Crossrail Integration Facility and Test Automation – How an off-site fully automated testing facility increases resilience of complex signalling projects

Alessandra Scholl Sternberg, System Engineer, Siemens Mobility Limited

SUMMARY

Systems Integration has gained a higher level of importance as complex railway projects operate under tighter schedules than ever before and with limited access to tracks to run tests. This paper demonstrates how a fully automated off-site testing facility is extremely valuable to increase efficiency, cost-effectiveness and resilience of systems throughout a railway project life-cycle. The Crossrail Integration Facility is a great example of this practice.

The intricacies of the Crossrail Integration Facility are presented, including the integration of products from different suppliers, and integration of equipment necessary for transitions between CBTC (Communications Based Train Control), ETCS (European Train Control System) and TPWS (Train Protection & Warning System – the UK national train protection system). It will also be demonstrated how an automation structure with an extensive function library has been created to support the complete testing of the signalling system.

Systems Integration Facilities, such as Crossrail's, provide a means to perform thorough off-site interface, integration, timetable and transition testing, as well as simulation of faults to understand the system's behaviour under degraded and emergency situations. It is a cost and time-effective approach to de-risk the later stages of the project, which brings real benefits to the delivery of railway signalling systems.

1 INTRODUCTION

The railway is a very complex system, involving – in the UK – numerous stakeholders, such as: end customer, government, service operators, rolling stock owner, rolling stock supplier, infrastructure owner, infrastructure supplier, infrastructure maintainer – each with individual (and often conflicting) legal and financial objectives. Within each category there are potentially several different organisations that consider each other market competitors. There is also a complexity in technology, with various intricate signalling systems worldwide. A single line might operate under distinct systems in different areas, requiring complex transitions for safe operation of trains. Contemplating these facts makes it clear how Systems Integration is key for a railway line to work reliably, and how complex the System Integration process can be, as it depends on stakeholders with distinct objectives working together. Successful systems integration also relies heavily on system-level tests; however, urban areas are getting bigger and denser, and service hours are getting longer – some railways run 24h a day. The real railway is not as available for system testing as would be desirable, which leaves a lack of infrastructure supporting system integration and cooperation between suppliers.

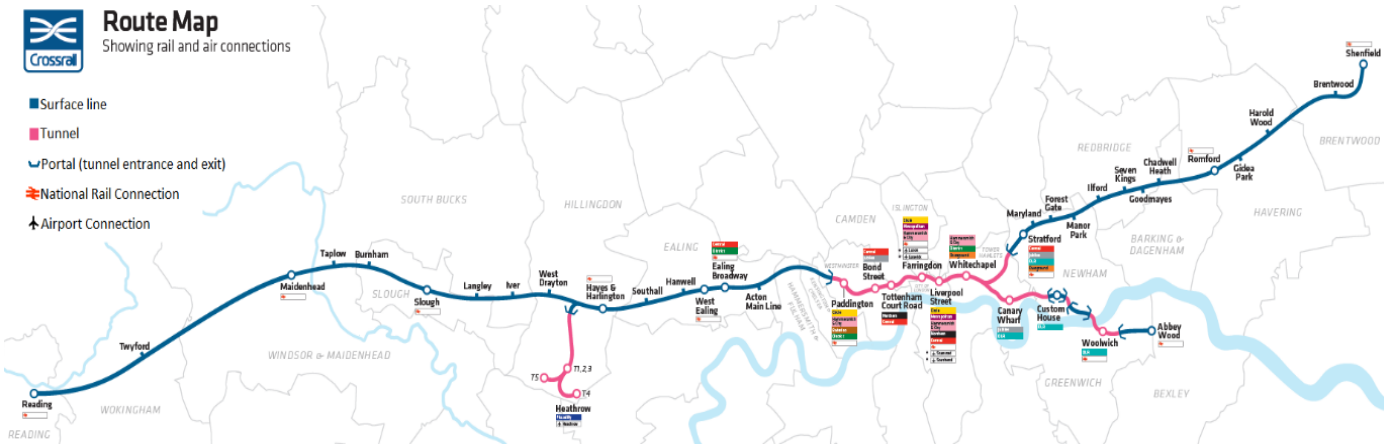

Figure 1: Route map of Elizabeth Line. Source: http://www.crossrail.co.uk/route/maps/route-map

The signalling system's main purpose is to optimise train movements whilst keeping them safe at all times and under all circumstances. Therefore, the system must be proved to be resilient, meaning that it must behave safely and reliably even under unintended operation or under the influence of external faults. It is very challenging to test the system's response to unintended scenarios, even if longer access to it were granted, because to be able to put the system under certain complex situations, a lot of negotiations between stakeholders and risk assessments would have to be undertaken. The difficulties mentioned can be lessened with the use of system integration facilities. A great example of this practice is the Crossrail Integration Facility.

Crossrail (the future Elizabeth Line in London – figure 1) is a major railway project that goes across one of the largest urban areas in the world. It is currently one of Europe's largest infrastructure projects, and it is estimated that 200 million passengers will use it each year. It operates under three distinct train control systems - ETCS, CBTC, TPWS (UK train protection system) -, with high capacity and throughput. To be able to deliver a robust operational railway in a tight schedule and with limited access to test tracks, the Crossrail Integration Facility has been implemented to provide off-site interface testing and integration of critical systems.

