *output data collect; *test validation; *build server; *unit tests result collector;
When	[action run] *action is performed;		
Then	[expected changes check] *read ad validate output;		

11.03.2020

2.1.4   Component interaction model

The diagram below shows the basic interaction actions between the identified application components testing a specific scenario. The sample scenario consists of testing the first step from an end to end flow of processing payment transactions - collected from an application (used by the treasury department of a given corporation) and exported for other banking applications. This step consists of collecting the payments from the back office application and processing them and then held them into an exit queue for user decision (whether to authorize them).

For the complete description of the flow, see the *FinTPc architecture* document.
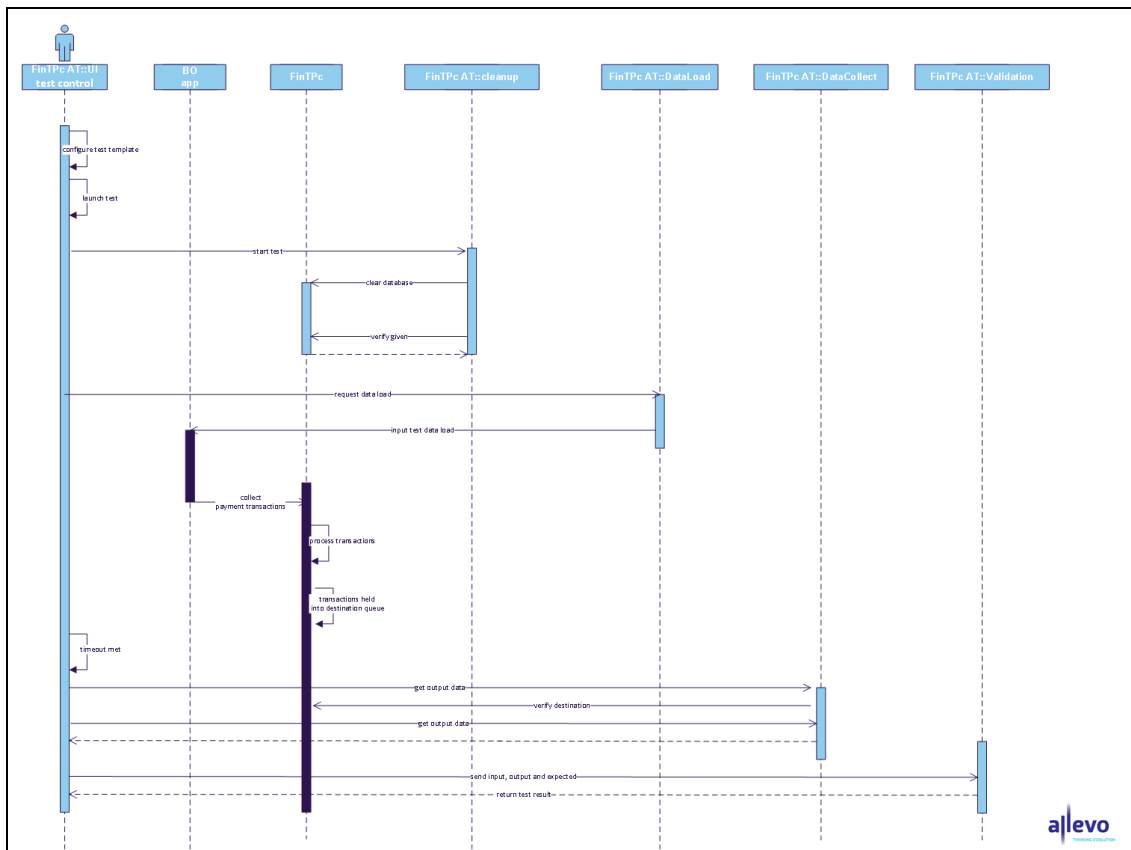
FinTPc AT:: UML sequence diagram – component interaction model

