Footwear quality tool

Functional documentation

CONTENT

[CONTENT 1](#_Toc107074041)

[ILLUSTRATIONS 3](#_Toc107074042)

[OVERVIEW 5](#_Toc107074043)

[1 Introduction 6](#_Toc107074044)

[1.1 Machines 6](#_Toc107074045)

[1.1.1 Profilometer 6](#_Toc107074046)

[1.1.2 Checkweigher 6](#_Toc107074047)

[1.2 Engineering department 7](#_Toc107074048)

[1.3 PTC Cloud Database 7](#_Toc107074049)

[1.4 Application front-end 8](#_Toc107074050)

[1.5 Application external connections resume 9](#_Toc107074051)

[2 Application 9](#_Toc107074052)

[2.1 Data model 10](#_Toc107074053)

[2.1.1 HP. Footwear.ProgramsTable 12](#_Toc107074054)

[2.1.2 Hp. Footwear.ProgramsVersionsTable 12](#_Toc107074055)

[2.1.3 HP. Footwear.JobsTable 12](#_Toc107074056)

[2.1.4 HP.Footwear.ProfilometerModelDS 12](#_Toc107074057)

[2.1.5 HP.Footwear.PurchaseOrderDS 12](#_Toc107074058)

[2.1.6 HP.Footwear.ShipmentPORelationDS 12](#_Toc107074059)

[2.1.7 Hp.Footwear.PartsBarcodeTableDS 12](#_Toc107074060)

[2.1.8 HP.Footwear.VisualInspectionPartsTableDS 12](#_Toc107074061)

[2.1.9 HP.Footwear.CheckweigherPartsTableDS 12](#_Toc107074062)

[2.1.10 HP.Footwear.CheckweigherPartsTableDS 12](#_Toc107074063)

[2.2 Kepware connection 12](#_Toc107074064)

[2.2.1 Profilometer 12](#_Toc107074065)

[2.2.2 Checkweigher 13](#_Toc107074066)

[2.2.3 Thingworx connection 14](#_Toc107074067)

[2.3 Application workflow 15](#_Toc107074068)

[2.3.1 Profilometer 15](#_Toc107074069)

[2.3.2 Checkweigher 18](#_Toc107074070)

[2.3.3 Program manager 20](#_Toc107074071)

[2.3.4 Job viewer & Export 22](#_Toc107074072)

[2.4 Application functionalities 24](#_Toc107074073)

[2.4.1 Profilometer 24](#_Toc107074074)

[2.4.2 Checkweigher 27](#_Toc107074075)

[2.4.3 Program manager 29](#_Toc107074076)

[2.4.4 Job viewer & Export 31](#_Toc107074077)

[2.5 CSV Export 32](#_Toc107074078)

[2.5.1 Profilometer 32](#_Toc107074079)

[2.5.2 Checkweigher 32](#_Toc107074080)

[2.6 Database comunication 33](#_Toc107074081)

1. ILLUSTRATIONS

[Figure 1 - Profilometer PLC Configuration 6](#_Toc107074082)

[Figure 2 - OPC communication configuration 6](#_Toc107074083)

[Figure 3 - PLC configuration 6](#_Toc107074084)

[Figure 4 - Database entity name and configuration 7](#_Toc107074085)

[Figure 5 - Built-in services that we use 8](#_Toc107074086)

[Figure 6 - Mashup example. Main screen of Footwear application 8](#_Toc107074087)

[Figure 7 - Login pop up 8](#_Toc107074088)

[Figure 8 - Application external structure 9](#_Toc107074089)

[Figure 9 - HP.Footwear project and elements 10](#_Toc107074090)

[Figure 10 - Latest version in production 10](#_Toc107074091)

[Figure 11 - Footwear data model 11](#_Toc107074092)

[Figure 12 - Profilometer Tags from Kepware 13](#_Toc107074093)

[Figure 13 - Checkweigher Tag structure from OPC Kepware 13](#_Toc107074094)

[Figure 14 - Tags from Checkweigher Kepware 13](#_Toc107074095)

[Figure 15 - Tag values from the Quick Cliente application 13](#_Toc107074096)

[Figure 16 - Kepware Thingworx configuration 14](#_Toc107074097)

[Figure 17 - Industrial Connection 14](#_Toc107074098)

[Figure 18 - Remote Thing Configuration 14](#_Toc107074099)

[Figure 19 - Figma of the application descriptive visual document 15](#_Toc107074100)

[Figure 20 - Profilometer Screen 16](#_Toc107074101)

[Figure 21 - New Inspection Pop Up 16](#_Toc107074102)

[Figure 22 - Application ready to receive Barcodes 16](#_Toc107074103)

[Figure 23 - Visual Inspection Pop Up 17](#_Toc107074104)

[Figure 24 - New line created from a measurement 17](#_Toc107074105)

[Figure 25 - Stop Inspection PopUp 18](#_Toc107074106)

[Figure 26 - Checkweigher screen 18](#_Toc107074107)

