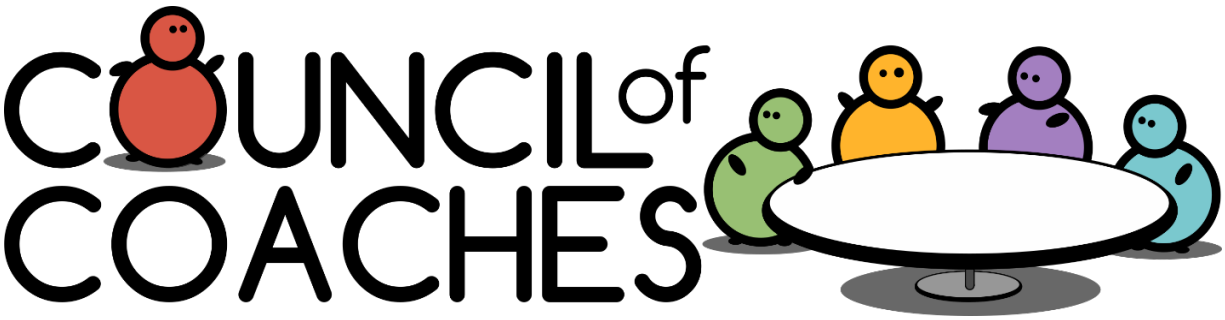

D5.4: Design of computational model based on theoretical and abstract model

Dissemination level:	Public
Document type:	Demonstrator
Version:	1.0.0
Date:	March 2, 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Document Details

Project Number	769553
Project title	Council of Coaches
Title of deliverable	D5.4: Design of computational model based on theoretical and abstract model
Due date of deliverable	February 29, 2020
Work package	WP5
Author(s)	Mark Snaith (UDun), Tessa Beinema (RRD), Daniel Davison (CMC), Randy Klaassen (CMC), Dennis Reidsma (CMC), Harm op den Akker (RRD)
Reviewer(s)	Álvaro Fides (UPV)
Approved by	Coordinator
Dissemination level	Public
Document type	Demonstrator
Total number of pages	14

Partners

- University of Twente – Centre for Monitoring and Coaching (CMC)
- Roessingh Research and Development (RRD)
- Danish Board of Technology Foundation (DBT)
- Sorbonne University (SU)
- University of Dundee (UDun)
- Universitat Politècnica de València, Grupo SABIEN (UPV)
- Innovation Sprint (iSPRINT)

Abstract

This deliverable provides a description and implementation of the Dialogue and Argumentation Framework (DAF), which represents a computational model based on the theoretical and abstract models previously reported in the public deliverable documents D5.1, D5.2 and D5.3.

Table of Contents

List of figures

Symbols, abbreviations and acronyms

API	Application Programmer's Interface
CMC	Centre for Monitoring and Coaching
COUCH	Council of Coaches
D	Deliverable
DAF	Dialogue and Argumentation Framework
DGDL	Dialogue Game Description Language
DGEP	Dialogue Game Execution Platform
DBT	Danish Board of Technology Foundation
CE	Coaching Engine
EC	European Commission
ISPRINT	Innovation Sprint
JSON	JavaScript Object Notation
M	Month
MS	Milestone
REST	Representational State Transfer
RRD	Roessingh Research and Development
SKB	Shared Knowledge Base
SU	Sorbonne University
UDun	University of Dundee
UG	Utterance Generator
UPV	Universitat Politècnica de València
UT	University of Twente
WP	Work Package

1 Introduction

This document of type “Demonstrator” accompanies the release of the first implemented version of Dialogue and Argumentation Framework (DAF) component, which represents a computational account of the theoretical and abstract models previously specified in Deliverables 5.1, 5.2 and 5.3. The document provides installation instructions for a demonstration version of the DAF (Section 3), and a description of the implemented system (Section 4).

2 Objectives

The objective of this deliverable is to provide a demonstration and description of the computational model that embodies the theoretical and abstract models reported on in Deliverables 5.1 through 5.3.

3 Dialogue and Argumentation Framework software

To demonstrate the Dialogue and Argumentation Framework (DAF), it has been bundled into a separate git repository with a web-based interface that allows a user to control the selection of dialogue moves provided by the framework:

https://gitlab.com/CouncilOfCoaches/daf

The demonstrator also allows for the editing and creation of dialogue protocols, creation and editing of content for use in dialogues, and the creation and editing of mock coaching variables for testing purposes.