2.2 Physical architecture

FinTPc AT has to be able to run on all architectures[1] supported by FinTPc, being designed and built as a module of the main application.
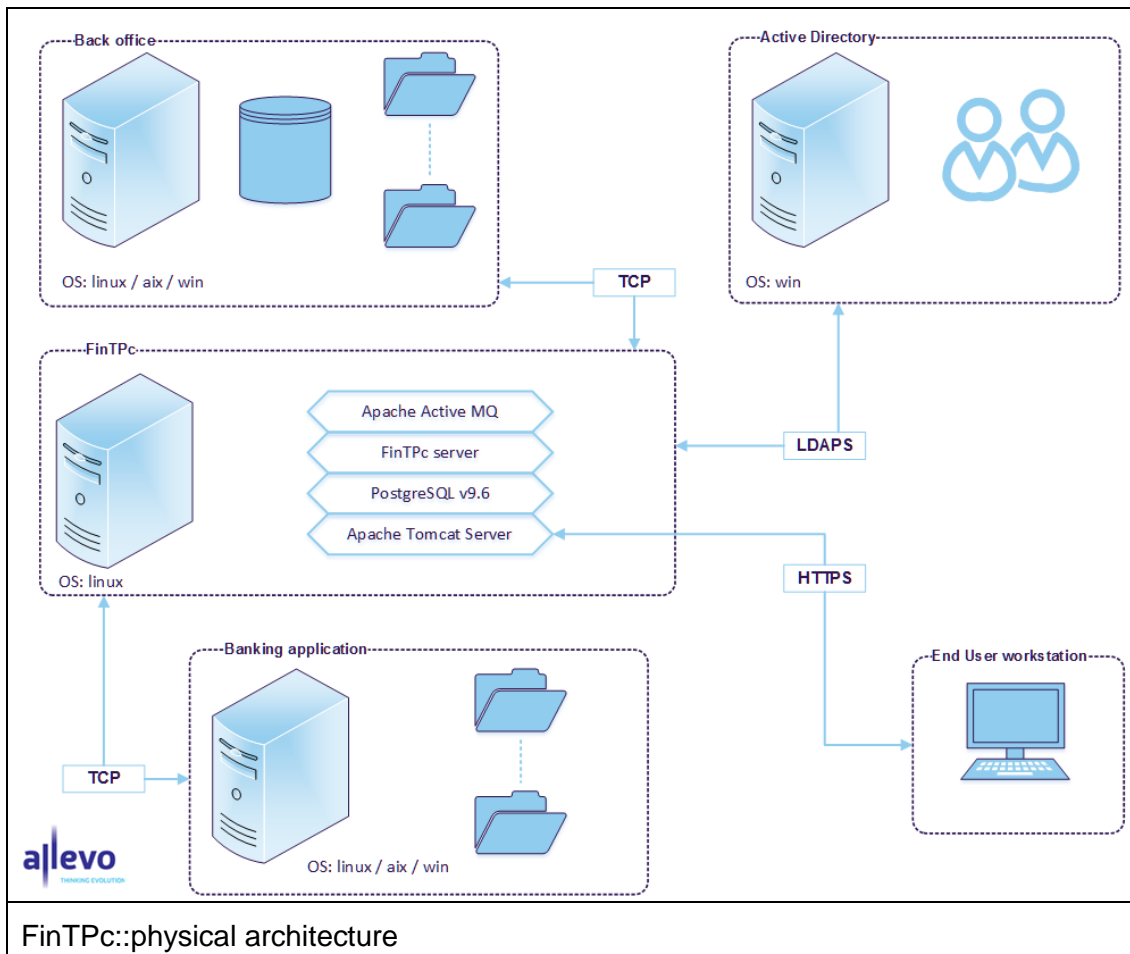
FinTPc::physical architecture

---

[1] FinTPc's physical architecture is detailed in the *FinTPc – architecture* document.

11.03.2020

2.3 Technology selection

The distribution model of FinTPc project is open source. Therefor the major architectural and technological constraint is represented by the compliance of FinTPc code and any other embedded product or library with GPL v3 license model. The design and implementation stage will include also advanced scanning procedures in order to be able to certify this license compliance.[2]

FinTPc AT has to be compliant to the same technological constraints already covered by FinTPc, being designed and built as a module of the main application.

---

[2] For a complete description, see the FinTPc – architecture documents.

11.03.2020