[Figure 27 - New Inspection pop up 19](#_Toc107074108)

[Figure 28 - Application waiting for weight 19](#_Toc107074109)

[Figure 29 - Application updated with the weight result 19](#_Toc107074110)

[Figure 30 - Stop Inspection PopUp 20](#_Toc107074111)

[Figure 31 - Program manager screen 20](#_Toc107074112)

[Figure 32 - New program Pop Up 21](#_Toc107074113)

[Figure 33 - New version screen 21](#_Toc107074114)

[Figure 34 - Model manager screen 21](#_Toc107074115)

[Figure 35 - Update model pop up 22](#_Toc107074116)

[Figure 36 - Programs manager screen 22](#_Toc107074117)

[Figure 37 - Program dropdown menu 23](#_Toc107074118)

[Figure 38 - Job dropdown menú 23](#_Toc107074119)

[Figure 39 - File download 23](#_Toc107074120)

[Figure 40 - Footwear mashups and services 24](#_Toc107074121)

[Figure 41 - Infotable containing the programs 25](#_Toc107074122)

[Figure 42 - Footwear repository 26](#_Toc107074123)

[Figure 43 - Infotable containing the programs 27](#_Toc107074124)

[Figure 44 - Footwear repository 29](#_Toc107074125)

[Figure 45 - Toggle visibility button and visible property 30](#_Toc107074126)

[Figure 46 - Delete program pop up 31](#_Toc107074127)

[Figure 47 - Profilometer CSV Format 32](#_Toc107074128)

[Figure 48 - Checkweigher CSV Format 32](#_Toc107074129)

[Figure 49 - Insert information example 33](#_Toc107074130)

[Figure 50 - Query example 33](#_Toc107074131)

[Figure 51 - Update example 34](#_Toc107074132)

[Figure 52 - Delete example 34](#_Toc107074133)

1. OVERVIEW

This application is a solution implemented for the Footwear line in HP DFactory. The objective is to capture the information of the quality process after the part printing and to send this information to the engineering department.

The overall objective is to have a fully automated process in the shopfloor where the operator has a minimum interaction with the machines involved in the quality inspection. These machines are the profilometer and the checkweigher.

The current state of the project is a semi-automated process with both machines working independently and with the interaction of the operator to place the parts in both machines as well as selecting the programs in the checkweigher.

This application will be a part of a holistic solution called Operator Advisor, sharing the information with it and exchanging orders and data.

# Introduction

This section contains the information about the external elements of the application.

## Machines

The base of the project is both machines: Profilometer and Checkweigher as they will gather the information about the process. Both machines are currently separated but we expect they will be unified in the future.

### Profilometer

The profilometer machine is a custom tool created to localize the holes in the part and take the measurements of the hole-hole distance and send it or store it locally. This machine has an embedded PC that controls the measurement and selects the program to be compared. This program contains the specifications for the part that is been inspected. It has an actuator that rejects the part in case the specifications are not matched.

The machine is controlled by a Siemens PLC S7-1500:

Imagen que contiene Aplicación

Descripción generada automáticamente

Figure 1 - Profilometer PLC Configuration

### Checkweigher

The checkweigher machine is a tool created by Mettler Toledo with a custom PLC. This machine has a moving belt that has a separated conveyor belt integrated with a scale that weights the part. When the part is weighted it sends the information via OPC. It has an option to control the specifications as the profilometer. It has an actuator that rejects the part in case the specifications are not matched.

The communication is stablished via OPC protocol, and the custom PLC is unknown:

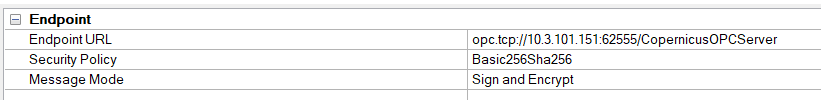


Figure 2 - OPC communication configuration

Tabla

Descripción generada automáticamente

Figure 3 - PLC configuration

## Engineering department

The engineering department is an important external connection since the processed info ThingWorx sends should be read properly through their system.

They are also in charge to modify the programs and specifications of the parts.

## PTC Cloud Database

This application works with Thingworx cloud database (PostgresSQL) located in PTC servers. The communication with this database is through the database entity.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figure 4 - Database entity name and configuration

This database entity has some built-in services that we should use to manage the database and update, add, remove any information we want to.

It is recommended to not use any user-created services unless is a test. Since this entity belongs to a different project and it isn’t deployed with our current project.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Figure 5 - Built-in services that we use

## Application front-end

The HMI part of the application are the Thingworx Mashups that creates the application front end. This front end is the visual part of the application and the part that interacts the user with. For the mashups to load it is necessary to have the server running.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 6 - Mashup example. Main screen of Footwear application

In order to log-in the user must enter its credentials:

Captura de pantalla de computadora

Descripción generada automáticamente

Figure 7 - Login pop up

## Application external connections resume

Once we have all the external connections explained the application external structure is a multiple input multiple output system that depends on: Operators, Database, Machines and returns a final CSV resume from the inspections.

