

UNIFIED INTERFACE IMPLEMENTATION IN A 4GL FRAMEWORK: THE I-GET UIMS TOOL

The I-GET tool is a language-based UIMS for dialogue control implementation. It requires that interface developers write explicitly interface code in the I-GET dialogue specification language. Interactive construction facilities are not provided, since this would require a specific toolkit to be considered, an approach conflicting with one of the main objectives of the I-GET tool: openness with respect to underlying utilised toolkits. I-GET supports integration and utilisation of multiple toolkits in parallel through its powerful toolkit interfacing method. The run-time architecture of interfaces developed using the I-GET tool is illustrated in Figure 1.

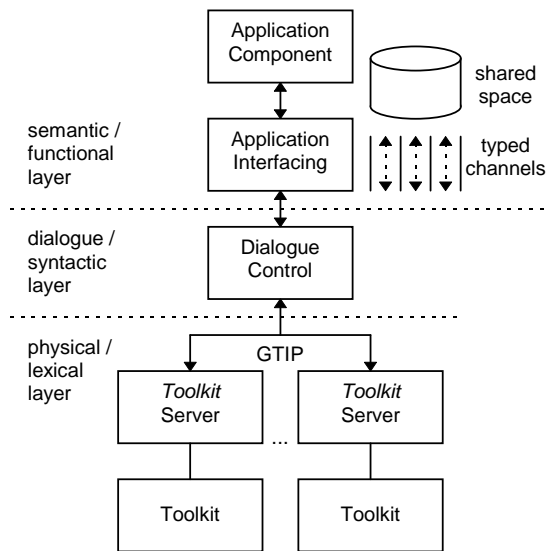


Figure 1. Run-time architecture of I-GET interfaces

Toolkit Server

A toolkit plays the role of an intermediate translator between the Dialogue Control components' "view" of the particular toolkit and the original / native toolkit programming interface. Implementationally, the "real" physical interaction elements will be managed and locally maintained at the toolkit-server side, while the Dialogue Control components' view consists of lightweight software elements, which gain a physical substance through the toolkit servers' translation "filter".

Toolkit Interfacing

The toolkit interface mechanism of the I-GET tool is not realized via a separate running process. Instead, it is based on a special purpose semantic protocol, called *Generic Toolkit Interfacing Protocol* (GTIP). This protocol has been designed in order to enable physical separation between the programming interface of a toolkit (i.e. object classes, data types and procedures) from the real implementation of physical interaction elements (i.e. rendering, device handling, display algorithms).

Dialogue Control

The implementation of the Dialogue Control component is realized through the I-GET language (we also use the phrase "Dialogue Control specification" to denote this type of implementation). The Dialogue Control specification is translated by the I-GET compiler into a C++ implementation, which can be compiled and linked (with the run-time library) into an executable file by using a C++ compiler.

Application Interfacing

The Application Interfacing component software is provided with the I-GET public release. The role of this module is to coordinate and mediate communication between the Application Component and the Dialogue Control component; in this context, it maintains the space of shared objects and controls "flow" of messages through the message channels.

Application Component

The Application Component collects together all the various modules supplying non-interactive functionality. The I-GET tool provides a "template" structure requiring implementation of a small set of special-purpose C++ functions, which are the minimal requirements for making an Application Component for the I-GET run-time architecture. An Application Component never communicates directly with the Dialogue Control component, but it may either "access" the shared space or post messages in a message channel. In the same manner, an Application Component is implemented by utilising I-GET library functions for receiving notifications about events occurring in the communication space (which is maintained by the Application Interfacing process).

The I-GET language consists of four key logical layers of constructs illustrated in Figure 2: (i) **API layer**, for the specification of the application interfacing space; (ii) **Agent layer**, which concerns the specification of agent classes; agent classes play the role of dialogue control component classes; (iii) **Objects layer**, which is related to the levels of interaction objects supported in the I-GET language; physical objects and virtual objects are both supported; and (iv) **Common layer**,

