

Daidalos Framework for Successful Testbed Integration

Miguel PONCE DE LEON¹, Frances CLEARY GRANT¹, Marta GARCÍA MORENO², Antonio Romero

VICENTE² Mark RODDY, Czeslaw JEDRZEJEK.

¹ *Waterford Institute of Technology, Cork Road, Waterford, Ireland*

Tel: +353 51 302952, Email: miguelpdl@tssg.org, fcleary@tssg.org

² *Telefónica I+D, Address, City, Postcode., Spain*

² *Telefónica I+D, Parque Tecnológico de Boecillo, Parc. 119-120, 47151, Boecillo, Spain*

Tel: +34 983 367500, Email: martagm@tid.es, arv266@tid.es

³ *Lake Communications, BIC, Ballinode, Sligo, Ireland*

Tel: +353 71 9156830, Email: mark.rodny@lakecommunications.com,

⁴ *ITTI, ul. Palacza 91A, Poznan, 60-273., Poland*

Tel: +48 61 861 00 73, Email: czeslaw.jedrzejek@itti.com.pl

Abstract — Daidalos Testbed integration framework required detailed planning and implementation, constantly adapting to the demanding changes of a research project as it advances from development phase to integration phase. This paper describes the various integration and validation efforts required to deploy an operational daidalos Testbed infrastructure, demonstrating the effort required to achieve a successful overall integration process. With such a large scale project as Daidalos with a consortium of 49 partners, the Testbed deployment, operation and management was indeed an immense task having to create and enforce Testbed processes suitable for the efficient and effective operation of the Daidalos system during integration and validation.

Key Words — Test bed, Integration Management, Taskforces, validation.

I. DAIDALOS INTRODUCTION

Daidalos [1] is a world in which mobile users can enjoy a diverse range of personalized services – seamlessly supported by the underlying technologies and transparently provided through a pervasive interface. Daidalos demonstrates the results of the projects work through a strong focus on user-centred and scenario-based development of technology. Furthermore, as an "integrated project" it requires integration of individual components developed in WP2, 3 and 4 into a common prototypical solution. This integration, according to the Daidalos working structure to be fulfilled by Work Package 5, reaches a level of complexity in terms of number of components and their dependencies to be included, number of partners involved as well as time and budget constrains which is almost a unique experience in European R&D programmes.

Innovative targeted areas within Daidalos focuses on

- Layer 2-3 in order to develop an integration of heterogeneous network technologies. Such as Terminal mobility, Ad-hoc systems.

- The Services and Network Management level above layer 3. Such as QoS, A4C, Multimedia service platform.
- Pervasive systems, which involves innovations within network technologies and software infrastructures. Such as context management, Personalisation, Pervasive service management, security & Privacy.

These three main work areas were integrated into a single architecture driven by a scenario-based design approach. Finally, the integration of the projects innovative achievements into practical assessment allow for concept validation, system evaluation and public demonstration.

Due to the previously mentioned large amount of components, dependencies between them and the number of players involved in the integration tasks, a detailed integration planning and effective test bed operation have been crucial in Daidalos for a successful integrated Testbed. Strategies identified and led by an Integration management team and followed closely by the key stakeholders (Daidalos Management and Work Package Leaders) were set up to cover various integration aspects required for the Official testbeds, such as Testbed requirements, Integration components delivery planning and other key milestones identified for the integration process. Details on such milestones used will be discussed within this paper and it will also provide you with an insight into some of the positive and negative aspects of the processes adopted and the results achieved following the completion of the daidalos integration and validation phase.

II. STEPS LEADING TO INTEGRATION

Building up the system from components and packages of components was a nontrivial task because each time logical entities were mapped to physical devices, they

also had to be transferred to the larger test bed site with physical mapping specific to this site. The main steps leading to integration were identified as the following.