Diagrama

Descripción generada automáticamente

Figure 8 - Application external structure

To have a functional system every part must be properly connected and running:

* Kepware should be connected to Thingworx.
* Thingworx must be communicating with the PTC Cloud service.
* The CSV extension must match the engineering system input format.
* The Front-End must be functional and free of blocking errors so the production does not get stopped or delayed.

To categorize the communications with external parts we can divide them in two blocks:

* Cloud communications: Database, Mashup System and CSV creation.
* Ethernet communications (local): Checkweigher and profilometer machines.

# Application

The application itself is a group of entities, data shapes, styles, mashups, and services that works together to bring a solution to the footwear quality process needs. All the objects that belongs to the application are related to the Thingworx project called **HP.Footwear**.

Every new element created for this project MUST be included here so the deployment to production is made properly.

Captura de pantalla de computadora

Descripción generada automáticamente

Figure 9 - HP.Footwear project and elements

The latest version is 1.0.63 on 23/06/2022.

Una captura de pantalla de una red social

Descripción generada automáticamente

Figure 10 - Latest version in production

The main objective of the application is to communicate with the machines and bring to Thingworx all the information from the inspections and to store it into the database for further analysis. As we have a complex data model, we need to process the information to give it the right format.

## Data model

The data model involves the tables in the Database and their structure. How we store the information is important and determine the way we will retrieve it in the future.

We have implemented a modular data model with the following tables:

* Visual Inspection, profilometer and checkweighter results tables.
* Jobs table and barcode tables.
* Programs and programs versions tables.
* Profilometer model table.
* Purchase order and shipment table.

