Cognitive Intervention Asset

TUGraz – T3.4F – Apache 2.0 – Client side C#

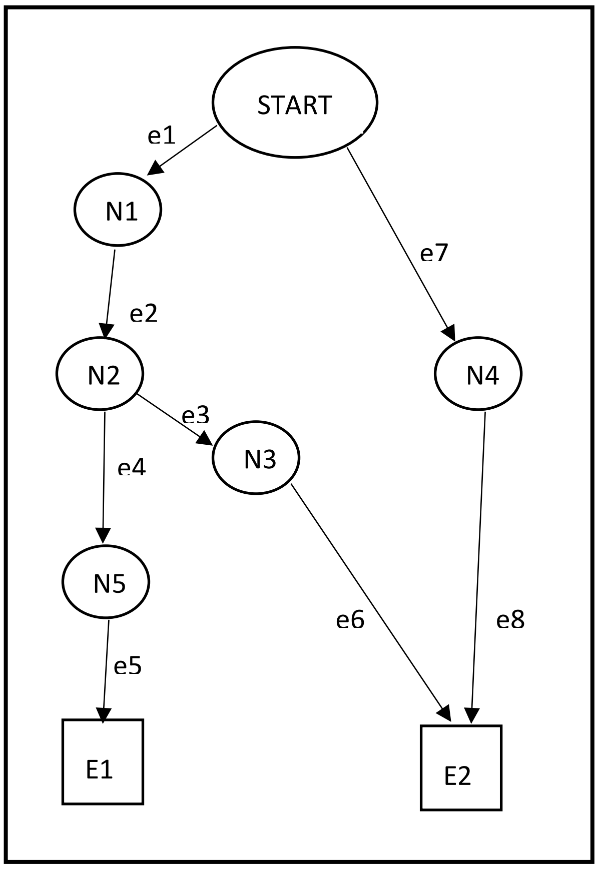
About this asset

This asset is implemented as a client-side component in C# and is used to support the actions, reflections, and cognition of the learner. It uses game events (log data of the interactions of the player with the game) and provides hints and suggestions to the player to support. The asset should be instantiated only once by the game, the asset architecture prevents multiple asset creation..

Asset Mechanics

This Asset uses traces of the player's interactions with the game received from the game to detect anticipated and wrong behaviour. An authoring tool is used to create a model of wanted and unwanted behaviour, which is represented in XML format and is used as data-input for the asset. Also the reaction in terms of hints and suggestions on the observed behaviour is modelled within this authoring tool. The hints and suggestions constitute the interventions and consist of an intervention type and intervention instance. Those two pieces of information (both of which are represented as text strings) are supplied to the game, which is in charge of performing the specific intervention.

A tree approach is used to model wanted and unwanted behaviour (an example depicted in the figure). The nodes represent current states and the edges represent game events or interactions of the player in the game. Thereby beginning with an always-activated starting node, nodes are activated, if a received event activates an edge leading from an activated node to them. A node is deactivated, when it was used to activate another node. Each node contains information about the activation duration (time until the node is deactivated again) and triggered intervention type (correspond to the set of intervention instances triggered, if necessary).



Modelling wanted behaviour:

If an active node is deactivated, because the activation duration is over, it is assumed, that the player did not perform the next move to achieve game goals and the intervention type associated with the node is triggered. For example, if it is wanted that the player performs the activities e1, e2, e4, and e5, than the nodes N1, N2, and N5 are assigned with an activation duration. If the player reaches E1, everything is fine, an according event can be triggered as defined for E1. If the player, however, stops at N5, than the event is triggered that is defined for N5.

Modelling unwanted errors:

If a series of tracks is leading to an end-node (node with no connection to other nodes), it is assumed, that the player performed moves leading to an error and the intervention type associated with this node is triggered. In this case end-nodes are assigned with interventions (but without a time/duration information). The assigned intervention is triggered if the player reaches this node.

Modelling Interventions

Interventions in this setting are represented as intervention types, each of which consist out of one or more intervention instances. Both, intervention types and instances are modelled as text. In case an intervention is triggered, a Method is called, taking two input texts (intervention type and instance) and performing a desired action. This action is specified via the asset's interface. To give an example: In case the 'Player perform very poorly' intervention type and 'Go on – you can do better!' intervention instance is triggered one can display the intervention instance to the player and slow down the game speed. What is done with the supplied information is totally up to the game developer.

Cognitive Intervention Types

Introduction

In the Cognitive Intervention Asset different types of interventions are defined and used, based on their intended purpose. The asset comes with a set of predefined, pedagogically meaningful types of interventions (adapted from Kickmeier-Rust, Steiner, & Albert, 2011) that can be used with, adapted to, and extended for an individual game.

Cognitive interventions**,** in general, aim at supporting learning and enhancing knowledge acquisition. They comprise metacognitive interventions and knowledge interventions; from each of these main categories different types of interventions can be distinguished. All together form the menu of adaptation covered by the cognitive intervention asset.