1. Decide on scope of integration
2. Prepare a scenario
3. Build a deployment model
4. Prepare and validate MSC
5. Build a Testbed
6. Gather all needed components (delivery procedures)
7. Build network architecture based on deployment model
8. Install and configure components (base on deployment model,)
9. Prepare and perform test cases

It is to be noted that not all elements of integration design patterns [2] suitable for a software production company can be applied to Daidalos.

The following figure 1 depicts the general defined integration and testing flow for all daidalos demo sites:

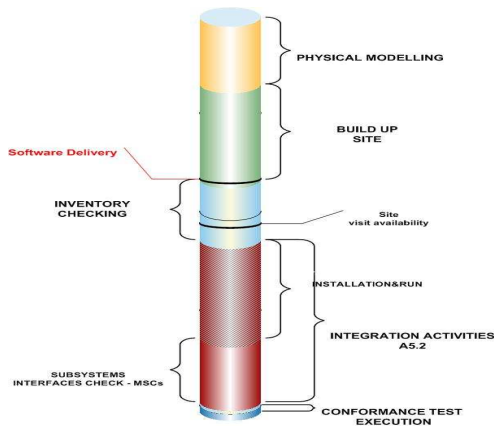


Figure 1. General Daidalos Integration Flow

III. TESTBED INTEGRATION PLANNING

Testbed Integration Planning is an important phase to identify the scope of integration and to sufficiently plan and manage the integration process. To help this planning stage, A 'Handbook for Daidalos Integration-oriented Developments' [3] adapted known best practices in open software development as well as experience from previous EU projects to the needs of a large IP projects in the areas of methodology and organisation. They encompassed pre-integration activities, configuration and installation requirements, component identification and tracing changes.

With the complexity of the Daidalos system and its operation at different OSI layers it was difficult for work packages to develop unified integration views within their respective subsystems. Functional subsystems were initially pre-integrated within each

work-package, then a cross work-package incremental process is executed. Integration was split into 3 stages

1. PreIntegration
2. Subsystem Integration
3. Intra WP Level Integration

Not all these stages were applicable to each work-package, it was dependant on their software structure and subsystem definition.

Scenarios in Daidalos were used to guide and merge the development of the various technologies and conceptual models. The detailed description of the initial university and automobile scenarios was quite extensive, and complete scenario descriptions have been only used to derive business modelling and architecture work, in addition to overall guidance of the technical development in the project. On the other hand, portions of these two scenarios were chosen for defining and describing an integrated demonstrator. It was decided early in the project that, in order to promote technical integration and to fully understand the challenges of integrating various enabling technologies into a working piece of system, the project should aim at one integrated demonstrator on which most of our integration work would be focused.

The selection of a sub-scenario for this integrated demonstrator (called Nidaros) was guided by many factors: most innovative and useful technologies to be demonstrated, most promising technical development since project started to be included, combining the two scenarios into one demonstrator, demonstrating key concepts such as mobility, broadcast and pervasiveness, etc. We believe Nidaros demonstrates a balanced (however only partial) view of the innovations that Daidalos is trying to achieve.

IV. MANAGEMENT OF INTEGRATION

In a project of the magnitude of Daidalos, where deficiencies in workflow management immediately causes wasted effort, communication was one of the biggest challenges.

A mixture of integration meetings (3 of them) for the whole consortium, audio conferences, e-mail lists, collaboration server (MoreGroupware) and cvs for software versioning were used for exchange of information, documents and software.

Toward the end of the project wiki was widely adopted in WP5 and found increasingly effective. Throughout the Daidalos Phase I Integration stage the Wiki tool was used as a support tool during integration, to aid the integration planning process, to document integration status reports, to actively supply the latest news during component installation and execution and to provide a detailed list of Testbed and developer contact details. It was extremely helpful for integration information exchange and changes in software as well as status of integration and development work.

In order to assist the Integration management of the Daidalos project, the following three Integration steps were identified.

Step 1: Deployment View : This was linked to modelling (Telelogic Tau Tool) [4] and was the first approach for distribution of components to nodes.

Step 2: Physical Modelling: This was comprised of a detailed list of software *components* with the software and hardware requirements.