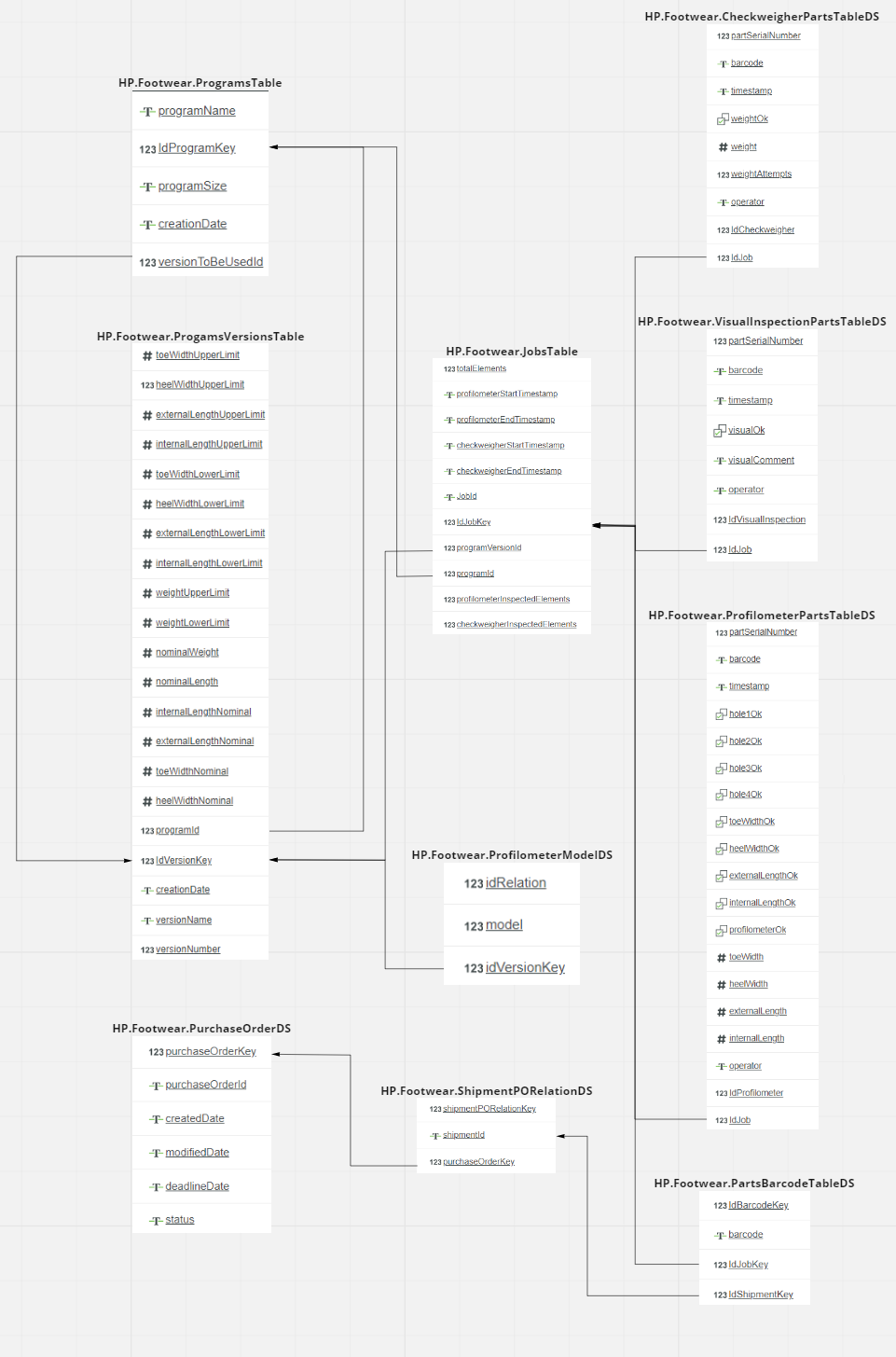


Figure 11 - Footwear data model

### HP. Footwear.ProgramsTable

Contains the program created by the engineer. This program does not contain specifications but just the name the operator will se on the inspection start.

### Hp. Footwear.ProgramsVersionsTable

This table contains the program version that CONTAINS the specifications. This will be also added by the engineer. A program can contain 0, 1 or more versions, and the version selected for the program will be stored in the HP.ProgramsTable under the field versionToBeUsedId.

### HP. Footwear.JobsTable

This contains the inspection jobs. Each job is related to a program and a version program from traceability.

### HP.Footwear.ProfilometerModelDS

This table contains the relation between a program version and the model number on the profilometer for the automatic mode where ThingWorx select the model the Profilometer will use.

### HP.Footwear.PurchaseOrderDS

Contains the information about a purchase order from a client.

### HP.Footwear.ShipmentPORelationDS

Contains the shipment information related to a purchase order.

### Hp.Footwear.PartsBarcodeTableDS

Contains the barcodes that are being scanned in the inspections. Barcodes are related to a Job and to a Shipment.

### HP.Footwear.VisualInspectionPartsTableDS

Contains the information from the visual inspection of the parts. Is related to a job.

### HP.Footwear.CheckweigherPartsTableDS

Contains the information from the checkweigher inspection of the parts. Is related to a job.

### HP.Footwear.CheckweigherPartsTableDS

Contains the information from the checkweigher inspection of the parts. Is related to a job.

## Kepware connection

To read from the machines we must stablish a connection first with the machines and then with ThingWorx. We have both types of connections: one from development to OPC aggregator and the other from production environment to Kepware.

### Profilometer

The profilometer is a Siemens PLC we have described in the introduction. After we stablish the connection we must create the tags with the addresses we need so we can read the information from there with the Siemens protocol.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Figure 12 - Profilometer Tags from Kepware

### Checkweigher

The checkweigher is a custom PLC and we need to communicate through OPC protocol with the PLC and the addresses are different from other Siemens PLC.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 13 - Checkweigher Tag structure from OPC Kepware

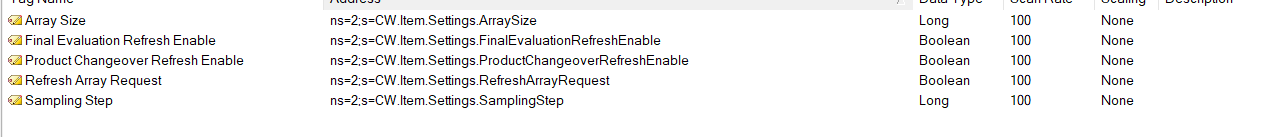


Figure 14 - Tags from Checkweigher Kepware

Once we have these machines connected, we can watch the information with the OPC Quick Client application embedded in Kepware.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Figure 15 - Tag values from the Quick Cliente application

### Thingworx connection

The connection to thingworx is something more complex as we need to stablish the “bridge” that is an entity Industrial connection.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figure 16 - Kepware Thingworx configuration

Figure 17 - Industrial Connection

We must create Remote Things to read the Tags and link them to the Industrial Connection.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 18 - Remote Thing Configuration

With these steps we have completed the linking between ThingWorx and Kepware, and we are ready to process the information and develop the application.

## Application workflow

The workflow for the application is not so complex as we have 4 different threads. The main ones are the machine ones.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 19 - Figma of the application descriptive visual document

You can study in detail the Figma structure on the following link: [Figma document](https://www.figma.com/file/U60d6aMdDucoWgVP5cP8OU/Footwear_Services?node-id=0%3A1)

### Profilometer

We Will start with the profilometer workflow as is the first one in the inspection order. We will start at the profilometer screen.

Texto

Descripción generada automáticamente

Figure 20 - Profilometer Screen

In this screen the inspection is stopped so we need to click on new Inspection to start a new one. When we click a Pop Up will appear asking for the job information.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figure 21 - New Inspection Pop Up

Once we fill the information needed if we click in the start new inspection button the application will get ready to receive new barcodes.

