Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Grant Agreement Type: Collaborative Project
Grant Agreement Number: 611391

D3.4.2
System Integration Progress Report

Due date: **1/4/2016**
Submission Date: **18/4/2016**

Start date of project: **01/04/2014** Duration: **54 months**

Organisation name of lead contractor for this deliverable: **University of Skövde**

Responsible Person: **D. Vernon** Revision: **1.3**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Executive Summary **3**

Principal Contributors **4**

Revision History **4**

1 Overview **5**

2 Software Development Questionnaire **6**

3 YARP Component Generator **7**

4 Streamlined Integration Procedure **7**

5 Software Integrated into the Release Version **8**
5.1 Components Submitted by Vrije Universiteit Brussel 8
5.1.1 actuationSimulator 8
5.1.2 eyeBlinking 8
5.1.3 fallingReaction 9
5.1.4 socialReaction 9
5.1.5 naoInterface 9
5.1.6 reactiveSystemGui 10
5.2 Components Submitted by University of Skövde 10
5.2.1 sensoryInterpretationLogger 10
5.2.2 componentChecker 10

6 Schedule for Rolling Out Functionality of the DREAM system **10**

References **11**

Executive Summary

Deliverable D3.4 is an annual progress report of the integration on the software developed in work packages WP4-WP6 in the release version of the DREAM system. This is the Month 24 progress report.

All partners responsible for developing software in DREAM are very active. Several prototype capabilities have been developed and demonstrated informally and an extensive amount of software having been committed to partners directories on the DREAM SVN repository. However, only a small number of software components have been integrated into the release version of the DREAM system. To determine the reasons for this, a questionnaire was distributed to canvass the views of developers on the DREAM software engineering standards and procedures. The responses to this questionnaire flagged a need to clarify, streamline, and simplify the software integration procedures as orginally set out in Deliverable D3.3: Quality Assurance Procedures. This has now been done and the revised procedure has been documented on the DREAM wiki [1] and the standards related to component functionality have been relaxed slightly. In parallel, researchers at Plymouth University developed a utility application — a YARP Component Generator — which provides an easy way to generate the structure of a component that, in principle, conforms to the DREAM standards. Once adopted, this will provide a way of generating software components that can be fast-tracked through the quality assurance procedures prior to integration. Finally, steps have been taken to create a schedule of for rolling out the functionality of the DREAM system, partly in response to Recommendation 3 in the first review report and partly in an effort to secure a firm commitment from the partners to submit software for integration into the release version in a timely manner.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Erik Billing, University of Skövde
Pablo Gómez, Vrije Universiteit Brussel
Emmanuel Senft, Plymouth University
David Vernon, University of Skövde

Revision History

Version 1.0 (DV 08-04-2016)
First draft.

Version 1.1 (DV 12-04-2016)
Adapted to reflect revised integration procedure.

Version 1.2 (DV 15-04-2016)
Updated list of software integrated into the release version.
Added note about integration compliance check utility.

Version 1.3 (DV 18-04-2016)
Updated list of software integrated into the release version.
Added note to refer to the online component reference manual on the wiki.

1 Overview

All partners responsible for developing software in DREAM are very active. Several prototype capabilities have been developed and demonstrated informally and an extensive amount of software having been committed to partners working directories on the DREAM SVN repository[1] (see Fig. 1). However, only a small number of software components have been integrated into the release version of the DREAM system.

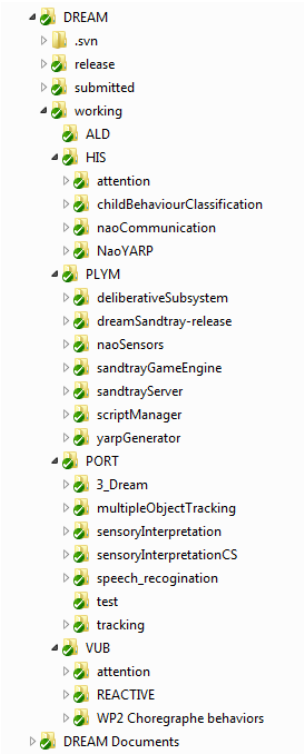

Figure 1: Software in the Working branch of the DREAM SVN repository.

To determine the reasons for this, a questionnaire was distributed to canvass the views of developers on the DREAM software engineering standards and procedures. The responses to this questionnaire flagged a need to clarify, streamline, and simplify the software integration procedures as orginally set out in Deliverable D3.3: Quality Assurance Procedures. The revised procedure has been documented on the DREAM wiki [1] and the standards have been relaxed slightly. Specifically, five of the mandatory standards relating to component functionality have been reclassified as recommended standards. This gives the developers more freedom when it comes to the manner in which they decide to design the functional aspects of their code while at the same time keeping a tight rein on the file structure, the software documentation, the component re-configurability, and component tests; effectively, all the things are critical for successful system integration.