2 THE CROSSRAIL INTEGRATION FACILITY

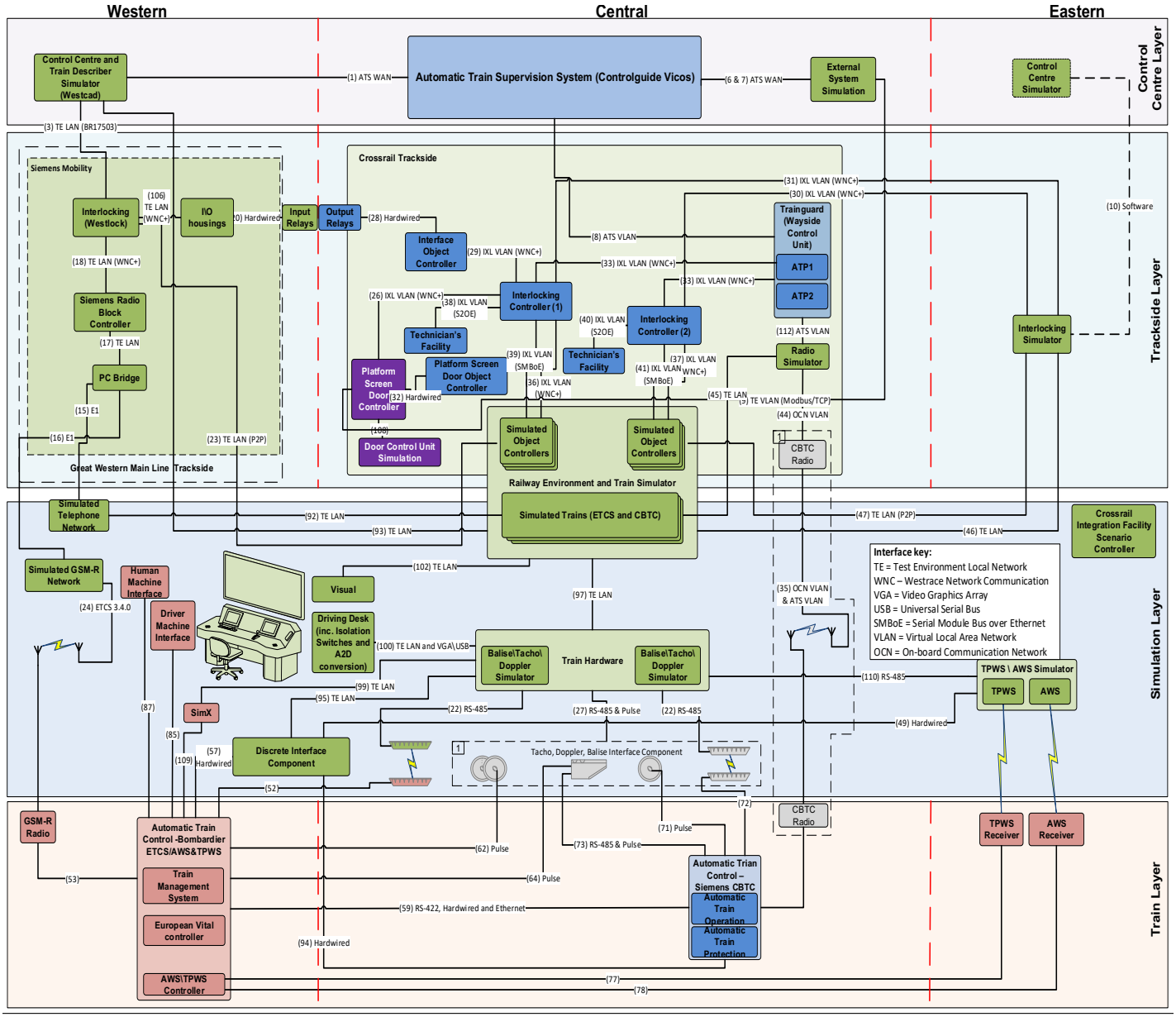

Figure 2: System Architecture Diagram

As already highlighted, the need of a System Integration Facility for the Crossrail project was identified due to the tight project schedule and the ongoing complex civil works, combined with an intricate signalling system, encompassing three distinct train control and protection systems, and the pressure for minimum disruption to the existing live London Underground system. Thus, the Crossrail Integration Facility was designed and implemented in phases, planned accordingly to the development of the products used within the system.

The objective of building an Integration Facility is to test and prove the functioning of system interfaces prior to their installation and deployment. It is not intended to replace any steps of the test & commissioning phase of the project, but to support it by identifying and mitigating defects early.

The Crossrail Integration Facility is a testing facility with 112 interfaces between a mix of real products – the same as the ones used in the railway - and simulators. The products therefore can be categorised under "constituent domain items", which are the subsystems under test, and "test execution domain items", which allow for the integration and operation of the whole system under the integration facility environment. As far as the subsystems under integration tests are concerned, they are part of a system controlling a real railway.

The system architecture (Figure 2) diagram illustrates the integration of the products. It is colour coded as follows:

- Blue – Constituent domain item supplied by Siemens Mobility;
- Green – Test Execution Domain items supplied by Siemens Mobility;
- Salmon – Constituent domain or Test Execution Domain item supplied by Bombardier;
- Purple – Constituent domain or Test Execution Domain item supplied by Knorr-Bremse.

2.1 Key Constituent Domain Items

The list that follows cover the key sub-systems within the Crossrail Integration Facility that are identical to the ones used in the real railway. They comprise the sub-systems that are under test.

The **Automatic Train Supervision (ATS)** - Siemens Controlguide Vicos - is the Central Operating Section Control System, responsible for monitoring and controlling train movements. It is equipped with Automatic Route Setting (ARS) and Automatic Train Regulation (ATR), and is capable of adjusting individual train times to optimise traffic.

The **Interlocking** - Siemens Trackguard Westrace Mk2 - provides point, route, Platform Screen Door (PSD) interlocking functions and secondary train detection functions from axle counters. (- the reader is reminded that the primary train detection function in the Central Operating Section is the train position reporting).

