AGI engineering Subscribe

AGI INFRASTRUCTURE

TECHNOLOGICAL ASPECTS

Mykola Rabchevskiy
May 24

Any AGI system is inherently complex; therefore, technological aspects largely determine the degree of development success. Technological aspects include:

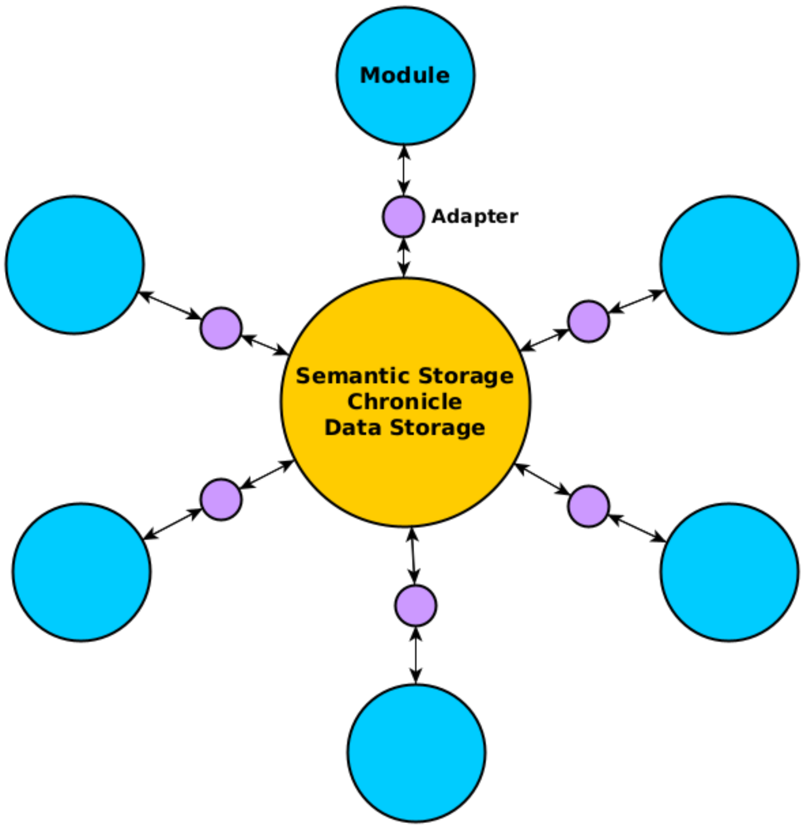
- a way of combining many modules into a single system
- human-machine interface
- organization of system testing
- choice of programming language

Without pretending to be universal suitability for our infrastructure option, we list the considerations in favor of the choice made.