In parallel, partners at Plymouth University have developed a YARP Component Generator. This is a utility application which provides an easy way to generate the structure of a component that, in principle, conforms to the DREAM standards, including different required files and folders, and the code needed to use YARP ports. Some minor issues with compliance with standards arising from the

[1]The DREAM SVN repository can be accessed here https://dreamproject.aldebaran.com/svn/dream/DREAM/.

streamlining of the integration procedures and the software engineering standards are currently being addressed.

Steps have been taken to create a schedule for rolling out the functionality of the DREAM system, and therefore integrating it into the release version. This exercise is being done partly in response to Recommendation 3 in the first review report and partly in an effort to secure a firm commitment from the partners to submit software for integration into the release version in a timely manner (i.e. in a timescale that will allow third-party use and validation of the software with subsequent revisions, whenever necessary).

In the following five sections, we provide an overview of the questionnaire on software engineering standards and short summary of the revised wiki DREAM Software Integration Guide. We follow this with an overview of the YARP Component Generator and the software that has been integrated so far in the release version of the DREAM system. We conclude with an overview of the planned schedule for rolling out functionality over the remainder of the project.

2 Software Development Questionnaire

A poll of developers' views taken approximately one year after the launch of the standards, quality assurance procedure, and the operational system architecture components and example application. This poll was taken to determine what could be improved in the software standards, integration procedure, system architecture, and support documentation and software, and to gauge the impact of the operational system architecture on their engineering practices in the context of the DREAM project [2].

The poll took the form of a multiple choice questionnaire with twelve questions formulated in a manner that elicited feedback, positive or negative, as well as inviting comments on what needed to be removed, changed, or added for each question topic. The initial version of the questionnaire had more questions but we reduced the number to increase the likelihood of obtaining meaningful responses. Each question offered five choices: Strongly agree; Agree; Neither agree nor disagree (I don't know / I don't have an opinion); Disagree; and Strongly disagree. Questions were formulated both positively and negatively so that some questions required agreement to express a positive response with other questions required disagreement to express a positive response. In this way, we attempted to eliminate biased responses as a consequence of the manner in which the questions were framed.

Furthermore, views on each issue were elicited using more than one question; conclusions are drawn only if the answers are consistent.

The questionnaire was completed by each team of developers in the consortium. This resulted in four sets of responses, representing the consolidated views of each partner involved with developing software in DREAM.[2]

The results of the poll were positive: developers felt that the balance of mandatory and recommended software engineering standards allows sufficient flexibility to design their own code and that the time and effort required to familiarize themselves with the mandatory standards was worthwhile. Significantly, they did not think there should be fewer mandatory standards and they agreed that the support software, i.e. the protoype component and example applications, helped illustrate the standards and assisted them in writing their own components. Crucially, there was a strong consensus that the operational system architecture made it easier to design software that can be integrated into the DREAM RET application and they prefer this approach to a more standard paper-based description

[2]The questionnaire and an anonymous version of the results are available at www.dream2020.eu/software_questionnaire.

of the architecture. There was also agreement that the system architecture provided sufficient specification of how their software should interface to other subsystems. With one exception, they also felt that the working architecture with operational port interfaces makes it easier to design software that can be integrated (the operational port interfaces makes it harder in the sense of raising the threshold on correctness before a component can be submitted for integration).

The news was not all good, however. The responses to the questionnaire clearly signalled that developers were less impressed with the clarity of the integration procedure and emphasized the need for them to be streamlined.

The questionnaire responses also suggested that integration could be fast-tracked if the software component was generated by the YARP Component Generator, a utility application that was developed by partners at Plymouth University. Fast-tracking integration would effectively by-pass the quality assurance procedure (documented in Deliverable D3.3 and on the DREAM wiki) that currently involves manually checking the software against the checklist of conditions that a software components must satisfy if it is to be accepted for integration into the release version. We have agreed in principle to take up this suggestion and will do so in practice once we have checked that the Generator does in fact create components that are compliant with the new streamlined standards and procedures.

The next section provides a short overview of the YARP Component Generator. This is followed by a summary of the streamlined integration procedure.