Step 3: Physical Mapping: This was comprised of the final Layouts for the test-sites and involved the following activities.

- 1) Gather the list of components that are used.
- 2) Get the specific component requirements in terms of software, dependencies etc.
- 3) Define the deployment view which also includes e.g. network hardware
- 4) Create the physical modelling which means defining the distribution of components onto physical nodes

V. DAIDALOS TESTBED

Daidalos Official testbed sites were selected from a list of candidates within Daidalos partners considering a number of requirements (such as UMTS/WLAN coverage, transport facilities, availability of specific equipment, broadcast availability, logistics for demonstrations/workshops and expertise from former projects, among others) to decide which sites were more suitable for the chosen scenarios: automotive and university. The three Daidalos Official Sites that were finally selected for their fulfilment of all requirements to be used in Work Package 5, were Stuttgart (Germany), Aveiro (Portugal) and Sophia-Antipolis (France). Figure 2 shows the official Demo Site of Sophia Antipolis:

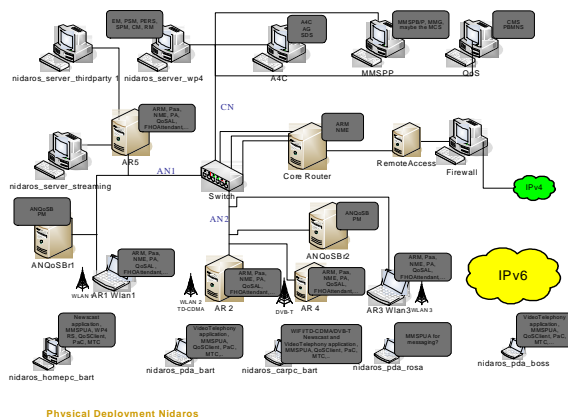


Figure 2. View of Sophia Antipolis demo Site

The Official Test Sites were provided with already-tested modules from work packages 2-4, at different delivery versions. Developed modules from WP2-4

were integrated, tested at system level and then validated in the Official Sites

Demonstration of the full Nidaros scenario was a major challenge in terms of balancing smoothness of demonstration with explanation and visualization. Where functionality of platform components is difficult to illustrate due to the fact that the components are not visible on the actual user interface, visualisation tools provide a means to convey the sequence of operations occurring behind the scene.

The demonstration is inherently mobile, ranging from the home location (i.e. Bart's HomePC) to the CarPC and the "simulated" airport location.

Visualisation, however, helps the audience to stay in synchronism with the demonstration even if they cannot be in several places at the same time (e.g. to witness a handover from a HomePC service to the CarPC).

Visualisation was useful during the initial integration phase. The following figure 3 conveys an example of the DLR visualisation tool created and used specifically for the daidalos project to enhance the visual aspect of the pre integration phases. This visualisation tool shows the simulated status of the network components and the messages from the major WP4 subsystems and third party services. The colour of the arrows indicates the presence of recent message traffic over these connections.

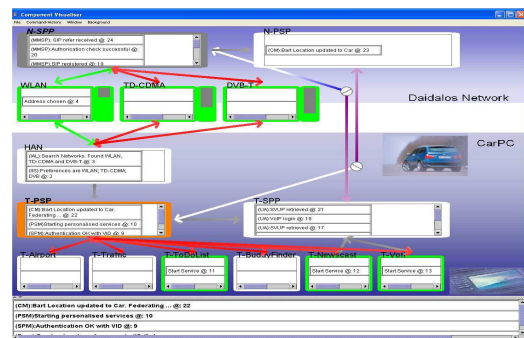


Figure 3. Visualisation Layout for the CarPC

VI. CONFORMANCE TESTING

The final deployment of Nidaros was in the Aveiro Testbed but the first integration of all the components involved in this scenario was in the Sophia Antipolis test site, but continuing problems with network stability, related to Mobile IP, prevented a full detailed conformance test at this site.

The conformance testing specification for the Daidalos demonstrator called the Nidaros – was developed in Deliverable D512 Conformance Test Specification [4], which was the basis for test process both in Sophia Antipolis and Aveiro testbeds.

