# Table of Contents

<table>
<thead>
<tr>
<th>APICe Products</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alchemist</td>
<td>3</td>
</tr>
<tr>
<td>ALOO</td>
<td>3</td>
</tr>
<tr>
<td>CArtAgO</td>
<td>3</td>
</tr>
<tr>
<td>Home Manager</td>
<td>3</td>
</tr>
<tr>
<td>Molecules of Knowledge</td>
<td>4</td>
</tr>
<tr>
<td>MS-BioNET</td>
<td>4</td>
</tr>
<tr>
<td>P@J</td>
<td>4</td>
</tr>
<tr>
<td>ReSpecT</td>
<td>5</td>
</tr>
<tr>
<td>simpA</td>
<td>5</td>
</tr>
<tr>
<td>simpAL</td>
<td>5</td>
</tr>
<tr>
<td>SODA</td>
<td>6</td>
</tr>
<tr>
<td>TraumaTracker</td>
<td>6</td>
</tr>
<tr>
<td>TuCSoN</td>
<td>6</td>
</tr>
<tr>
<td>tuProlog</td>
<td>7</td>
</tr>
</tbody>
</table>
APICe Products

Research activities in the APICe lab often result in products that can be shared and freely used, consisting mainly of software technologies — programming languages and frameworks, software infrastructures — but also including methodologies for the engineering of complex computational systems.

Below, the current list of products actively supported — mainly by the members of the aliCE research group — along with contact details and the corresponding web spaces — whenever available. The list is ordered alphabetically.

Alchemist

Alchemist is a simulation framework offering both high performance and flexibility. Alchemist tries to get the best from ABMs and stochastic simulators and build an hybrid.

Space: Alchemist Home
Contact: Danilo Pianini

ALOO

ALOO is a general-purpose concurrent programming language based on objects and agents as first-class abstractions. It is a Concurrent OOP language where concurrency aspects (parallelisms, asynchronous programming, etc) are modeled in terms of agent-oriented abstractions. Or, it can be conceived as an Agent-Oriented Programming language not based on logic programming - like state-of-the-art AOP languages - but on OOP. The language comes with a platform (compiler, runtime) and an Eclipse-based IDE. ALOO is the most recent evolution of the simpAL project.

Space: ALOO Home
Contact: Alessandro Ricci

CArtAgO

CArtAgO is an infrastructure for the development and execution of artifact-based computational environments for Multi-Agent Systems.

CArtAgO is based on the A&A (Agents and Artifacts) meta-model model, which introduces the artifact abstraction as a first-class building block to model and engineer those objects, resources, tools, that can be dynamically constructed, shared and co-used by agents to perform their activities, both individual and social.

Space: CArtAgO Home
Contact: Alessandro Ricci

Home Manager
Home Manager is a prototype agent-based platform for the construction of Socio-Technical Smart Spaces and Smart Environments, starting from the control of an intelligent home to a wider Smart Living scenario, where home users are situated in space and time in the surrounding environment. The house is seen as an intelligent environment made of independent devices that participate to an agent society: following the Butlers vision and architecture, Home Manager aims both to enable a rich interaction experience for the user in the interaction with the smart house, and to anticipate his/her needs based both on the user's habits and policies, while respecting the general system policies which capture the global goals and constraints. The implementation is Java-based, developed on top of the TuCSoN agent infrastructure, and deployable also on low-cost devices such as Raspberry Pi2.

Space: Home Manager Home  
Contact: Roberta Calegari

**Molecules of Knowledge**

Molecules of Knowledge (MoK for short) is a model for knowledge self-organisation, exploiting the biochemical metaphor for its basic abstractions, and biochemical coordination as its coordination model.

Space: MoK Home  
Contact: Stefano Mariani

**MS-BioNET**

MS-BioNet (MultiScale-Biochemical NETwork) is a computational framework for modelling and simulating large networks of compartments hosting a chemical solution and communicating through an enhanced model of chemical reaction addressing molecule transfer.

Space: MS-BioNET Home  
Contact: Sara Montagna

**P@J**

P@J is a framework for enhancing interoperability between Java and Prolog, based on the tuProlog open-source Prolog engine for Java. P@J achieves smoother language interoperability through two stacked layers:

- an API layer for automated mapping of Java types into Prolog types (and viceversa) and seamless exploitation of the Generic Collections Framework;
- an annotation layer, that aims at truly extending Java programming with the ability of specifying Prolog-based declarative implementations of Java methods, relying on Java annotations.