The **Platform Screen Door (PSD) Control Unit** - Knorr Bremse Platform Door Controller (PDU) – which connects to 27 individual simulated Door Control Units (DCUs) of the platform screen doors as in Bond Street station.

The **Trackside Automatic Train Control (ATC) System** - Siemens Trainguard Mass Transit (TGMT) Communication Based Train Control (CBTC) Wayside Control Unit (WCU) - is the main trackside element of the ATC system used in the Central Operating Section of Elizabeth Line.

The **Train-borne Automatic Train Control (ATC)** is composed of:

- European Train Control System (ETCS), Bombardier Transportation (BT) European Vital Controller (EVC);
- Communication Based Train Control (CBTC) Siemens Trainguard Mass Transit (TGMT) On-board Control Unit (OBCU), which is equipped with Automatic Train Operation (ATO) and Automatic Train Regulation (ATR) within the CBTC area;
- Automatic Warning System (AWS) and Train Protection & Warning System (TPWS) Train Module – Mors Smitt UK Ltd AWS & TPWS Specific Transmission Module (STM).

The **Automatic Warning System (AWS) and Train Protection & Warning System (TPWS) Trackside Module** – Mors Smitt UK Ltd AWS & TPWS Track Simulation Module.

2.2 Key Test Domain Items

It is not possible to test the Constituent Domain Items without a real railway, unless Test Domain Items that simulate a railway system with trains running is provided. For this reason, the Key Test Domain Items listed here are included in the Crossrail Integration Facility. With the Test Domain Items working correctly, the Constituent Domain Items can be tested, as then they believe that they are operating a real railway.

The **Railway Environment and Trains Simulation (RETS)**, at the very centre of Figure 2, is a PC-based software application that provides the trackside equipment simulation for signals, points and axle counters; and provides the interlocking I/O for those objects. It also provides fully simulated trains, which are capable of communicating with real CBTC and ETCS wayside equipment, and one simulated train that hosts the real Train-borne Automatic Train Control equipment, including representative train interfaces.

The **Interlocking of the Great Western Main Line Area** is a Siemens Trackguard Westlock, being used to emulate the Alstom Smartlock for the purposes of the Crossrail Integration Facility only.

The Interlocking for the Great Eastern Main Line area is simulated – it is represented by the **Interlocking Simulator (CIPHOST)** box in the architecture diagram.

The **Driving Desk** (Figure 3) is designed to represent the Class-345 Bombardier Train cab, and interfaces with the real train control systems. The screen in the Driving desk interfaces with a **Visual** Display software, which resembles the driver's view when driving on Elizabeth Line.

The **Train Hardware** software simulates the train wiring and train relay logic using the inputs from the Driving Desk and on-board ATC equipment, and provides the necessary outputs in response. It also performs the train dynamics calculations for determining the speed and acceleration of the train, the output of which is used to provide simulated Doppler Radar and Tachometer inputs to the on-board ATC equipment. It simulates Balises and energises the AWS magnet and TPWS antenna in the Mors Smitt AWS & TPWS trackside module.

Figure 3: Class 345 Train Driving Desk with Visual software

3 TESTING

To be able to prove that the system performance aligns to the expected behaviour and that the system integration process is successful, system-level tests need to be undertaken. The integration facility provides means for thorough off-testing of diverse natures, such as:

- Interface testing: test the interfaces between all systems and applications;
- Integration testing: test that products work together to provide the desired emergent properties of the system;
- Timetable testing: Introduction of as many trains as a real railway timetable will run, plus testing the movement of trains in and out of the sidings;
- Transition testing: Run trains between Great Western Main Line (GWML), Central Operating Section (COS) and Great Eastern Main Line (GEML) to test the transitions between CBTC, TPWS and ETCS;
- Stress testing: test the performance of the system and interfaces under overloaded conditions;
- Faults testing: Introduction of faults to understand the system's behaviour under degraded and emergency situations.

Stress testing and fault testing are related to system resilience - these two types of test put the system into unexpected situations due to external factors. An example of a stress test would be to create a timetable with more throughput of trains than the system was initially designed for. A real example of this test will be detailed in section 4.4. One can easily understand the advantages of having a controlled factory environment to perform such a test, as the conditions required can be readily orchestrated in an inexpensive way without the need to negotiate access to the client and other shareholders of the system. It would be very difficult to test the system under this scenario in the real railway system - there would be several safety implications and necessary agreements. This highlights the benefit of the Crossrail Integration Facility to test and consequently improve the resilience of the system.

4 TEST AUTOMATION

To increase the utilisation of the facility without the need for continuous human interaction, and to facilitate the execution of repetitive tests - while also making the system integration tests more consistent and reproducible - the Crossrail Integration Facility is equipped with an extensive system automation library. The test automation is designed to interact with the key software components to enable complete testing of the system, with most tests being completely autonomous.

A possible consequence of running automated tests is that the behaviour of the system might be different than when being operated by a human or by a script command, depending on how the automation is designed. Some functionality might not be available for the end user, but it is available for the system integrator writing the test automation. As an illustration, the signaller's workstation is designed so that if a user wants to set a route for a coming train, they should click the entry signal, followed by a click at the exit signal of the route, and then click request route button. If the same operation is requested by the automation through a "backdoor" command, depending on the design of the flow of commands within the signaller's workstation, even though the route appears to be set the same way, the signaller's workstation was not designed to be used in that manner, and the operation might have skipped a crucial check step in its execution. Therefore, it is important to make sure that the automation does not affect the result of the test. For this reason, in the Crossrail Integration Facility, the freeware software AutoIT was chosen for most of the automation functionality. It provides the ability to manipulate mouse moves and key stroke inputs in Windows environments, so that the system sees no difference in the input provided by the automated tests and the input provided by a human user operating the system.