Captura de pantalla de un celular

Descripción generada automáticamente

Figure 22 - Application ready to receive Barcodes

Once we scan a barcode another pop up will appear and here, we can choose between fail or pass in the visual inspection. We can also write a comment and a textfield input is added so the operator can scan a “S” or “s” and automatically select the pass option.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 23 - Visual Inspection Pop Up

After this the conveyor is ready to a part to be placed and scan the measurements. When we place a part the machine scan the measurements are sent them to thingworx so we can represent them in the table and show a PASS or FAIL message depending on the inspection result.

Captura de pantalla de un celular

Descripción generada automáticamente

Figure 24 - New line created from a measurement

Other functionalities are: Erasing a selected line, cancel a part in the middle of an inspection, stop the inspection at any time, get back to the menu when the inspection is stopped and change the mode of execution.

When we click on stop inspection a Pop Up will appear asking if we want to save the information in the database (and hence into a CSV) or stop without saving.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figure 25 - Stop Inspection PopUp

If we change the mode from Automatic to Manual, then the profilometer will use the stored specifications in the model. The automatic mode reads the specifications from Thingworx.

### Checkweigher

The checkweigher application functionality is similar but without the Visual Inspection section at the beginning of the inspection.

Captura de pantalla de computadora

Descripción generada automáticamente

Figure 26 - Checkweigher screen

In this screen the inspection is stopped so we need to click on new Inspection to start a new one. When we click a Pop Up will appear asking for the job information.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figure 27 - New Inspection pop up

Once we fill the information needed if we click in the start new inspection button the application will get ready to receive new barcodes.

When a Barcode is entered the application waits for the weight.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Figure 28 - Application waiting for weight

When the weight gets into the application, we update the line with the result of the inspection.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Figure 29 - Application updated with the weight result

Other functionalities are: Erasing a selected line, cancel a part in the middle of an inspection, stop the inspection at any time and get back to the menu when the inspection is stopped.

When we click on stop inspection a Pop Up will appear asking if we want to save the information in the database (and hence into a CSV) or stop without saving.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figure 30 - Stop Inspection PopUp

### Program manager

The program manager application is a solution to created / delete / update programs and its versions so the operator can modify the specifications in any inspection.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Figure 31 - Program manager screen

In this screen we can select to add a new program and a Pop Up will appear to fill the new program information.

Icono

Descripción generada automáticamente

Figure 32 - New program Pop Up

If we click in new version, then a screen appears to fill the new version specifications.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 33 - New version screen

If we click on Model Manager we change the screen to modify the model relations from the profilometer.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 34 - Model manager screen

We can click the button “Update model” and we can select a new model for a created version.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 35 - Update model pop up

Other functionalities in the main screen is to select the current version of the program double clicking the program version and to show a program in to the program list when we start an inspection.

### Job viewer & Export

The Job viewer is the simplest application in this bundle. Its main objective is to show a visual resume of the inspections and to download a CSV file.

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Figure 36 - Programs manager screen

We can select the program, and the size and then a dropdown menu will show us all the jobs corresponding to that program and size independently of the version used. We can also write the job id. Then depending on the tab selected we will see the result of the profilometer / checkweigher / visual inspection.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 37 - Program dropdown menu

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 38 - Job dropdown menú

If we click in “Export CSV” button a CSV file will be created in the repository: HP.Footwear.Repository and the file will be also downloaded locally into the machine.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 39 - File download

## Application functionalities

The skeleton and muscles of the application are the services located in the entities of the projects. These services process the information not only to send it to the database and to the CSV files but also manage the interaction with the user in the front-end.

The system of services is linked to the mashups as the inspections are now operator-guided processes and we need not only the machine info but the operator interaction to go through the workflow.

In the following schema we will show you the services we use in the mashups and are triggered by the interaction of the user or the machine actions.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figure 40 - Footwear mashups and services

This schema can be studied in the following link: [Services schema.](https://www.figma.com/file/6fSX7WhX6PHAJqkMmUPxOr/Footwear_Services?node-id=0%3A1)

We will explain in detail the functionality of the services in the following parts but for the full code access to ThingWorx composer must be granted.

### Profilometer

We are going to start with the profilometer as is the first application in the workflow.

For achieving the profilometer work we need to detect some interactions with the machine and the operator, these interactions are the following:

#### Start inspection by the operator

This functionality starts when the operator clicks on the “Start new inspection” button. This triggers a Pop up *HP.Footwear.Profilometer.NewInspection.PU* This pop up will have 3 fields: Job Id, Program and total parts. These 3 fields will be the parameters of the service ***profilometerStartInspection*** and go back to the initial screen. The service will load all the information about the job into temporal infotables and will trigger the inspection start into the thingworx system, as well as change the buttons availabity and send the machine the model change in case the automatic mode is selected.

#### Job information input by the operator

The job ID of the current job is now written manually by the operator and it’s not unique.

The program input will be selected from a dropdown menu that it’s fed by an infotable list that gets updated with the program database changes.

Tabla

Descripción generada automáticamente con confianza media

Figure 41 - Infotable containing the programs

The total elements number is not a requisite to stop the inspection but more parts than total elements won’t be allowed once we reach the limit. The barcode input text field will be blocked.