A set of defined taskforces were involved in the procedure of the realisation of the conformance tests:

- **Aveiro Test-bed Taskforce:** In charge of managing the test-bed.
- **WP2 Integration Support Taskforce, WP3 Integration Support Taskforce, and WP4 Integration Support Taskforce:** Experts from the different Work packages were supporting the activities of the Aveiro test-bed taskforce.
- **WP5 Conformance Tests Taskforce:** A group of people from WP5 that were in charge of carrying out the tests and collecting the results, following the guidelines of D512 [4].

Due to the definition of the tests and the scenario itself the biggest problems we faced were those related to the network layer, Mobile IP and WP2 network drivers, which were unstable. It was decided to fragment the tests and start testing and debugging small pieces of functionality.

Different plans of action were defined, in which the tests were started with just one mobile terminal, then progressively adding more into the system in order to increase the complexity gradually.

The main challenge that comes with complex scenario's like Nidaros is to keep this complexity away from the user. This puts a heavy burden on the service platform and on the enabling services itself. The key factor is, to know as much as possible about the user's situation and preferences and to use this information for a continuous reconfiguration of the services itself.

The Nidaros scenario is also composed of the following 3rd party sub-services:

1. **Newscast-Service** –When the scene starts, Bart is watching a personalised newscast at home.
2. **ToDoList-application** – This application acts like an automated filofax.
3. **SIP-Service** – is used to make voice-calls, send messages, redirection, voicemail & presence management
4. **Location-Service** - is used to find location information.
5. **Traffic Information -Service** - provides airport information to the Car-PC
6. **Airport Information -Service** – is used to provide Bart with airport information.

The conformance tests were split into two main areas, Nidaros step Specific conformance tests and Nidaros performance tests. Figure 4 conveys statistical representations of results obtained during an Aveiro conformance test validation iteration.

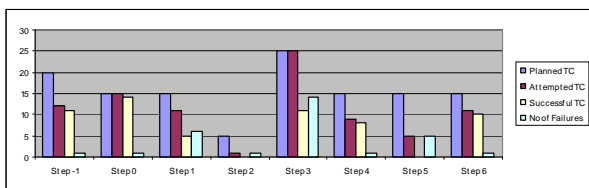


Figure 4. Nidaros Conformance Test Statistics: Aveiro iteration

When faced with a large number of components, both developed by Daidalos and also third party software, a minor problem in one of them can compromise the whole system. Some hard decisions were taken to achieve a stable integrated platform so as to be useful for our future work. Because we use third party software, which was creating stability problems, we needed to update this software. That required a change in Linux distribution and kernel version, which achieved the expected result of getting a more stable system. But, on the other hand, because of the time spent on this change, we finished with extremely hard deadlines for the conformance testing.

Using a scenario based validation process proved to be the most practical and useful way to convey the innovations of the individual work groups from the Daidalos project. The following text conveys a sample of a selected excerpt from the overall defined Nidaros scenario.

Bart is driving on the motorway, which triggers the launch of the Traffic information Service. Bart arrives at the airport, and continues to drive towards the car parking area of the airport. His car system screen shows the airport information carousel. Information about all flights will be shown. Bart parks and leaves his car, taking his PDA with him.

A transfer is invoked on the transferring components in the old service and starts the new service on PDA and stops the old service on CarPC. The airport arrivals information can now be seen on Barts PDA as he enters the airport arrival terminal.

VII. SAMPLE TEST CASE EXECUTION

Best testing component/subsystem practises can be classified into several groups [5]. The most obvious debugging and testing techniques, widely recognized and documented are listed below:

- Development of Functional Specifications.
- Reviews and software inspections.
- Formalisation of preconditions and postconditions.
- Usage of Functional tests and Variations.

The Daidalos research project is not concerned with production testing quality, rather demonstration and verification of solutions from the end user point of view. Such purpose requires scenario conformance Testing.

This section presents an example of one of the Nidaros conformance tests used within the conformance validation phase.