Metacognitive Interventions

All meta-cognitive interventions address meta-cognition, i.e. they aim at supporting the player in or prompt the player to think about his/her own learning and thinking processes and on their knowledge. The different types of meta-cognitive interventions are aligned with the phases of the self-regulated learning process (e.g. Zimmerman, 2002).

This type of interventions is quite generic; the same intervention instances may be used across different kinds of games.

- **Metacognitive intervention - planning**
    - These interventions try to promote or support forethought on learning, i.e. planning and preparation for a problem solving process, learning task, game mission.
    - Examples of intervention instances:
        - What do you think are feasible strategies for tackling this task?
        - Think about how best to start working on this.
        - First of all take a moment to think about your goal.
- **Metacognitive intervention - performance**
    - This intervention type is provided in the performance phase, i.e. during the learning or problem solving process, in order to prompt self-observation and -monitoring

  - Examples of intervention instances:
    - Try to keep track of what you are doing.
    - Monitor your own progress.
    - In each step of your solution approach keep in mind the overarching goal that you pursue.
- **Metacognitive intervention - success-reflection**
  - This intervention type aims at fostering reflection on successful problem solving performance and/or learning progress after working on a task or after a game situation.
  - Examples of intervention instances:
    - Think about what you have learned in this task.
    - Don't you think that mastering this can help you to tackle also future missions.
    - You have made good progress. Hold on to think about how you reached this goal.
- **Meta-cognitive intervention - failure-reflection**
  - This intervention type aims at fostering reflection on erroneous or unsuccessful problem solving performance and/or learning progress after working on a task or after a game situation.
  - Examples of intervention instances:
    - Think about how you could do better next time.
    - Reflect on what went badly. What would be an alternative solution approach to tackle this problem?
    - How can you learn from this unsuccessful attempt?

Knowledge Interventions

In certain situations it may not be sufficient to prompt players to think about their own cognitive processes (i.e. through metacognitive interventions). Knowledge interventions directly target domain knowledge and knowlege acquisition. They are dependent on the individual game and its targeted learning objective and subject matter.

This type of interventions is specific to the knowledge domain and learning goals of an individual game; intervention instances need to be defined for an individual game.

- **Knowledge activation intervention**
  - This intervention type aims at activating or reminding the player of knowledge that he/she (assumingly) has already available (e.g. because of previous performance), but is not demonstrated by the player in the current interaction/task
  - Examples of intervention instances:
    - Earlier we've come across some things that should not be included in a CV. Try to remember.
    - Try to remember the major components of a business plan, as discussed earlier.
    - Remember what are suitable and less suitable opening questions in an interview.
- **Knowledge acquisition intervention**
  - This type of intervention explicitly conveys knowledge. It will be used, for example, if it seems that a player gets stuck in a problem solving process (because he/she lacks

knowledge required for successful mastering it) and needs further support to accomplish the task.

- Examples of intervention instances:
    - You always need to include contact information in your CV.
    - A successful business plan includes an analysis of the competitive landscape of your business.
    - Make sure to close the interview on a positive note, for example expressing your enthusiasm for the job opportunity and summing up your suitability for the job.

References

Kickmeier-Rust, M.D., Steiner, C.M., & Albert, D. (2011). Apt to adapt: Micro- and macro-level adaptation in educational games. In T. Daradoumis, S. Caballé, A.A. Juan & F. Xhafa (Eds.), Technology-enhanced systems and tools for collaborative learning scaffolding. Studies in Computational Intelligence Vol. 350 (pp. 221-238). Berlin: Springer.

Zimmerman, B.J. (2002). Becoming a self-regulated learner: An overview. Theory into Practice, 41, 64-70.

Asset interfaces

- An interface for sending traces to the asset will be implemented (This interface might be omitted and traces might be taken from the tracker.). A C# representation in the asset interface could be:

  ```
  void sendTrace(String trace)
  ```

- An interface to check for interventions due to wanted behaviour. This is needed because it is checked for interventions only when receiving a trace. That especially means, that nodes get deactivated only when receiving a trace:

  ```
  void refresh()
  ```

- A method for handling interventions need to be specified (the asset gets a method of the game and sends interventions to this method when needed) – it can be set via the following interface call:

  ```
  void setInterventionDelegate(CognitiveInterventionDelegate del),
  ```

  whereas CognitiveInterventionDelegate is an interface for a method taking two input strings, the intervention type and the intervention instance.