#### Barcode scan by the operator

The first input of the part to scan is the barcode. When a barcode is scanned by the operator the text field is filled and an automatic ‘enter-key’ is pressed, so we can load the barcode into our temporal info tables that will be updated later the process, this will be performed by the service ***profilometerReadBarcode***. This triggers the pop-up *HP.Footwear.Profilometer.VisualInspection.PU.*

#### Visual inspection by the operator

Once the visual inspection Pop-Up appears we will need to choose between PASS or FAIL. An optional comment is possible. Then we click on “Validate” so we can trigger the service ***profilometerReadVisualInformation.*** This service will load the information of the Visual inspection into our temporal info tables. Then we go back to the main screen of the profilometer to wait for the measurement to get done by the machine. If the visual inspection is failed then the process for that part is finished and the program waits for another barcode to get scanned.

#### Measurement realised by the machine

Once the machine gets the measurements done it sums a value of 1 into an address called *itemsCount*. This change triggers a subscription in our system that triggers the service ***profilometerReadMeasurements*** and stores the measurements into our temporal infotables for be saved in the database later. This also gets to the front-end the information of the measurements for the operator to check if they are correct.

#### Part measurement cancel by the operator

When an operator wants to cancel the inspection process it needs to click on “Cancel part”, this will trigger the service ***profilomterCancelPart*** that will stop the inspection for the part and erase the info collected until that moment. The application will again be waiting for a new barcode.

#### Erase a measured part by the operator

If a part inspection is corrupt or the operator has made a mistake it can erase a selected part of the inspection clicking on that line and then in the button “Erase item”, this will trigger the service ***profilometerDeleteSelectedRow*** that will erase all the info about that item that is selected.

#### Stop inspection by the operator

Once the operator wants to stop the inspection because all the parts has been inspected or because it’s a wrong inspection it can click on the button “Stop inspection”. This will trigger the pop up *HP.Footwear.Profilometer.StopInspection.PU* with two options: one for stopping without saving any inspection so the profilometer application will come back to the initial state waiting for a new inspection, or two: for stopping and saving into the database. If the operator selects the first option the service will be ***profilometerStopInspectionNoSave***, if the operator selects the second option the service will be ***profilometerStopInspection***.

#### Save to database and export as CSV by the operator

This second service will trigger the CSV generator service ***profilometerExportCSV*** and the save to database services called ***profilometerAddInfoToDatabase, profilometerAddJobToDatabase*** and ***profilometerAddVisualInspectionToDatabase.***

After this the CSV will be available in the repository and the database will contain the information about the inspection.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figure 42 - Footwear repository

#### Change the profilometer mode by the operator

If we want to change the profilometer mode we must click on the button “Manual mode”, and it will trigger a pop up with a password entry so the operator cannot change manually the mode. This will be done through the service ***profilometerToggleManualMode.***

#### Back to the menu by the operator

When an inspection is stopped the operator can come back to the menu clicking on the button “Back”.

### Checkweigher

For achieving the checkweigher work we need to detect some interactions with the machine and the operator, these interactions are the following:

#### Start inspection by the operator

This functionality starts when the operator clicks on the “Start new inspection” button. This triggers a Pop up *HP.Footwear.Checkweigher.NewInspection.PU* This pop up will have 3 fields: Job Id, Program and total parts. These 3 fields will be the parameters of the service ***checkweigherStartInspection*** and go back to the initial screen. The service will load all the information about the job into temporal infotables and will trigger the inspection start into the thingworx system, as well as change the buttons availabity and send the machine the model change in case the automatic mode is selected.

#### Job information input by the operator

The job ID of the current job is now written manually by the operator and it’s not unique.

The program input will be selected from a dropdown menu that it’s fed by an infotable list that gets updated with the program database changes.

Tabla

Descripción generada automáticamente con confianza media

Figure 43 - Infotable containing the programs

The total elements number is not a requisite to stop the inspection but more parts than total elements won’t be allowed once we reach the limit. The barcode input text field will be blocked.

#### Barcode scan by the operator

The first input of the part to scan is the barcode. When a barcode is scanned by the operator the text field is filled and an automatic ‘enter-key’ is pressed, so we can load the barcode into our temporal info tables that will be updated later the process, this will be performed by the service ***checkweigherReadBarcode***. This triggers the pop-up *HP.Footwear.Checkweigher.VisualInspection.PU.*

#### Measurement realised by the machine

Once the machine gets the measurements done it sums a value of 1 into an address called *itemsCount*. This change triggers a subscription in our system that triggers the service ***checkweigherReadWeight*** and stores the measurements into our temporal infotables for be saved in the database later. This also gets to the front-end the information of the measurements for the operator to check if they are correct.

#### Part measurement cancel by the operator

When an operator wants to cancel the inspection process it needs to click on “Cancel part”, this will trigger the service ***checkweigherCancelPart*** that will stop the inspection for the part and erase the info collected until that moment. The application will again be waiting for a new barcode.