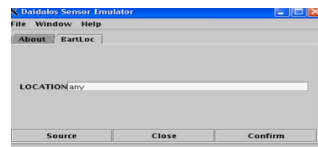
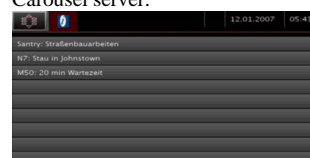
N3:05: - Launch of traffic service due to location change.	
Steps	Verdict
1. TESTER: Ensure that other applications such as Newscast are terminated (or if Traffic Info Service is already running as a consequence of the previous test, N3:04, this should also be stopped)	PASS
2. SUT: Car PC is in main menu mode. It takes time for the BMW GUI to come up properly, so wait until you see the 'INFO' button appearing and select it. 	PASS
3. TESTER: Trigger context change using Generic Sensor Emulator to indicate that CarPC location is now on the Motorway (this is also done as part of TC N3:04, so if the test is being performed separately it may be necessary to change the location sensor to an initial value which is not the motorway). 	PASS
4. SUT: The traffic information system should start automatically and become populated with the information being broadcast from the Carousel server. 	PASS

Table 1 Detailed Test Execution: N3:05

Test Result: Overall pass.

Comments: The PSP-Container was started on the carPC. Using the generic sensor emulator, location was set to 'any', to indicate that the carPC was on the motorway. The test was executed four times. It failed one time due to a wrong configuration but was successful the other three times. The Newscast finishes and a simulated location event (Bart is on the motorway) triggers the launch of the Traffic Service (Bart has heard enough of the news and presses the Stop button on the CarPC Newscast GUI. Now all unicast sessions have stopped, DVB-T is started/activated. DVB-T carries Traffic Info in the Carousel) and Personalisation sub-system to change network preferences from:

- | | | |
|------------|---|------------|
| 1. WLAN | ⇒ | 1. WLAN |
| 2. TD-CDMA | | 2. DVBT-T |
| 3. DVB-T | | 3. TD-CDMA |

Personalisation sends these new network preferences to IIS, IIS sends this information to the mobile terminal controller which in turn activates the DVB-T bridge.

The following actions describe additional system execution steps to describe how to use the system upon arrival at the airport.

Car comes within range of the Airport

- In the 'Location' tab, enter the location value of Bart as **car@airport**, then wait
- You should see the Traffic Service recomposing itself into an Airport Service
- The Airport Service should begin displaying Airport information
- Log into Barts PDA (wait until fully logged in)

Bart exits car at airport and walks to terminal

- Now in the Location tab of the Sensor Emulator, enter Barts location as **airport**, then wait
- You should see the Airport Service from the CarPC transfers to the PDA (using an appropriate GUI display on the PDA)
- You should see Airport Information being displayed on Bart's PDA.

VIII. TESTBED ISSUES DURING SETUP AND RUN PHASE

With such a large scale project as Daidalos, it was inevitable and expected that problems would occur during test bed deployment, Integration and validation phases.

Packages dependency issues also had a negative impact, for example lost corrections, poor versioning (using older versions of components and libraries), incompatible version on operating systems, and network administration and routing problems.

Third Party software, which Daidalos partners had no control over, was responsible for major issues with integration and a change of Operating System was required before this software functioned correctly (e.g. Mobile IP). Specific issues encountered during the run process appeared in many areas in addition to errors in code:

- Timing for sessions/certificates, such as session terminations due to certificate expiration (in CAN), Stability problems (faulty state machines or even memory leaks)
- Settings unsynchronised scenario changes, problems with parallel works on 2 machines (remote restarts interfere)
- Creating dummy adapters direct access component omitting the correct path, Incomplete or faulty definition of interfaces (e.g.: Care of Address instead of Home Address), Faulty interfaces
- Lack of tolerance on some events: restarting server forces a restart of all clients, sometimes a container can be used only as a "transport