3.1 Pre-requisites

The DAF uses Docker[1], which means all dependencies are loaded as part of the container build process and as such this pre-requisites list is minimal.

- A desktop or laptop PC running Windows, Linux or OSX/macOS, and that supports the Docker platform.
- Docker, with docker-compose installed.

3.2 Installation

While the installation instructions provided here are correct at the time of writing this deliverable, please also consult the README file for any new or updated instructions.

1. Download the code from the Council of Coaches repository; to access the code in GitLab you will have to provide us with your GitLab account in order to grant permissions, since it is currently private:

   https://gitlab.com/CouncilOfCoaches/daf/-/archive/master/daf-master.zip

2. Unpack the contents into any folder you want. We will henceforth refer to this folder as *{installation}*.
3. Open a terminal/command prompt/PowerShell window (depending on your operating system); for brevity, we use "terminal" in subsequent instructions.
4. Navigate to the *{installation}* folder in the terminal.
5. Run the command:

   ```
   docker-compose up
   ```

6. The containers will start to build. This process may take several minutes. During this time, you may see output in the terminal relating to a "ConnectionRefusedError"; this is normal.
7. When the DAF is ready, the following line will be printed in the terminal:

   ```
   daf_controller      | Dialogue and Argumentation Framework ready
   ```

3.3 Usage

1. Open a web browser and enter the URL:

   http://localhost:8080

2. When the page has loaded, select one of the four options: **Test a protocol**, **Edit protocols**, **Edit content**, or **Edit coaching variables**

3.3.1 Test a protocol

1. Select a protocol from the drop-down menu and follow the instructions.

[1] https://www.docker.com

2. After entering the name of the user, you will be presented with a simulation of the dialogue. You can select both the agent and the user moves and examine how the dialogue unfolds and progresses.

3.3.2 Edit protocols

1. Select a protocol from the drop-down menu, or choose **New protocol**.
2. Edit or create the protocol and click **Save**; if you are creating a new protocol, be sure to enter a name in the box.

3.3.3 Edit content

For the purposes of this demonstration, the argument models and dictionary are raw edits of the underlying JSON representation. We leave to future work a full toolkit that provides a user-friendly interface on the underlying data.

1. Choose either **Edit argument models** or **Edit dictionary**
2. Create or edit the content when it is displayed in the box

3.3.4 Edit coaching variables

The interface to edit coaching variables is provided to allow dialogues to be tested independent of the Shared Knowledge Base (SKB, see D3.5 (op den Akker, Beinema, Hofs, & van Schooten, 2019)). For this demonstrator, coaching variables are expressed in a JSON object as key-value pairs.

4 Architecture description

4.1 Overall description

The Dialogue and Argumentation Framework consists of two core modules connected in a message-oriented architecture via a central controller: Dialogue Game Execution Platform (DGEP) and the Utterance Generator (UG). A high-level overview of this architecture, along with relevant connections to other components in the overall Council of Coaches architecture, is shown in Figure 1 below.