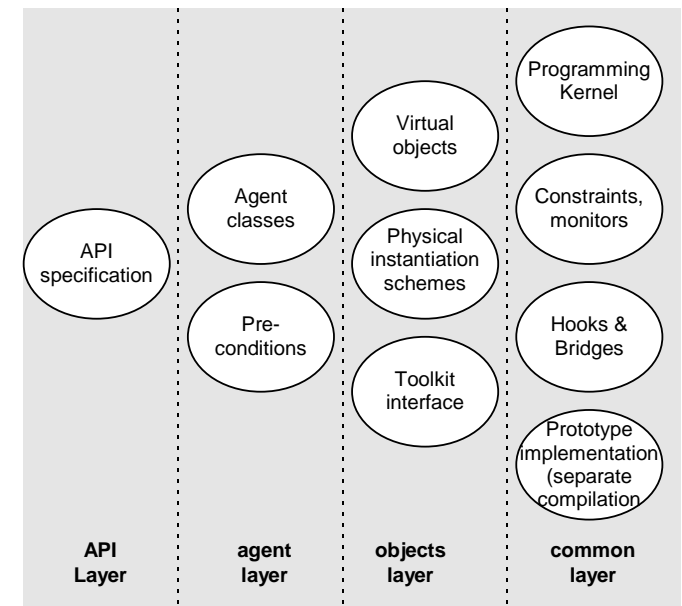


Figure 2. Layers of the I-GET language

which provides language constructs which can be employed in any of the above three layers: (a) programming kernel; (b) constraints and monitors; (c) hooks and bridges; and (d) prototype / implementation facility, which support separate compilation.

The I-GET tool supports different *development tasks* which are logically associated to the following implementation layers (see Figure 3):

- *Functional Layer*, which concerns the development of the Application Component and the design & specification of the Application Interfacing space.
- *Dialogue Layer*, which concerns the design & implementation of the User Interface.
- *Physical Layer*, which concerns the integration of toolkits of interaction elements, a task concerning the toolkit interface specification, as well as the toolkit server development.

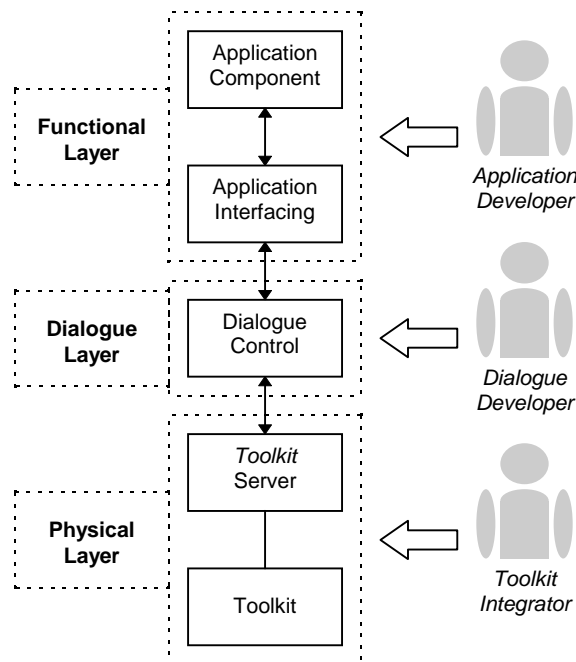
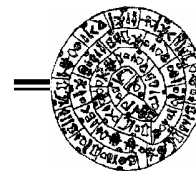


Figure 3. Development roles in the I-GET tool



FOUNDATION FOR RESEARCH AND TECHNOLOGY - HELLAS

INSTITUTE OF COMPUTER SCIENCE
Science and Technology Park of Crete
Heraklion, Crete, GREECE

Human-Computer Interaction Laboratory &
Centre for Universal Access and Assistive Technologies

THE ACCESS CONSORTIUM

Part of this R&D work has been carried out in the context of the TIDE TP1001 ACCESS project "Development platform for unified ACCESS to enabling environments", partially funded by the European Commission (DG XIII). The ACCESS consortium comprises: CNR-IROE, Italy (Prime Contractor); ICS-FORTH, Greece; University of Athens, Greece; RNIB, UK; SELECO, Italy; MA systems Ltd., UK; Hereward College, UK; STAKES, Finland; VTT, Finland; PIKO Systems, Finland; University of Hertfordshire, UK.

CONTACT INFORMATION

Dr. Anthony Savidis

Foundation for Research and
Technology-Hellas (FORTH)
Institute of Computer Science (ICS)
Human - Computer Interaction Laboratory
Heraklion, Crete, GR-70013 GREECE

Tel.: + 30 - 2810 - 391749
Fax : +30 - 2810 - 391740
E-mail: as@ics.forth.gr
WWW: <http://www.ics.forth.gr/hci/>

The I-GET UIMS Tool

<http://www.ics.forth.gr/hci>