unit”, need to write scripts starting only parts of a container

IX. CONCLUSIONS & RECOMMENDATIONS

Daidalos phase I provided a unique experience for assessment of methodologies and technologies in software engineering for a very complex telecommunication systems. Surely, complexity of integration has been underestimated. Also communication, particularly interworkpackage communication, became quite a problem for a team of 200 developers and integrators. For some large commercial system a core team does not exceed 100 members. The most unpredicted proved to be small reliability of crucial external systems and severity of this problem was at the verge of being managed by Daidalos. Also developer churn became a factor, it is estimated to be 20% but for certain activities the percentage could be as high as 30%. Kernel modules were most critical. Change of distribution from mandrake to Ubuntu mostly solved a problem of MIPL, but introduced some software libraries dependencies.

There have been two major rounds of collocated Nidaros integration testing; one in November 2005 at Sophie Antipolis, before audit; and the final one in March 2006 in Aveiro. The Sophie Antipolis test was distorted by instability of Mobile IP functionality for integrated system. In between, there was continuous effort with remote installation and testing. Most of the components and interfaces planned for Nidaros have been available for the Sophie Antipolis cycle although not all of them performed successfully in the integrated Nidaros system. Due to complexity of the Nidaros system a change of underlying platform such as Linux kernel could cause non-compliance of previously working subsystem

Running the test site was a massive operation in terms of availability of sufficient logistics: number of computers and devices, manpower to configure network and computers and install components. The Daidalos consortium could not for financial and logistics reasons keep test sites operation on continuous basis. The test had to be prepared before each major test cycles. This meant lack of proper network and system configuration first 2 or 3 days of a major test cycle. The following recommendations were identified as suitable enhancements to the Integration Phase.

- On site integration must be improved with a stable basic platform very early in the integration phase. A permanent test-bed should be identified, which is accessible for all partners.
- All integration activities during technical weeks should be held at the identified test sites.
- In order for the modelling process to contribute to the integration process, this may work if targeted testing is linked with modelling.

During daidalos I the modelling process only really contributed to documentation, but has the possibility to be more constructive for the integration phase once the correct process and methodology is implemented.

- It is recommended that certain experts from each Work Package arrive early to the test site and check out that the test-bed deployment is suitable for their software to be deployed on, any inconsistencies should be raised at that stage rather than when the integration team arrive on site at a later date.
- There is a need to implement more effectively pre-integration test-beds, running a common Daidalos architecture and performing intra-WP integration prior to the scheduled inter-work package integration cycles.

Most harmful reasons of failures during the integration process are configuration settings such as: hard-coded IP addresses, session/certificate lifetimes, incorrect port numbers, etc. These kinds of errors may be undetected until integration starts. That's why developers should always use configuration files instead hard coding. It is easier and less time consuming to correct a configuration file than fixing poor coded software which involves a complete compilation/build/reinstallation cycle. We have to remember that configuration files save developers time – any one on testbed with adequate knowledge can fix it without disturbing a developer. Also developers should consider using stable well known software frameworks for WP integration. For example for logging, testing, visualization and service management. OSGi framework and logging framework used in WP4 proved that frameworks can save time by giving additional functionality for a small additional work cost. Another lesson learned from integration is that remote access to the testbed should be more restricted during demos or conformance tests to avoid disruptions or negative impacts on a previously working testbed accidentally. This implies also that a partner responsible for a testbed should have more control over installed components and their versions. Some clear control procedures have to be elaborated.

REFERENCES

- [1] DAIDALOS - Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services (EU Framework Programme 6 Integrated Project), <http://www.ist-daidalos.org/>
Steve Berczuk , Brad Appleton, *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, Addison-Wesley, 2002.
- [2] D513 handbook for Daidalos Integration-oriented Developments'
- [3] <http://www.telelogic.com/corp/products/tau/index.cfm>
- [4] D512 Conformance Test Specification
- [5] James A. Whittaker, Florida Institute of Technology. "What is Software Testing? And Why it is so hard?" <http://www.sba.oakland.edu/faculty/rajagopa/Courses/515/testin g2.pdf>