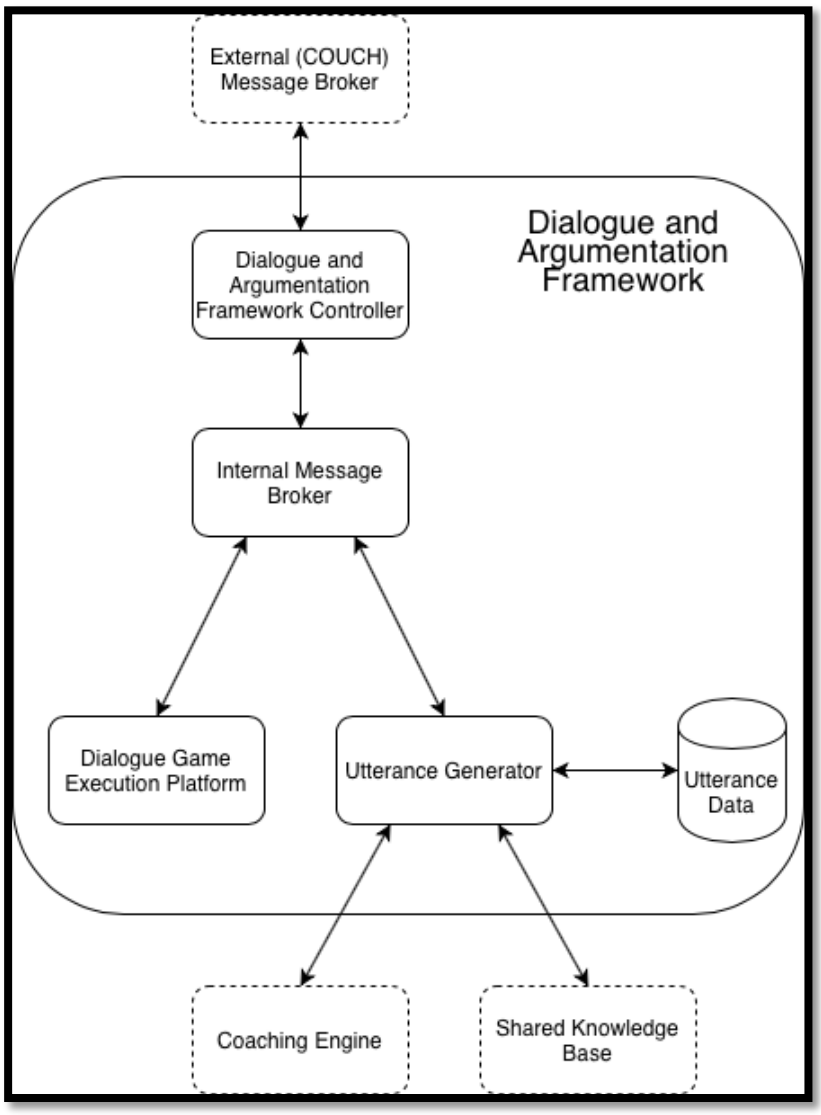

Figure 1: Dialogue and Argumentation Framework architecture.

In the following subsections, we describe the components in an order that is clearest from a cross-referencing standpoint, rather than the order in which they appear (top-down) in Figure 1.

4.2 Dialogue Game Execution Platform

DGEP is the primary module in the Dialogue and Argumentation Framework, interpreting and executing dialogue games expressed in Dialogue Game Description Language (DGDL). This version of DGEP has been fully reimplemented based on the version reported in (Bex, Lawrence, & Reed, 2014).

Dialogue game execution provides a structure to dialogues between coaches and users, and the coaches themselves. When a participant (coach or user) makes a certain move, DGEP processes that move and determines the legal next move type(s) and speaker(s).

4.3 Utterance Generator

The Utterance Generator (UG), previously called the "Filstantiator", filters and instantiates the abstract move types provided by DGEP. Filtering reduces the set of move types with respect to, for instance, a coaching strategy (see D3.3 (Beinema & op den Akker, 2018)) or particular coach personality. Instantiation populates move types with concrete content.

The UG receives input from a database of Utterance Data (described in Section 4.3.1), the Coaching Engine (CE) and Shared Knowledge Base (SKB). The Coaching Engine provides information relating to the coaching strategy being followed[2]. The Shared Knowledge Base provides values for variables in a dialogue, for instance the number of steps or exercise minutes a user should aspire to when setting a goal.

The UG communicates directly, via REST APIs, with the Coaching Engine (CE) and Shared Knowledge Base (SKB) in the overall COUCH system. This is because the communication reflects a simple request-and-response model, and thus a message-oriented approach would introduce unnecessary overhead.

4.3.1 Utterance data

Underpinning the Utterance Generator is a database of utterance data. This data consists of:

1. **Generic argumentation models** that allow coaches to provide arguments to support the advice they are giving the user. For instance:

   ```
   achievable_goal({current_goal_value}_{current_goal_type});
   achievable_goal(X) => set_goal(X);
   ```

   represents a generic model of an argument for setting a particular goal. The first line is an argument premise where {current_goal_value} and {current_goal_type} are variables in the Shared Knowledge Base (see Section 4.3), and the second line is an inference rule. Let us assume the values of these variables are "10,000" and "steps" respectively; then, the UG will generate a concrete premise:

   ```
   achievable_goal(10000_steps);
   ```

   This in turn allows for a concrete rule:

   ```
   achievable_goal(10000_steps) => set_goal(10000_steps);
   ```

   which in turn allows for an argument to be built. Using this argument, a coach will recommend the user set a goal of "10,000 steps", which if queried will be justified by saying it's an "achievable goal". Arguments are built from premises and rules using the TOAST system (Snaith & Reed, 2012) that implements ASPIC+, a theory of structured argumentation (Prakken, 2010).

