**Reactive Planning**

Behavior Oriented Design (BOD) (Bryson, 2001) takes inspiration both from the well-established programming paradigm object-oriented design (OOD) and Behaviour base AI (BBAI), to provide a concrete architecture for developing complete, complex agents, with multiple conflicting goals and mutual-exclusive means of achieving those goals.

POSH planning is the action selection for reactive planning derivative of BOD. POSH combines faster response times similar to reactive approaches for BBAI with goal-directed plans (Bryson, 2008). A POSH plan consists of the following plan elements:

- **Drive Collection (DC):** The root node of the plan. It contains a set of Drives and is responsible for giving attention to the highest priority Drive. To allow the agent to shift and focus attention, only one Drive can be active in any given cycle.

- **Drive (D):** Allows for the design and pursuit of a specific behaviour as it maintains its execution state. The releaser, is a precondition using sensory in to determine if the drive should be pursued. The drive execution frequency limits the rate at which the Drive can be executed.

- **Competence (C):** A self contained basic reactive plan (BRP), each containing one or more Competence Elements (CE), each of which has a priority and a releaser.

- **Action Pattern (AP):** Used to reduce the computational complexity of search within the plan space and to allow a coordinated fixed sequential execution of a set of Actions.

The Instinct Planner is a reactive planner based on the POSH planner. It includes several enhancements taken from more recent papers extending POSH (Gaudl and Bryson, 2014). In an instinct plan, the AP contains one or more Action Pattern Elements (APE), each of which has a priority, and links to a specific Action, Competence, or another AP.

**Future Work**

Further debugging features: We plan to continue developing this new editor, implementing debug functions such as “fast-forward” in pre-recorded log files and usage of breakpoints in real-time.

Use with games AI: We plan to develop a new planner, for the game DEFCON, to be used alongside the editor. We will test the hypothesis that having access to transparency information can you help you cooperate better with an intelligent agent.

Empirical study on how it helps developers: A beta version of the editor will be released to inexperienced AI developers. We plan to gather feedback on how ABOD3 and its debugging capabilities helped them understand, develop, and tune intelligent agents.

**References**


**The Editor**

Real-time debugging: Designed to allow not only the development of reactive plans, but also to debug such plans in real time to reduce the time required to develop an agent. This allows the development and testing of plans from a same application.

Log file support: Allow the usage of log files, while simulating their execution in real time. A media player is included and is synchronised with the log, allowing you a reconstruction of the run.

Debugging of the RS, using captured log and video files.

System Architecture Diagram of ABOD3, showing its modular, expandable design. All of ABOD3 was written in Java8 and JavaFX.

**Use Cases**

ABOD3 was used in two experiments to determine the effects of transparency on the mental models formed by humans.

Subjects can show marked improvement in the accuracy of their mental model of a robot observed, if they also see an accompanying display of the robot’s real-time decision making as provided by ABOD3.

Providing transparency information by using ABOD3 does help users to understand the behaviour of the robot, calibrating their expectations.

**References**

**Authors**

Andreas Thedorou
PhD Student University of Bath

Dr Joanna Bryson
Reader, University of Bath
Fellow, Princeton’s Centre of Information Technology Policy