Asset dependencies/requirements

- The Game Storage is used as local storage.
- (It may collaborate with the Competence Assessment Asset)

Milestones

Milestone 1

- t1.1: Creating the first version of the design document, defining the API and creating a dummy asset with the API implemented

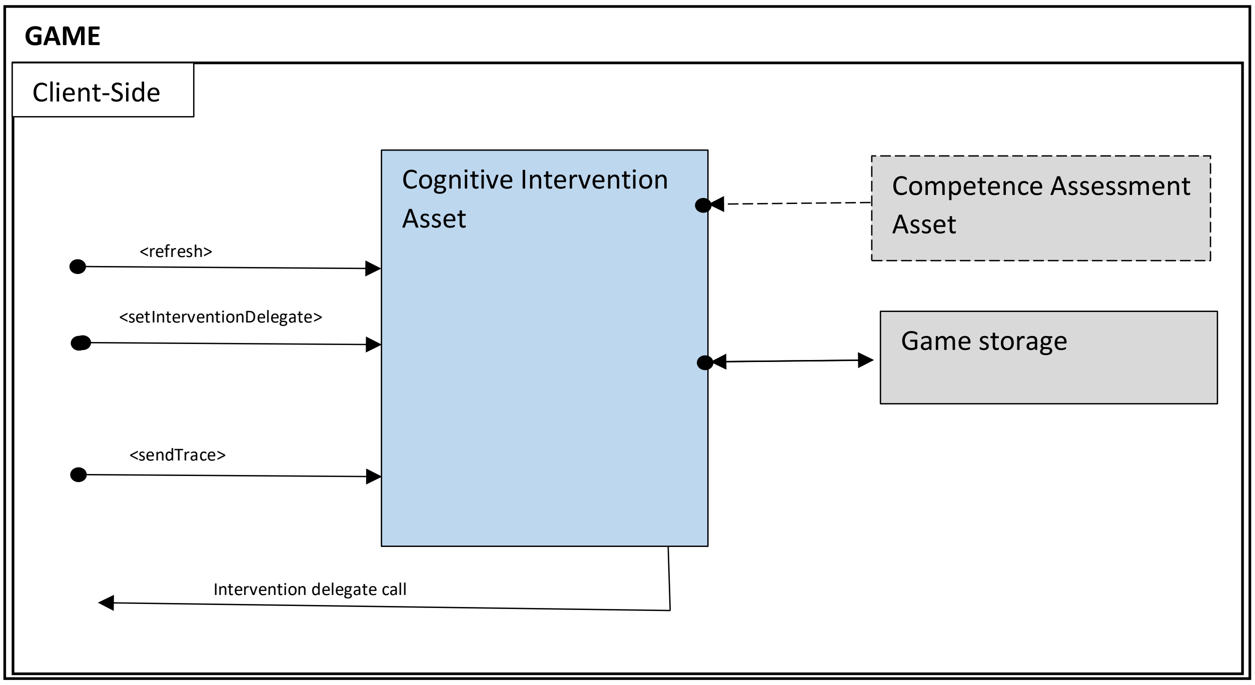
Milestone 2

- t2.1: Create Software Asset in line with Asset-Manager infrastructure.
- t2.2: Integrate an interface for the Competence Assessment Asset.
- t2.3: Elaborate identifier-dictionary for the Asset interface.

Milestone 3

- t3.1: testing the asset with a game
- t3.2: instructions and scripts for building and deployin

Graphical representation


GAME
Client-Side
Cognitive Intervention Asset
Competence Assessment Asset
Game storage
<refresh>
<setInterventionDelegate>
<sendTrace>
Intervention delegate call

←	… Flow of information
●	… Communication start
<*>	… Interface's exposed method
🟦	… Asset component

Set up the Asset

For the cognitive intervention Asset, there are two things to do (additionally to creating the Asset) when setting it up:

- Define what should happen in case an intervention is triggered. Therefore a method has to be defined performing the desired action when receiving a text-string (the intervention instance) and delivered to the Asset. For this purpose, a delegate created and set via the Asset. The C# code could look like the following, when prompting the intervention out to the console:

```
CognitiveInterventionAsset cia = CognitiveInterventionAsset.Instance;
CognitiveInterventionDelegate ciD = intervention => Console.WriteLine(intervention);
cia.setInterventionDelegate(cognitiveInterventionDelegate);
```

- Supply the tracking description. The Asset is working with traces received from the game and these traces need to be interpreted. These interpretation rules and the intervention instances need to be supplied in a XML-file, created via a JS-HTML page (more information where to find and how to use this tool will be supplied at a later stage). This XML need to be available with a given file-identification, which is stored in the Asset-settings (as property XMLFileLocation). To load this File, also an IDataStorage-Bridge needs to be implemented.

Use the Asset

For using this Asset tracks need to be submitted to the Asset. At the moment this is only possible by sending the traces via the Asset interface, using for example the following C# code:

```
cia.sendTrace("pick up mouse");
```

One can check for an intervention, using for example the following C# code:

```
cia.refresh();
```

Later on, it is planned to omit this function call by requesting all tracks via the UCM tracker – then all traces are only send to the tracker.

Deployment

For the source code the following GitHub-link can be used https://github.com/RAGE-TUGraz/CognitiveInterventionAsset - it contains the Visual Studio solution of this asset's C# implementation. Furthermore the broken links to external asset DLLs need to be fixed for each project and the Bridge code need to be adopted to the new environment, e.g. changing the IDataStorage path.

For integration into Unity, the resulting DLLs need to put into a folder in the Unity working-directory.

Unit test

For executing unit tests, the source code need to be open in visual studio and all links need to be fixed. In the test-explorer all tests can be executed.