The test automation permits 24/7 utilisation of the system, enabling robustness tests to be executed without human interaction for long periods of time. Therefore, full utilisation of the facility's time can be achieved without the need for staff working on a shift basis. Logging is also provided by the automation for debugging purposes, in combination with product specific logging.

An extensive system automation library has been written, which enables complex set-ups to be achieved, health checks to be accurately performed, endurance testing to occur over extended periods and the implementation of tests of repetitive nature.

4.1 Test Automation Structure

The test automation is composed of a Scenario Controller application and Test Clients. The Scenario Controller runs in a dedicated automation computer in the Crossrail Integration Facility. The Test Clients run in the computers that hold software applications of the Crossrail Integration Facility that require automation. The connection between the Scenario Controller connects to the Test Clients via TCP interface. The Test Clients cannot communicate with each other through the automation network.

Figure 4 illustrates the automation network (for simplicity purposes, not all client nodes are represented in this diagram). As one can see, the Scenario Controller can send and receive messages from all applications holding Test Clients; however, the Clients are not able to communicate between themselves through the automation. The Clients only send messages to the controller of the nature of health checks or function execution results.

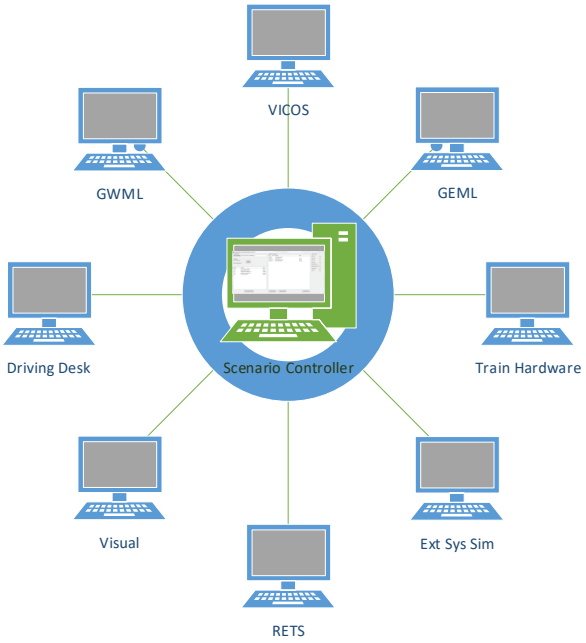

Figure 4: Illustration of the automation network structure.

4.1.1 Scenario Controller

The Scenario Controller application was developed in house. It is composed of a Graphical User Interface (GUI) as shown in figure 5. On the left, the GUI displays a list of test scripts. Tests can be run repeatedly, or different tests run sequentially. The result of the execution of the test is populated as the tests finishes. The middle of the GUI holds the list of sent/received messages. On the right, the GUI displays a list of Test Clients to which the Scenario Controller PC is connected. Through this connection, the Scenario Controller is able to request the execution of the specific automation functions in each product of the system.

The request to execute an automation function can be sent from the Scenario Controller to a Test Client either using the debug functionality or test scripts. The debug functionality sends a request to execute a single automation function to a single Test Client. The test scripts are simple text files with a sequential list of automated functions, which are assigned to different clients.

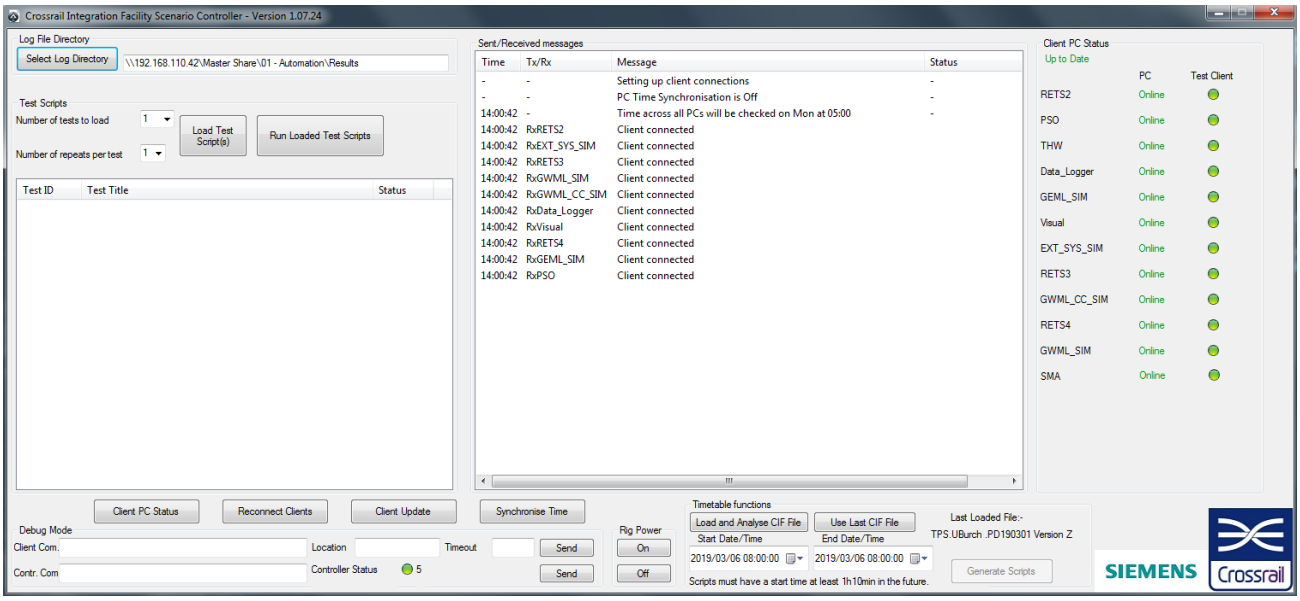

Figure 5: Scenario Controller application

4.1.2 Test Clients

The Test Clients hold the specific automation functions for each software application that is automated. They stay on standby until receiving a message from the Scenario Controller. Then, they execute the requested function and return a value depending on the outcome of the execution – in most cases it is either a "pass" or a "fail".