Space: P@J  
Contact: Mirko Viroli
ReSpecT

ReSpecT is a logic-based coordination language both to define the behaviour of tuple centres and to coordinate software agents. ReSpecT assumes a tuple-based coordination model, where communication is based on logic tuples, and the behaviour of the coordination media in response to communication events is not fixed once and for all by the coordination model, but can be defined and tailored according to system's global requirements. According to this conceptual framework, ReSpecT has a twofold role:

• as a specification language - ReSpecT makes it possible to filter communication events and associate them to reactions by means of first-order logic tuples.
• as a reaction language - ReSpecT supports the notion of reaction as a computational activity to be executed locally to a tuple centre in response to communication events, by providing for both a logic-based syntax for the definition of reactions, and a model for their execution.

Space: ReSpecT Home
Contact: Stefano Mariani

simpA

simpA is a framework extending the basic Java environment with an agent-oriented abstraction layer for programming complex (concurrent in particular) applications.
simpA is based on the A&A (Agents and Artifacts) programming model, which introduces high-level metaphors taken from human society, namely agents - analogous to humans, as executors of activities and activities — and artifacts — analogous of the objects, resources, tools that are dynamically constructed, used, manipulated by humans to support / realise their individual and social activities.

Space: simpA Home
Contact: Alessandro Ricci

simpAL

simpAL is a general-purpose programming language and platform for developing concurrent and distributed applications.

simpAL is based on the A&A (Agents and Artifacts) programming model, which introduces high-level metaphors taken from human society, namely agents — analogous to humans, as executors of activities and activities — and artifacts — analogous of the objects, resources, tools that are dynamically constructed, used, manipulated by humans to support / realise their individual and social activities.

Differently from simpA, simpAL is not a Java extension, but a fully independent language and platform.

Space: simpAL Home
Contact: Alessandro Ricci
SODA

SODA (Societies in Open and Distributed Agent spaces) is a methodology for the analysis and design of complex agent-based systems. SODA is not concerned with *intra-agent issues*: designing a multi-agent system with SODA leads to defining agents in terms of their required observable behaviour and their role in the multi-agent system. Instead, SODA concentrated on *inter-agent issues*, like the engineering of societies and infrastructures for multi-agent systems. Recently a new and extended version of the methodology has been proposed, which takes into account both the Agents and Aartifacts (A&A) meta-model, and a mechanism to manage the complexity of system description.

**Space:** SODA Home  
**Contact:** Ambra Molesini

TraumaTracker

Developed in collaboration with the Trauma Center and the Emergency Department of the "Bufalini" Hospital in Cesena, Italy, TraumaTracker is a project in which agent technologies are exploited to realise Personal Medical Digital Assistant Agents (PMDA) supporting a Trauma Team in trauma management operations. This project aims at exploring the fruitful integration of software personal agents with wearable/eyewear computing, based on mobile and wearable devices such as smart-glasses.

The key functionality of TraumaTracker is to keep track of relevant events occurring during the management of a trauma, for two basic purposes. The first one is about tracking, i.e. to have an accurate documentation of the trauma, to automate the creation (and management) of reports and to enable offline data analysis. The second one is a first kind of assistance, in terms of the real-time generation of warnings and suggestions to be perceived through the smartglasses, by analysing and reasoning upon the tracked events and data. The system has been designed and developed using cognitive agent technologies based on the Belief-Desire-Intention (BDI) architecture, as supported by the JaCaMo agent platform.

**Space:** TraumaTracker Home  
**Contact:** Angelo Croatti, Sara Montagna, Alessandro Ricci

TuCSoN
TuCSoN (Tuple Centres over the Network) is a model (and related infrastructure and technology) for the coordination of Internet agents.

TuCSoN exploits a notion of local tuple-based interaction space, called tuple centre, which is a tuple space enhanced with the notion of behaviour specification. By programming its behaviour in response to communication events, a tuple centre can embody coordination laws. Several issues critical to Internet applications, such as heterogeneity and dynamicity of the Internet nodes, can then be charged upon tuple centres, and transparently to agents, which can then be designed independently of the different node architectures, according to a straightforward interaction protocol.

The TuCSoN coordination language provides agents with a twofold perception of the TuCSoN interaction space, as either as a global space made up of uniquely denoted coordination media, or a collection of local spaces associated to Internet nodes. This suits both roles of Internet agents, that is, as either network-aware entities or (network-unaware) local agents.

Contact: Stefano Mariani

---

tuProlog

TuProlog is a Java-based light-weight Prolog for Internet applications and infrastructures. For this purpose, tuProlog is designed to feature some interesting qualities: it is easily deployable, just requiring the presence of a Java VM and an invocation upon a single JAR file; its core is both minimal, taking the form of a tiny Java object containing only the most essential properties of a Prolog engine, and configurable, thanks to the loading and unloading of predicates, functors and operators embedded in libraries; the integration between Prolog and Java is as wide, deep, clean as possible; finally, interoperability is developed along the two main lines of Internet standard patterns and coordination models.

Contact: Enrico Denti