#### Erase a measured part by the operator

If a part inspection is corrupt or the operator has made a mistake it can erase a selected part of the inspection clicking on that line and then in the button “Erase item”, this will trigger the service ***checkweigherDeleteSelectedRow*** that will erase all the info about that item that is selected.

#### Stop inspection by the operator

Once the operator wants to stop the inspection because all the parts has been inspected or because it’s a wrong inspection it can click on the button “Stop inspection”. This will trigger the pop up *HP.Footwear.Checkweigher.StopInspection.PU* with two options: one for stopping without saving any inspection so the profilometer application will come back to the initial state waiting for a new inspection, or two: for stopping and saving into the database. If the operator selects the first option the service will be ***checkweigherStopInspectionNoSave***, if the operator selects the second option the service will be ***checkweigherStopInspection***.

#### Save to database and export as CSV by the operator

This second service will trigger the CSV generator service ***checkweigherExportCSV*** and the save to database services called ***checkweigherAddInfoToDatabase, checkweigherAddJobToDatabase.***

After this the CSV will be available in the repository and the database will contain the information about the inspection.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figure 44 - Footwear repository

#### Back to the menu by the operator

When an inspection is stopped the operator can come back to the menu clicking on the button “Back”.

### Program manager

This application manages the programs and versions system. In order to manage the system we need some functionalities

#### View the current programs and versions

The application starts with the mashup HP.Footwear.Viewer.MU This mashup shows the programs list and the versions of the programs if the program is double clicked. The double click triggers the service ***queryProgramVersionInfo*** to show the version info in to the right infotable.

#### Toggle program visibility

If the engineer doesn’t want a program to be selected from the operator he can toggle the visibility and the program won’t be shown in the list of New Inspection Program.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figure 45 - Toggle visibility button and visible property

In the info table we can check what programs are visible and what programs are not visible.

#### Add new program

If we want to add a new program we must click on the button “New program” that will trigger the pop up *HP.Footwear.Program.ProgramAddNewProgram.* In this pop up we can fill the new program information that will be the parameters for the service ***programAddProgramToDatabase.***

#### Add new version

If we want to add a new program we must click on the button “New version” that will trigger the mashup replace for *HP.Footwear.Program.ProgramAddNewVersion.* In this mashup we can fill the new version information that will be the parameters for the service ***programAddVersionToDatabase.***

#### Delete version

If the engineer wants to delete one version, it must click the button “Delete version” while the version is selected. This will trigger the service ***queryDeleteVersionFromVersionId*** that erase the version selected and all its information. For the version to be erased all the related jobs must be erased previously.

#### Delete program

If the engineer wants to delete one version, it must click the button “Delete program” while the version is selected. This will trigger the service ***programDeleteProgram*** that erase the program selected and all its information. For the program to be erased all the related versions must be erased previously.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figure 46 - Delete program pop up

#### Manage profilometer models

To manage the profilometer models we must click on the button “Manage profilometer models”. This will replace with the mashup *HP.Footwear.ProfilometerModelViewer* where we can manage the profilometer models and update them.

### Job viewer & Export

To view and export the programs we must have some functionalities in this application:

#### Select the program and size

We need to select first the program in the dropdown menu right to the label “Program”, this selection will load the information to show into the next dropdown menu to choose the size.

#### Select the job or write the job

After the program and size are loaded the last dropdown menu will show the jobs that are inspected under that program and size specifications independently from its version. Another way to select the job is to write the full job id in the text field input. This will load the job id immediately in the tables.

#### Select the inspection

Now the job is selected the information is loaded into the tables, but since we have three different inspections: visual, checkweigher and profilometer we created a tab system to show these inspections in different tables, so the user must select the tab that represents the inspections to be shown.

#### Export as a CSV

Depending on the selected inspection (Profilometer or checkweigher) if the user clicks on “Export CSV” button this will trigger the service to download and load the CSV file into the repository. This functionality is only available for profilometer or checkweigher, as engineer department doesn’t need the visual inspection results.

## CSV Export

For the CSV export section, we must explain that both profilometer and checkweigher CSV are different format and has some cells we must write after processing the job information.

### Profilometer

The profilometer CSV must contain the Job ID, the parts, the nominal of each measurement (Toe width, heel width, external length and internal length), the sample number (for the moment is always 0), and the deviation of the part from the nominal.