All functions are developed by the supplier of the Crossrail Integration Facility and are bespoke to each product within the system. Some examples of automated test functions are listed and explained below:

Example of VICOS functions

- VicosTakeControl() – This function takes control of the control areas within the COS. Taking control is necessary before executing any commands like setting routes, setting Temporary Speed Restrictions (TSRs), station skips, etc. within the Signaller's Workstation.
- SetCOSRoute() – This function sets a specific route, provided that it sits in an area that the Signaller's Workstation currently has control over.
- CancelCOSRoute() –This function cancels a specific route, provided that it sits in an area that the Signaller's Workstation currently has control over.
- SetTSR() – This function sets a temporary speed restriction of the specified speed in a specified section of a track.
- CancelTSR() – This function cancels a temporary speed restriction.
- SetHeadCode() – This function sets a specified train description (referred to as headcode in the UK) for a train in a specified location. It is used extensively in timetable tests.

Example of RETS functions

- StartupRETS() – This function starts the RETS application in the desired mode. In Crossrail, RETS can be used either in Traffic Sim mode – only simulated trains – or in SI Lab mode – with an emulated train that interfaces with the real Train-borne ATC system.
- LoadTestScript() – This function loads a test script into RETS.
- StartRETSScript() – This function starts the execution of the loaded test script.
- OpenTrainDMI() – This function opens the Driver Machine Interface (DMI) of a specified simulated train.

All functions include checks to ensure that its execution was successful. If a failed, or an unknown or unrecoverable scenario is found, then the Test Client returns a “fail” message in response to the Scenario Manager message that requested the execution of that function. A “fail” message then interrupts the test and makes the Scenario Manager proceed to a clean-up process, which includes saving logs and taking screenshots of all Test Clients, so that the facility user understands the state of every product at the time of the failure.

4.1.3 Automation Logging and Debugging

The Crossrail Integration Facility is equipped with comprehensive logging. Most software applications have specific built-in logging functionality. The automation provides extra logging.

Each Test Client logs its activities locally into an activity file. This file registers both commands received individually - sent through the debug mode functionality in the Scenario Controller - and the specific commands received through a test script. All messages received (automation functions) and sent (outcome of the executed functions) are logged with a time stamp.

There is also central automation logging, which is only used when executing a test script. Its location is determined by the user. In the case of a test failure, debug screenshots would be taken of all Clients and added to the Log folder for the specific test. The latest messages exchanged with each client, including the Test Client debug failure message, and a copy of the step result displayed in the controller would also be stored in the same location.

4.2 Resilience of the Test Environment and the Test Automation System

It is important to consider the resilience of the Test Environment itself when performing tests in the Integration Facility. The purpose of the Crossrail Integration Facility is not to test the Test Domain items’ behaviour. Those components are there merely to enable the testing of the system composed of the Constituent Items. Hence, a lack of resilience within a Test Domain item does not reflect the resilience of the system under test, even though it affects the result of the tests that are run within the Crossrail Integration Facility.

Furthermore, the Test Automation System’s resilience can affect the result of automated tests. Since it is a pre-programmed system, it can only deal with known scenarios. If the system finds itself in a state that was not anticipated, the automation will not have been programmed to cope with that, and will, consequently, fail the test, even though the system itself might have exhibited correct behaviour for that specific scenario. It is up to the user to determine if a “fail” result in the automation test is an automation failure or a system failure.

In addition, not all automation functions are resilient to user interaction. For example, if a user changes the view in the Signaller’s Workstation, the automation will simply change back the necessary screens if requested to set a train’s headcode – in this case, the automation is resilient. However, if the user simply closes a software application mid-test, the automation will fail the test. Therefore, some measures have been introduced into some Test Clients to avoid a user induced failure, such as blocking all user inputs in some computers while a test is being executed. That way, only the Test Automation System can interact with the software application during the execution of the automated tests.

4.3 Observed benefits of the Test Automation

Besides increasing the utilisation of the facility and facilitating the execution of repetitive tests, the test automation has brought other benefits. Firstly, it is very useful to have a tool that provides top-level logging of a test. Usually each product will provide its specific logging, and when a fault occurs, the system integrator must analyse and link logging from the different applications to understand the sequence of events and find the root source of the fault. This becomes more challenging if the time and dates between applications are not synchronised. Having top-level logging provides the system integrator with a reliable record of the sequence of events. Moreover, since the automation’s logging is also performed locally, it gives the user a reference, in case of faulty time synchronisation between the applications. This information can be then combined with the specific software logging performed locally, to more easily and readily identify faults of the system.

As the Crossrail Integration Facility has been built in parallel to the development of the key Constituent Items, the automation has also proved to be very helpful in allowing advanced system functionality testing whilst the product software itself was still undergoing product testing. The reason for this is that the automation can provide

workarounds for known issues quicker than waiting for the next release of the products – therefore, further development can continue to be made with the system, decreasing project execution time. An example of this scenario was found during timetable tests. When a train is entering the Central Operating Section (COS) from either Great Western Main Line (GWML) or Great Eastern Main Line (GEML), the VICOS-ARS is responsible for setting a slot request into the COS area, as soon as the headcode (UK's train description) of the incoming train is recognised as having a route through the COS. However, it was known that, at that stage of the VICOS development, this functionality was not working correctly. To work around this issue, automation specific functions were written to set both GWML and GEML slot requests when it identified that a train was meant to be routed into the COS. These functions were incorporated into the automated test scripts, so that the timetable tests could already run reliably.