3 YARP Component Generator

If one wants to write or change a component that is compliant with the DREAM software engineering standards, several files have to be modified, even if it affects just one port. While the Software Development Guide on the DREAM wiki is comprehensive, and is supported by example code on the wiki and in the SVN repository, there is no denying that fact that it takes considerable effort to write software that is compliant with the standards. Most of this overhead if fixed, in the sense that it is routinely the same code that is used to handle the configuration, communication, and coordination of the component, i.e. three of the four Cs in the Component-based Software Engineering methodology adopted by the DREAM project. Partners at the University of Plymouth pointed out that this process can be automated and they created a utility application — the YarpGenerator — to do exactly that. The following overview, adapted from the introduction to the accompanying user manual [3], explains the rationale and design of this utility.

The purpose of the YarpGenerator utility is to provide an easy way to generate the structure of a component that conforms to the standards, including the different files and folder required, and the code needed to use Yarp ports. Additionally, it provides a means to package this standard aspect of the source code, if desired, separately from the computation code of the component.

This utility enables the generation of compilable code, which can be added to a project (such as the DREAM integrated system) with minor additional modifications. Furthermore, it reduces the complexity of using Yarp ports by centralising the send/receive functions, and by providing if desired a simpler interface to access to the Yarp functionalities. For input ports, callbacks are used to enable event-based processing, without having to manually check for new information.

4 Streamlined Integration Procedure

The software integration guide on the DREAM wiki [1] has been substantially revised and streamlined. It now provides

- The procedure that should be followed to test a component before submitting it for integration.
- Guidelines on what files should and should not be submitted for integration and how these files should be organized in directories.
- The procedure that should be followed when actually submitting the component for integration.
- A revised and reduced checklist of conditions that a component must satisfy before it can be included in the release version of the DREAM software.
- The procedure that will be followed by the system integrators to integrate the component into the release version of the DREAM system.

The original version of the checklist appears in the quality assurance procedures described in Deliverable D3.3. This checklist is derived from the manadatory standards in Deliverable D3.2, Appendices A-C. A description of the system architecture, i.e. the placeholder system components and their associated ports, is included in the wiki System Architecture Guide for reference. This is described in more detail in Deliverable D3.1.

Partners at the University of Skövde have developed a utility to semi-automate the process of checking submitted software against the revised and reduced checklist of conditions mentioned above.

5 Software Integrated into the Release Version

Eight components have been integration into the release version in the past year. These are described briefly in the following sections. Please refer to the online component documentation on the DREAM wiki for more details of all these components.[3]

5.1 Components Submitted by Vrije Universiteit Brussel

5.1.1 actuationSimulator

This is the implementation of the actuationSimulator placeholder component. Its functionality is to provide the naoInterface component with the corresponding action primitives (moveSequence or say) given the outputs from the Reactive subsystem (fallingReaction, socialReaction and eyeBlinking).

The purpose of this module is to provide a preliminary actuation subsystem to send the reactive subsystem outputs to the corresponding robot.

5.1.2 eyeBlinking

This is the implementation of the eyeBlinking placeholder component. Its functionality is to provide the actuation simulator component with the corresponding blinking behavior given the outputs from the sensory information and social reaction, following a simplified version of Ford's blinking model [4].

Within the context in which DREAM will be applied, we need to recreate a blinking behavior mainly focused on the communicative behaviors and gaze shifts. For such reason, we consider that Ford et al.s model covers these needs and provide accurate data to implement their model.

[3]The on-line component documentation can be accessed here
https://dreamproject.aldebaran.com/svn/dream/DREAM/release/doc/component_doc/index.html

Ford et al. defines a model which considers multiple communicative facial behaviors and includes an individual blinking model for each of them. For each identified communicative behavior there is a probability to blink, a determined length, and so on. Moreover there is a passive behavior which simulates a physiological blink mechanism (for cleaning or humidifying the eye) that can be activated when no other blinking behavior has been triggered. To perform the blinking motion there is a blink morphology model which defines, based on statistics, if the blink is simple or multiple, full or half, its duration, etc.

5.1.3 fallingReaction

This is the implementation of the fallingReaction placeholder component. Its functionality is to provide the actuation simulator component with the corresponding falling reaction and speech given the outputs from the sensory information. Additionally, an interruption and a recovery signals will be sent when needed.