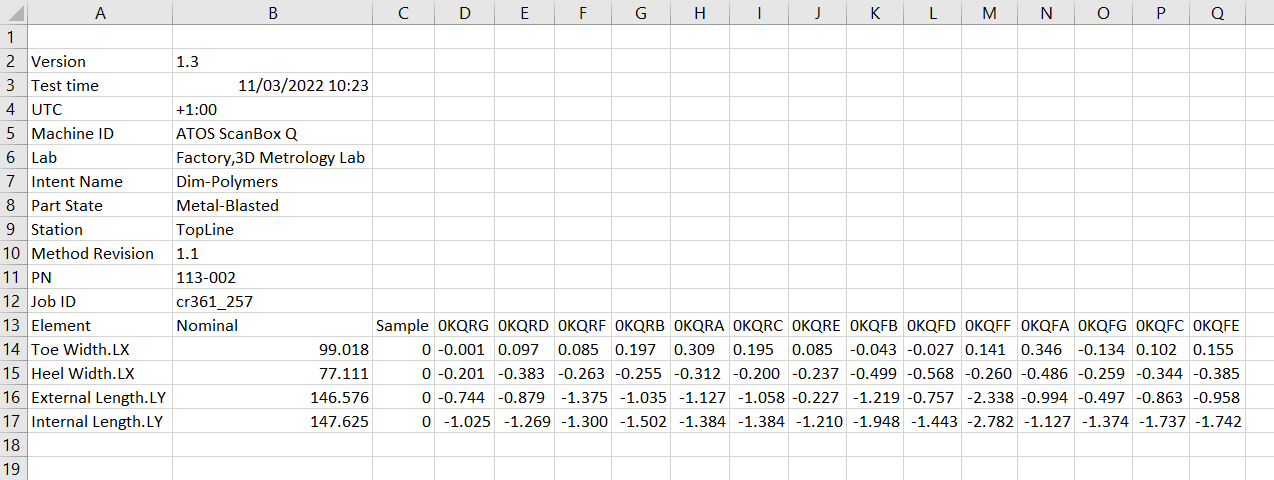


Figure 47 - Profilometer CSV Format

### Checkweigher

For the checkweigher CSV we need to also show the information from the inspection, the JobID, the parts and the weight from the part.

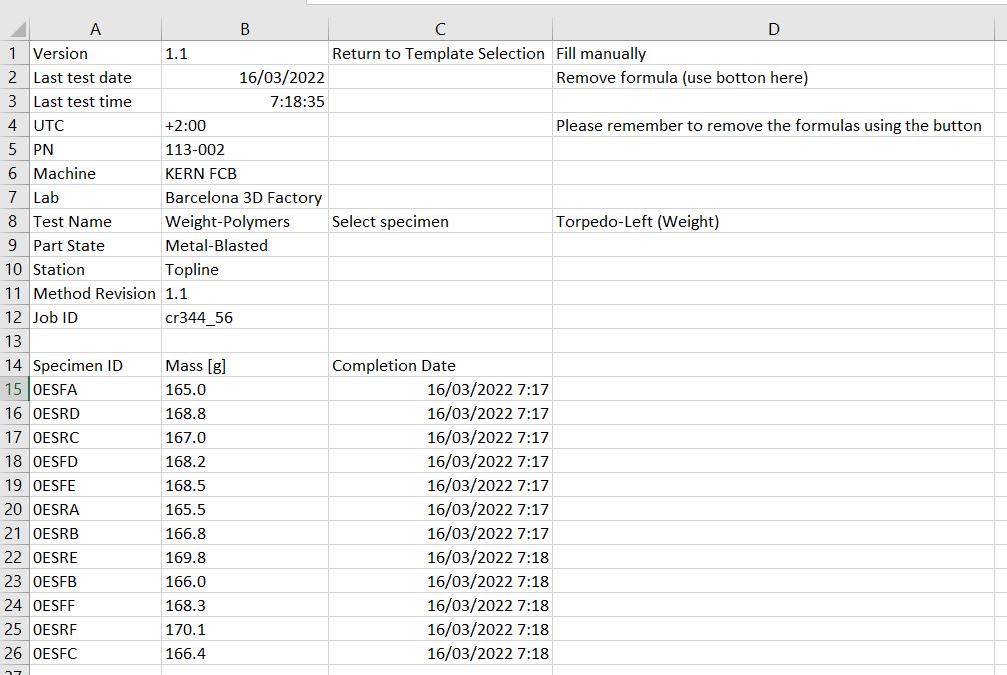


Figure 48 - Checkweigher CSV Format

## Database comunication

As we explained before to send and receive information to the database, we need to use the *PTC.SCA.SCO.PostgresDatabase* entity services called: ***input, update, delete, query***. We will show you some examples, because the services need an specific formatted json as input so it can show, modify or enter the information.

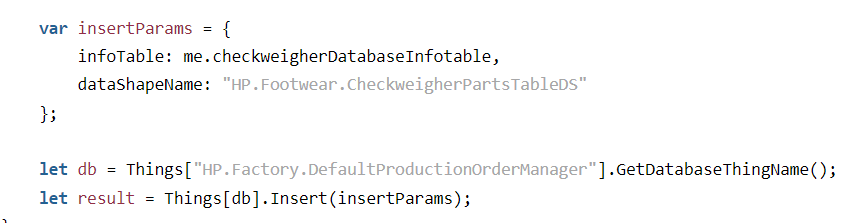


Figure 49 - Insert information example

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figure 50 - Query example

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figure 51 - Update example

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figure 52 - Delete example