4.4 Example of an Automated Test

As an example of an automated test, the details of a timetable test will be laid out here. The requirements for the peak frequency of trains in the real railway is represented in the following diagram for the area covered by the Crossrail Integration Facility.

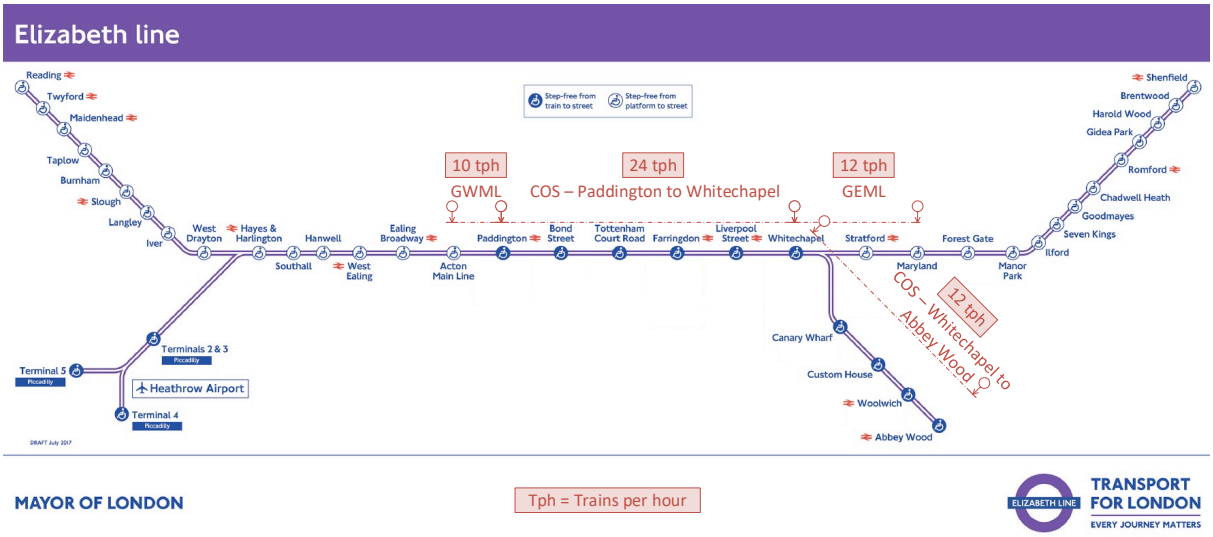

Figure 6: Peak frequency of train service pattern in Elizabeth Line.
Modified diagram from http://www.crossrail.co.uk/route/maps/route-map

The objective of the timetable resilience test in the Crossrail Integration Facility is to stress the system by timetabling more trains than the real system would run in reality. To be able to test this scenario, the system integration team has developed its own timetable, with frequencies – trains per hour (tph) – that exceed the peak frequencies showed in the diagram. The frequency of trains in each area in the resilience timetable test is shown in the following table.

RAILWAY REGION (AS DEFINED IN FIGURE 6)	FREQUENCY (TPH)
GWML	10
COS – Paddington to Whitechapel	30
COS – Whitechapel to Abbey Wood	15
GEML	15

A Common Interface File (CIF file) – the industry standard file format for Network Rail's (the owner and infrastructure manager of most of the railway network in Great Britain) timetables - has been generated for this test by the system integration team, so that it can be loaded into the system.

Distinctively from other automation tests, which require the user to write their own automation scripts, timetable tests have a specific functionality in the Scenario Controller. The timetable automation test script can be generated

automatically, once a CIF file is loaded into the automation system. The user only needs to specify a date and a time to run the test.

Even though all trains are simulated, the system only accepts trains being injected into the scenario from a limited number of locations. For this reason, once a date and time is specified, the automation system analyses which trains will be part of the test by working back when they would have to be injected in an allowed inject location.