The Falling Reaction module will be periodically checking the balance of the robot using the sensory information available. Changes in the balance may end up in a fall. In such case, a signal will be sent to the Self-Monitoring subsystem to interrupt any other running behavior, and a damage avoidance behavior that fits the situation will be triggered. Since the robot will be placed in a table and in case it falls it will be into the floor from certain height, there is no actual need to implement getting up behaviors. However, as the Nao robot includes such behaviors they will be taken into account. Additionally, the robot should include some speech acts to reduce the impact of such dramatic situation for the kid as saying that it has been a little bit clumsy or that it is tired today. Finally, back at its feet, the robot may apologize in order to engage the child back to the intervention or call the re-engagement module in the Deliberative subsystem and it will send a signal to the Self-Monitoring subsystem to restore the system functionality

5.1.4 socialReaction

This is the implementation of the socialReaction placeholder component. Its functionality is to provide the actuation simulator component with the corresponding social reaction, speech and facial expression given the outputs from the sensory information.

The purpose of this module is to provide the appropriate social behavior in order to give the impression of the robot being socially alive. This module receives as input the sensory information where it is specified the childs social and affective state i.e. whether she/he is expressing an emotion or is performing a physical behavior (such as touching the robot unexpectedly). For each of these behaviors there should be a set of facial expressions and speech acts available to choose among them. Ideally it should randomize among them in order to look less predictable. Such reactive facial expressions and speech acts should be defined by the therapists and will be stored in the library of the Actuation subsystem. The functionality of this module can be switched on and off when needed through the Self-Monitoring subsystem.

5.1.5 naoInterface

This is the implementation of the naoInterface placeholder component. Its functionality is to execute in the Nao robot those action primitives sent by the actuation subsystem. This is not part of the reactive subsystem.

5.1.6 reactiveSystemGui

This is a driver component to facilitate unit test of the foregoing components. It is not part of the final release, and, as such, it does not need to comply with the DREAM software engineering standards.

5.2 Components Submitted by University of Skövde

5.2.1 sensoryInterpretationLogger

This is a utility component and does not form part of the DREAM system architecture. It reads data from specified input ports and logs to a file. The component can be configured to log any number of input ports with a limited number of data formats. Two modes for logging are supported: Synchronous and Event Based, each having a different csv format.

5.2.2 componentChecker

This is also utility component. It is used to semi-automate the integration process by checking the submitted code against the revised and reduced checklist of conditions that a component must satisfy before it can be included in the release version of the DREAM software, as noted in Section 4 above.

6 Schedule for Rolling Out Functionality of the DREAM system

Recommendation 3 in the First Review report requires the creation of a schedule for rolling out the functionality of the DREAM system. Specifically, it states that the consortium should:

> ”Create a specific timetable for the delivery of the different modules/components of the project in order to build a good evaluation plan of the system at different stages of development and integration. Specifically, better articulate the bidirectional flow between WP2 and WP3 to roll out partial functionality.”

At a meeting in Amsterdam on the 18th February 2016, the principle investigators from each partner committed to producing such a schedule in the form of a list of components that will be delivered for integration, specifying for each component:

- Its functionality
- The primitives in Deliverable D3.1[4] that it implements (if any)
- The ports used from the system architecture (if any)
- Other ports used (if any) and the associated data protocol (specified in format similar to the system architecture ports)
- The planned date it will be submitted for integration

Regarding functionality, it was agreed that the information produced by each component should be complete in so far as it should provide all of the information specified in the system architecture, although it might possibly provide only an approximation of that information (e.g. limited accuracy of values or use of dummy values).

[4]The primitives in Deliverable D3.1 are derived from the specification of robot behaviour in Deliverable D1.2 and child behaviour in Deliverable D1.3.

Regarding inter-component communication, it was agreed that each component should strictly adhere to the port protocols specified in the system architecture. This applies only to components that override functionality exposed by the three system architecture components (i.e. sensoryInterpretation, childBehaviourClassification, and cognitiveControl) on the specified ports using the specified protocol associated with each primitive. Conceivably, there may be some sub-system components that don't directly implement any of the D3.1 primitives but instead send or receive their data to or from other components that *do* input and output data via the ports in the system architecture.

At present, it is planned to complete this schedule by the end of April 2016.

References

[1] https://dreamproject.aldebaran.com/projects/dream/wiki/Wiki.

[2] D. Vernon, E. Billing, P. Hemeren, S. Thill, and T. Ziemke. An architecture-oriented approach to system integration in collaborative robotics research projects - an experience report. *Journal of Software Engineering for Robotics*, 6(1):15–32, 2015.

[3] E. Senft and P. Baxter. Guidelines to use the Naoqi-Yarp interface and the Yarp component generator. Technical report, DREAM Project No. 611391, 2015.

[4] C. C. Ford, G. Bugmann, and P. Culverhouse. Modeling the human blink: A computational model for use within human–robot interaction. *International Journal of Humanoid Robotics*, 10(1), 2013.