METHOD FOR COMBINING MODULES INTO A SYSTEM

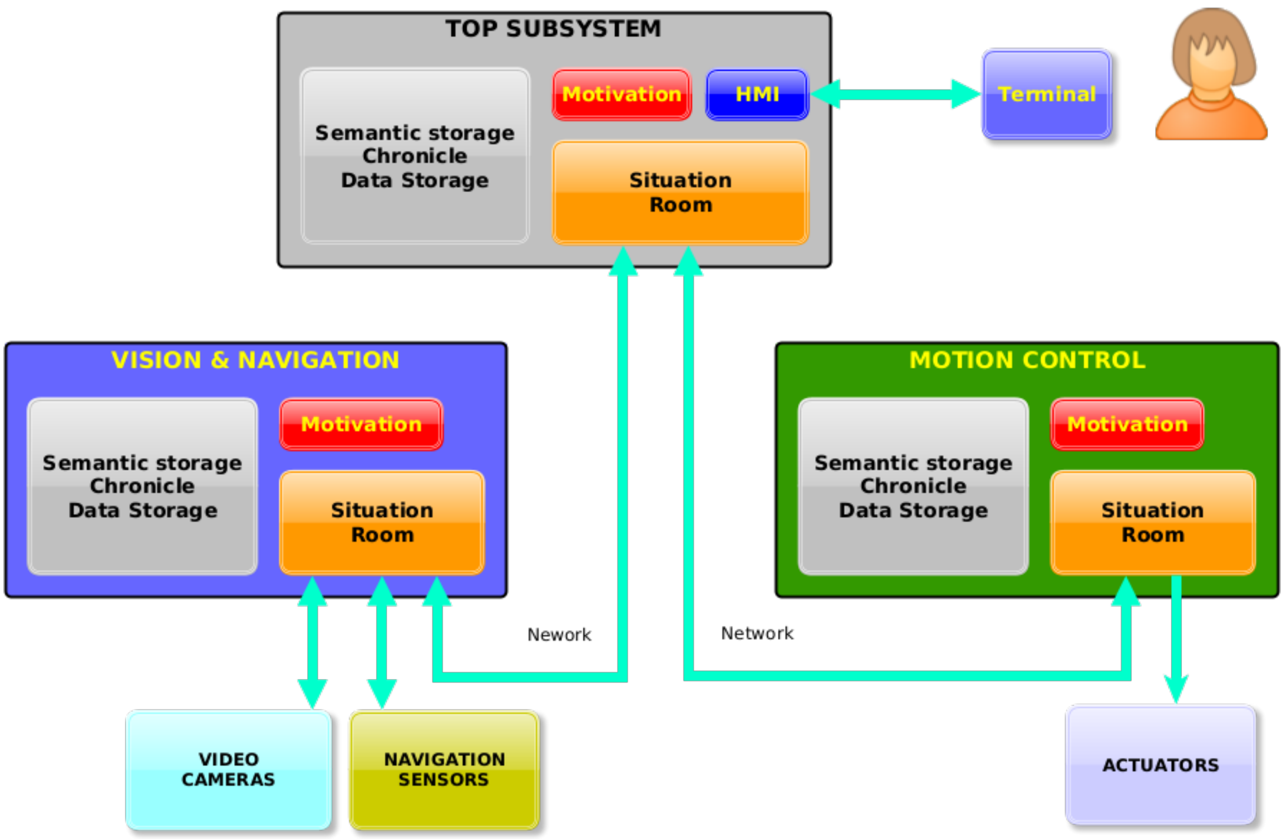
To ensure acceptable performance, AGI system modules must work simultaneously and asynchronously, exchanging data. This can be organized into channels for transferring data (messages) between them so that the data exchange structure is a general communication graph. This approach is used, in particular, in ***ROS*** (***Robot Operating***

Semantic Storage, ***Chronicle***, and ***Data Storage***) plays the role of an information hub, which is not only storage of long-term data but also serves as a transmission link in the operational exchange of data between modules. Each module has bi-directional access to "Storage" (using the API), so a module never requires more than one converter (adapter), regardless of the number of modules, and the number of data copies is minimized. ***Semantic Storage***, ***Chronicle***, and ***Data Storage*** are separate modules, logically connected and use a single data representation, form a kind of "backbone" of the AGI system. Accordingly, we use a "star" topology:



In the diagrams devoted to the system architecture, the arrows link data producers with data consumers, reflecting the logical connections between the system modules; the above diagram demonstrates the data exchange technology. Thus, these schemes do not contradict each other but reflect the difference between logical connections and technological implementation.

Using a multi-level architecture, it is presumed to use a hierarchical structure in which each unit/node is a separate computer (microcontroller), the exchange of data between which is provided by the network (***Ethernet***, ***CAN bus***, and so on). Each unit potentially has the entire set of components that a single-level system can have. Still, the parameters (amount of memory, the actual amount of information, performance, etc.) are selected according to the specifics of the corresponding node. The ability of all units to work autonomously in the absence of directives from the parent unit, informing it about events as needed, is essential. Below is a possible configuration for a robot:



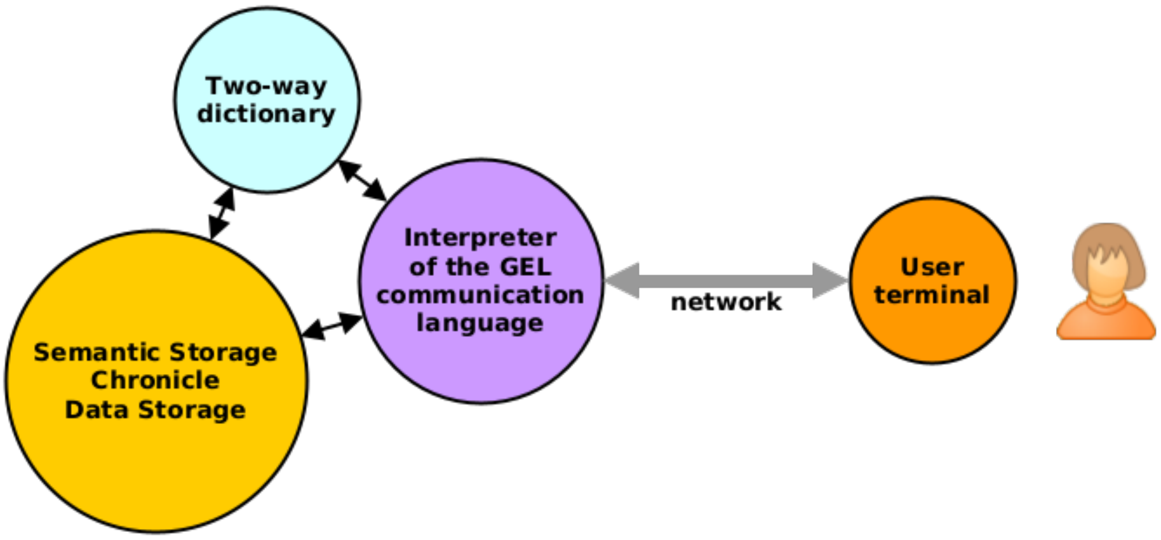
HUMAN-MACHINE INTERFACE

A necessary element of AGI is a component that provides access to the internal information of the system. It allows you to quickly observe the operation of the system, analyze the reasons for successful or undesirable actions, correct the behavior of the system (thereby maintaining explainability and correctability), extract the knowledge accumulated by the system (including the analysis of the concepts formed by the system) and replenish the system with knowledge from other sources. The connecting elements between the internal data representation and the human-friendly format are a two-way dictionary and an interpreter of a formalized data exchange language. Dictionaries associate internal IDs with their names, and the interpreter transforms the text in the data exchange language into an internal representation.

The design of a data exchange language is based on a trade-off between human usability and the complexity of an interpreter. The use of natural language seems to be too difficult, not only due to the many known problems of natural language processing but also because there is no uniform approach to processing different natural languages. On the other hand, well-known formal languages used in computer technology differ significantly from natural languages, creating difficulties for users. This includes overly rigid grammar and excessive verbosity; a large set of reserved words is also inconvenient, which prevents a reasonable choice of terms for the AGI system's concepts.

Our trade-off is the language ***Gel***, which, being quite formal, is intermediate between very fluid natural language and rigid languages (hence the name ***Gel***) used in AI (***RDF***, ***OWL***, ***Atomese***, and so on). A detailed description is the subject of a separate chapter, but just note that Gel has no reserved words, and the statements bear an obvious resemblance to natural language phrases. The concepts are named as it is in a natural language (changing which is reduced to changing the vocabulary, which makes it possible to use several languages simultaneously).

The HMI uses a network to communicate between AGI and the user/operator terminal. The terminal can be either purely textual or has a graphic area for displaying visual information if the AGI system has a computer vision subsystem.


Two-way dictionary
Interpreter of the GEL communication language
Semantic Storage Chronicle Data Storage
network
User terminal

TESTING

Like any technical system, AGI requires testing, and the specificity of AGI determines the specifics of testing and test design. As an intelligent system, AGI requires the involvement of the intellect in the testing process: the active participation of a person must determine the limits of the attainable characteristics in terms of the intelligence of behavior and the ability to self-learn.

AGI system requirements imply the possibility of use in various missions, which entails developing several test environments to confirm this.

Requiring AGIs to respond intelligently to new unknown situations requires a high degree of variability in test environments, allowing the tester to create these unexpected situations.

Significant differences between discrete and continuous environments of potential use require test environments of the appropriate type.

One of the most challenging aspects is that the ability of a particular system to scale in practice cannot be estimated theoretically and the ability to maintain performance when changing a virtual environment to a real one. For example, if a system is capable of functioning in the presence of one dynamic object with which it interacts, this does not mean that it is capable of behaving appropriately in the presence of several such objects; if the vision system works reasonably in a virtual environment, this does not mean that it will be acceptable when using real video cameras, and so on. Therefore, the choice of test environments requires special attention; in many cases, a natural test environment (test site) may turn out to be a cheaper and simpler option than a high-level virtual environment:
https://www.youtube.com/watch?v=bhcSb6hPEJo

PROGRAMMING LANGUAGE

A large amount of computation and a large amount of data to be processed require maximizing the efficiency of the AGI code. Our choice, C++, fully meets this requirement. Besides, C ++ has several equally essential advantages. The low-level memory access capability can significantly (several times) improve performance compared to Java. The set of libraries - both standard and others (for linear algebra, geometry, image analysis, etc.) are unprecedentedly large. Finally, the latest versions of the C ++ standard reduce the possibility of errors while adhering to the appropriate coding norms and significantly reduce the size of the code compared to C and Java. It is also vital that there is a sufficient number of programmers with relevant experience.

← Previous Next →