After the timetable script is generated, the Scenario Controller will synchronise the time between the different applications in the Crossrail Integration Facility and the test is ready to be started. An example of a very short automation test script for a timetable test is shown in figure 7.

```
;Automated generation of Automation script based on Timetable
SendMessage('FileCopy("\\192.168.110.42\Master Share\01 -
Automation\Test Scripts\Timetable\CIF File Interpreter\TPS.UBurch
.PD190314 Version A\*", "C:\RETS\Scripts\Timetable\TPS.UBurch
.PD190314 Version A\", 9)', "RETS2", 60)
SendMessage('StartupRETS("TrafficSim")', "RETS2", 120)
PowerControl("TGMT OBCU", "off")
SendMessage("StopCIPHost()", "GEML_SIM", 300)
SendMessage("StartupCIPHost()", "GEML_SIM", 300)
PowerControl("Interlocking Rack", "Reset")
SendMessage('StartupRETS("TrafficSim")', "RETS2", 120)
SendMessage("WorkstationLogin()", "PSO", 30)
SendMessage("AwaitTracksideOperational()", "PSO", 180)
SendMessage("StartTDTool()", "GWML_CC_SIM", 300)
SendMessage('LoadRETSScript("C:\RETS\Scripts\Timetable\TPS.UBurch
.PD190314 Version A\Tuesday 1000 to Tuesday 1030.rss")', "RETS2",
30)
SendMessage("VicosTakeControl()", "PSO", 180)
SendMessage("SetARSOn()", "PSO", 180)
SendMessage("ClearHeadCodes()", "PSO", 540)
SendMessage('SelectWorkstationView("1")', "PSO", 180)
SendMessage("TDARSOn()", "GWML_CC_SIM", 300)
WaitUntilTime("2019/03/19 09:34:07")
SendMessage('BlockUserInput("True")', "PSO", 120)
SendMessage("StartRETSScript()", "RETS2", 100)
SendMessage("SelectRETSView(2)", "RETS2", 100)
SendMessage("OpenTrainListWindow()", "RETS2", 100)
;******* Now starting specific commands for journeys ******
WaitUntilTime("2019/03/19 09:35:27")
SendMessage('SendHeadCodeToTDTool("9C69")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:37:17")
SendMessage('SetHeadCode("9C71#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:45:17")
SendMessage('SetHeadCode("9C79#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:47:27")
SendMessage('SendHeadCodeToTDTool("9C83")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:49:17")
SendMessage('SetHeadCode("9C85#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:51:17")
SendMessage('SetHeadCode("9C87#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:53:27")
SendMessage('SendHeadCodeToTDTool("9C89")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:55:17")
SendMessage('SetHeadCode("9C91#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:57:17")
SendMessage('SetHeadCode("9C93#09", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 09:59:27")
SendMessage('SendHeadCodeToTDTool("9C95")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:01:17")
SendMessage('SetHeadCode("9C97#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:03:17")
SendMessage('SetHeadCode("9C99#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:05:27")
SendMessage('SendHeadCodeToTDTool("9D02")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:07:17")
SendMessage('SetHeadCode("9D04#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:09:17")
SendMessage('SetHeadCode("9D06#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:11:27")
SendMessage('SendHeadCodeToTDTool("9D10")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:13:17")
SendMessage('SetHeadCode("9D12#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:15:17")
SendMessage('SetHeadCode("9D14#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:17:27")
SendMessage('SendHeadCodeToTDTool("9D16")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:19:17")
SendMessage('SetHeadCode("9D18#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:21:17")
SendMessage('SetHeadCode("9D20#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:23:27")
SendMessage('SendHeadCodeToTDTool("9D22")', "GWML_CC_SIM", 60)
SendMessage("SetGWMLSlotRequest()", "PSO", 180)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:25:17")
SendMessage('SetHeadCode("9D24#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
WaitUntilTime("2019/03/19 10:27:17")
SendMessage('SetHeadCode("9D26#10", "PDXPLA")', "PSO", 100)
SendMessage("CheckRETSScriptRunning()", "RETS2", 180)
;******* Now Finished specific commands for journeys ******
SendMessage("StopTDTool()", "GWML_CC_SIM", 300)
```

Figure 7- Example of a Timetable Test

The example of the automation test script in Figure 7 demonstrates how, in a timetable test, the automation does not need to set any routes for the trains, since the Signaller's Workstation is equipped with Automatic Route Setting (ARS). Therefore, the main focus of the automation is to set up all applications correctly and make sure that all injected trains have the correct headcode assigned to them. At the start of the script (box #1), one can see the necessary commands to set up the system, such as starting the RETS application; loading the appropriate test script in RETS for this test; restarting the interlockings, so that all routes in the layout are cleared out; taking control of all control areas in the Signaller's Workstation; clearing all headcodes left behind in the layout, and switching ARS on. After that, the system is ready to start the test. The commands after the message "Now starting specific commands for journeys" are executed while trains are running in the scenario. This part of the script mainly consists of assigning the correct headcode to each train, after its injection either in GWML area – using "SendHeadCodeToTDTool" function, which assigns a headcode to a train injected in Acton Main Line platform 3 – or in the COS area – using "SetHeadCode" function with "PDXPLA" parameter, as trains that are injected in the COS get their headcode in Paddington Platform A. The previously mentioned slot request workaround is visible here (highlighted in its first appearance in box #2) – the automation sets the slot request for all trains driving from GWML into the COS appropriately.

The Crossrail Integration Facility user can observe the movement of all trains through the Signaller's Workstation, in the same way as the operator of the real railway would. Figure 8 illustrates the Signaller's Workstation during a timetable test. The details of the layout are not visible; however, one can observe the numerous occupied track (red) sections, which correspond to the location of trains within the layout. With meticulous observation, one can see that each red section is accompanied by a small box with green outline – this box contains the headcode of the train within that occupied track section. The white tracks represent sections of routes that are set and locked for the moving trains, which were set automatically through the Automatic Route Setting functionality. Dark grey sections represent unoccupied tracks.