2. An **utterance dictionary** that maps the logical structures used in the argumentation models, to sets of sentences that represent the linguistic content of those structures. Each mapping may contain different linguistic representations based on: a) the particular dialogue move type (e.g. the phrasing for a "question" is different to that for an "assertion"); and b) different delivery styles (e.g. Socratic, authoritative). To extend the example from the generic argumentation models, the following is an extract from the utterance dictionary for "set_goal(X)":

   ```
   "set_goal(X)" : {
               "propose" : {
                   "socratic" : "How about we set you a goal of $X?",
                   "authoritative" : "We should set you a goal of $X"
   ```

[2] In the overall COUCH system, the Coaching Engine also selects the dialogue game to be executed (in DGEP) via a topic selection model, however this is done outside the DAF, which only receives the name of the game to be executed.

```
},
"confirm" : {
    "*" : "Sure, $X sounds like a good goal"
}
```

In this example, "set_goal(X)" can be used in either a "propose" move (i.e. the coach proposing the goal) or a "confirm" move (i.e. the user confirming the goal). When a coach proposes the goal, it can be expressed either in a Socratic (gentle) style (as a question), or an authoritative style (as a statement). When the user confirms the goal, there is only one style, represented by "*". The value of $X is populated by the value in "set_goal", which in turn is obtained via the argumentation models.

The utterance data is stored in JSON format using MongoDB[3].

4.4 Dialogue and Argumentation Framework Controller

This component handles communication between the core modules and other modules in the overall Council of Coaches system, with two exceptions (the Coaching Engine and Shared Knowledge Base, as explained in Section 4.3). The purpose of the controller is to direct messages received on the Council of Coaches message broker to the appropriate module(s) in the DAF and vice versa. A configuration file maps topics from the COUCH message broker to internal DAF topics.

This design allows for internal changes to the DAF without needing to alter the format or destination of the messages received from other COUCH components. For instance, if a new module were developed that was required to intercept all messages that were received, a simple change to the configuration file is all that is required, rather than modifying external (to the DAF) components to send their messages to a different destination.

4.5 Internal Message Broker

The internal message broker facilitates communication between the different modules. Using a message-oriented architecture, instead of, for instance, service-oriented, allows for a simpler deployment of new modules that send and receive content to and from existing modules, and reflects the overall Council of Coaches architecture.

[3] https://www.mongodb.com

5 Bibliography

Snaith, M., & Reed, C. (2012). TOAST: Online ASPIC+ implementation. In B. Verheij, S. Szeider, & S. Woltran (Ed.), *Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA 2012)* (pp. 509-510). Vienna, Austria: IOS Press.

Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation, 1*(2), 93-124.

Bex, F., Lawrence, J., & Reed, C. (2014). Generalising argument dialogue with the Dialogue Game Execution Platform. In S. Parsons, N. Oren, & C. Reed (Ed.), *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA 2014).* Pitlochry, Scotland: IOS Press.

op den Akker, H., Beinema, T., Hofs, D., & van Schooten, B. (2019). *D3.5: Shared Knowledge Base component.* The Council of Coaches Consortium.

Beinema, T., & op den Akker, H. (2018). *D3.3: Definition of tailored coaching strategies.* The Council of Coaches Consortium.

Acknowledgements

The Council of Coaches project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Headings and titles in this document, as well as the Council of Coaches logo use the Comfortaa font, designed by Johan Aakerlund and Cyreal and licensed under the Open Font License[4].

Additional text in this document uses the Roboto font, designed by Christian Robertson and licensed under the Apache License, Version 2.0[5].

The Council of Coaches logo and Blobmen graphics were *drawn freely* in Inkscape, licensed under the GNU General Public License[6].

[4] Open Font License: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL_web
[5] Apache License, Version 2.0: http://www.apache.org/licenses/LICENSE-2.0
[6] Inkscape License Information: https://inkscape.org/about/license/