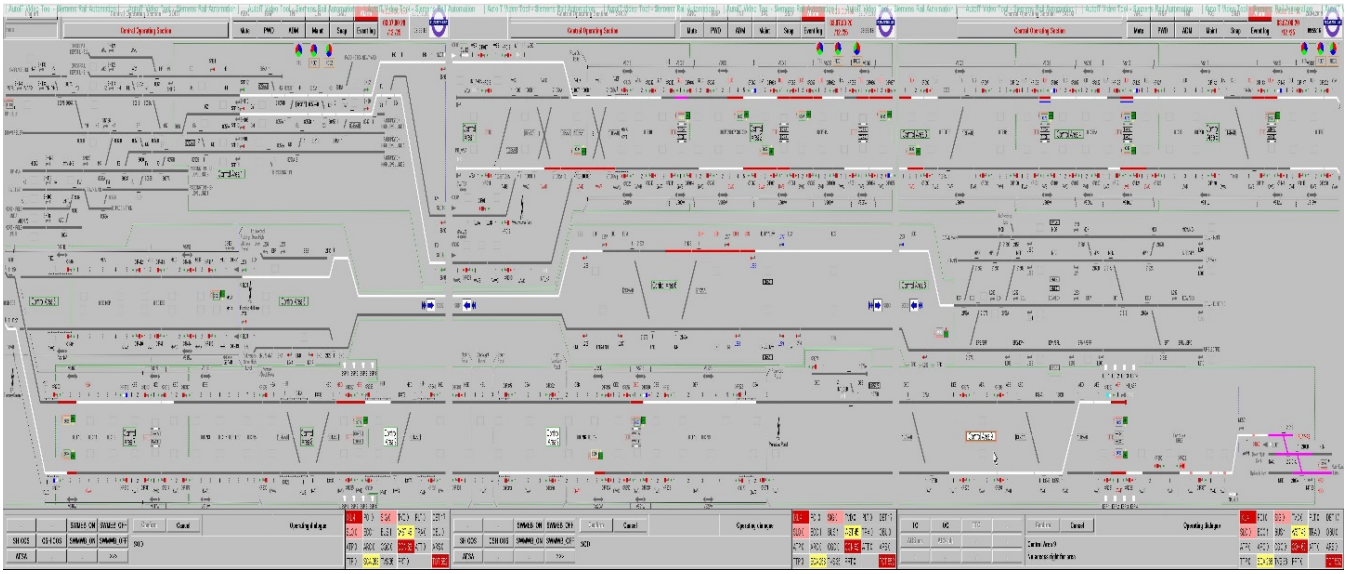

Figure 8- Signaller's Workstation display during an Automated Timetable Test created by the systems integration team

5 OBSERVED IMPACT OF HAVING AN INTEGRATION FACILITY IN THE PROJECT

Since it replicates the core parts of the real Crossrail signalling and control system, the Crossrail Integration Facility is itself a very complex system. A team with a deep understanding of the system was needed to build this facility. The investment in building the facility is, however, essential in a project like Crossrail, where if the key systems are not effectively integrated, there will be major cost, delay, functionality and even future safety implications. The cost and time necessary to run all tests that the Crossrail Integration Facility enables would be enormous in the real system.

The Crossrail project has been under pressure from the beginning, with extremely difficult civil engineering works to be completed under tight schedules, which creates the possibility for delays to occur. When this happens, it results in even less time available at the back end of the schedule, for the signalling testing. It is particularly challenging if test trains are required on site for the tests. A facility like the Crossrail Integration Facility cannot mitigate delays of a civil engineering nature, but it can be used to assist minimising the perpetuation and increase of project delays by giving the project other means of performing system integration testing and fault finding.

It is possible to identify several benefits from having a System Integration Facility in the Crossrail project:

- Fault-finding and debugging in a controlled off-site environment are a lot easier than it would be with the actual live system.
- The detection of defects early minimises the cost and time necessary for their rectification.
- Working in a controlled environment allows the use of some workarounds for specific product faults until a new product release is received. This allows system tests to continue even with known product faults, accelerating further fault finding and correction within the system and the products. Many of these

workarounds would probably not be possible in the real system, as thorough risk assessments would have to be performed before their implementation.

- It is an efficient way of de-risking the project, as an off-site facility provides the capability of executing tests that otherwise would be very impractical to perform in the live railway, such as stress tests, tests of how the system operates under degraded or emergency mode.
- Once in the operation phase of the project, the facility will provide the means to test planned updates off-site before being implemented on the real railway, so that general reliability is maintained during updates.
- It provides a means for the maintenance and operation teams to familiarise themselves early with the system, so that they can provide inputs to the project early on.
- It can be equipped with extensive test automation, obtaining all the benefits listed in section 4.2.
- Ultimately, in the long run, one of the best benefits that integration facilities can provide to the railway is the cooperation between the diverse stakeholders involved in running a safe and resilient railway. It is only through cooperation that we can work with complex systems in an increasingly globalised and intertwined world.

6 CONCLUSION

This paper has looked at how complex railway signalling projects lack the time and infrastructure access to perform extended testing and integration procedures because of tight project schedules. A solution to this problem is to provide a system integration facility. The Crossrail project is no exception, hence the decision to build the Crossrail Integration Facility.

This paper looked at the details of some of the products and software applications within the Crossrail Integration Facility. The system architecture highlights the number of complex interfaces that must be integrated for the system to work correctly. Finally, the Test Automation System was explained in detail, including its structure and automation functions. An example of a timetable test illustrated the use of the Test Automation System.

The paper demonstrated how establishing a fully automated off-site testing facility is extremely valuable to the Crossrail project. The Crossrail Integration Facility is used to run interface, integration, timetable and transition testing, as well as to test the system's behaviour under the introduction of faults or stress. The benefits of being able to run autonomous test scenarios include increased utilisation of the system and ease of execution of repetitive tests, in combination with ease of implementation of workarounds and fault finding with the extra logging provided. In addition, the scenarios related to testing the resilience of the system are much less complicated to set up in a controlled environment. Generally, it is evident that the Crossrail Integration Facility has brought several benefits to the Crossrail project itself and has given a lot more in return than the financial and time investment necessary to build it.

I believe that all future major railway signalling and control projects should use a System Integration Facility to test their products and the emergent properties of their system prior to on-site testing. This additional step within their system engineering process will also be very useful when updating a system – all updates can be tested in a controlled environment prior to their implementation. Updates can go through thorough resilience tests that would probably not be available otherwise. Clearly though, to make the most out of the investment in a System Integration Facility, it is important to make sure that the facility is completely integrated within the system engineering process, which means that the time for tests within the facility and time for fault rectification need to be incorporated into project schedules.