In the Proceedings of the IET International Conference on Intelligent Environments, Athens, Greece, July 2006

# AMBIENT INTELLIGENCE – KNOWLEDGE REPRESENTATION, PROCESSING AND DISTRIBUTION IN INTELLIGENT INHABITED ENVIRONMENTS

**J Dooley[1], V Callaghan[1], H Hagras[1],**
**P Bull[2], D Rohlfing[2]**

[1]*Intelligent Inhabited Environments Group, University of Essex, UK.*
*{jpdool, vic, hani}@essex.ac.uk*
[2]*Customer Networks Group, British Telecom, UK.*
*{phil.bull, dave.rohlfing}@bt.com*

Abstract: *Inhabited environments offer a diverse set of problem domains which can* benefit from the application of artificial intelligence (AI). This paper presents a work in progress conducted between British Telecom (BT) and the University of Essex. The work argues that a single knowledge representation and processing model (followed by traditional AI systems) is not adequate to fulfil the requirements of all inhabited environment problem domains (especially with near real-time temporal complexity). Based on this predicate, the work seeks to explore the use of multiple knowledge languages which each address a specific problem domain. 

Keywords:

## 1. INTRODUCTION

### 1.1 The Vision

Physical environments are currently seeing an increase in electronic device deployment and service sharing/delivery. In the future, these environments are set to be populated by a wide range of devices including computationally limited devices (so called embedded devices) which will be interconnected by both wired and wireless communications. The consequence of device deployment, interconnection and communication is an environment that is alive with technology. Electronic entities within these environments will have to interact with each other on an intelligent level if they are to co-exist in a functional, efficient and stable way. These environments, where technology is pervasive (wide spread) and intelligence ambient (occurring natively in the environment surroundings), are deemed "intelligent environments".

Humans have a limited ability to perceive the technology in such environments. Thus "intelligent inhabited environments" (IIEs) will only be of use if the layman can interface in way that feels natural and non-obstructive. The ambient intelligence of such an environment should therefore reduce the required "effort to use" for a user (to a level they personally feel comfortable with, this may vary depending on the individual).

### 1.2 A Technology Facade

"Beauty is in the eyes of the beholder"
-- Unknown origin

A beautified facade (which hides complexity) should exist as an interface between the user and the complex electronic environment (Fig.1). This facade presents the environment to the user in a way that allows them to perceive only things that they should be aware of.


Fig.1. Perception of technology by an inhabitant of an intelligent environment.

1.3 The Ambient Intelligence (AmI)

"Any sufficiently advanced technology is indistinguishable from magic."
-- (Clarke 1999)

The work discussed by this paper refers to enabling the AmI technology that sits “behind the scenes” hidden from the users scope. Enabling AmI requires that firstly it can be driven by Artificial Intelligence (AI), and secondly that intelligence can be ubiquitous across a physical environment.

AI Driving AmI. AI can be used for a multitude of purposes ranging from how resources available to the user are deployed, through intelligent control, to monitoring resource use. However, AI technologies are computationally expensive, and must be harnessed correctly if they are to be made available to embedded systems that need to operate at (or near) real time.

Ubiquitous Intelligence. Intelligent environments are composed of many networked peers. For intelligence to be transposed across the entire network, we must have a way of peer communication which can be used to exchange information. Such information could be social (interaction, negotiation, requests, etc.) or related to the exchange of objects of intelligence (knowledge) between peers.

## 2.THE USE OF AI

IIEs offer an intractably large and diverse set of problem domains (the set M in Fig.3.). This diversity of problem domains requires that the supporting AI must provide suitable representation and processing[1] flexibility. This flexibility allows AI dependants to access an intelligence that is less constrained in the problems it can solve.

### 2.1 Traditional AI

Traditional use of AI will typically take a well defined[2] problem and use a single appropriate method to produce an optimum representation and processing solution. This produces a tightly coupled, “made-to-measure”, problem-solution pair, consequently removing flexibility.

Furthermore, AI is a large umbrella given to a group of technologies that allow computers to perform tasks that normally require human intelligence. The varying technologies and methods that exist under this umbrella are diverse in the problems they tackle, and the way that they work. There is (currently!) no single AI methodology that is appropriate for all AI problems[3].

It has also been documented, (Davis 1993), that in choosing an AI solution, that choice unavoidably introduces certain ontological commitments which accumulate in layers. These commitments concern (among other things) how the world is viewed by such a system and subsequently a systems dependants. Needless to say that this severely compromises flexibility if a dependant wishes to reason in a way that is prohibited by the commitments.

The correlation between layers of ontological commitment and flexibility (for a given AI solution) can be seen in Fig.2. A graph of this form can be calculated (for an AI system which has accumulated a set of layered ontological commitments) using formulae (1).

$$f(x)=\sum_{i=0}^{n} g(L_i)-C \qquad (1)$$

where :

- $L_i$ = Layer i of ontological commitment
- $g(L_i)$ = Flexibility measure of $L_i$
- C = A bias accounting for loss of flexibility in addition to the accumulation of layers.
- n = The number of ontological commitment layers that exist.

1 Software that utilises AI, for example agents.

2 Methods like semantic networks, neural networks, fuzzy logic, game theory, heuristics, etc.

3 A single AI method appropriate for all problems could be seen as a “universal problem solver”. A “holy grail” of AI that some have strived to grasp.

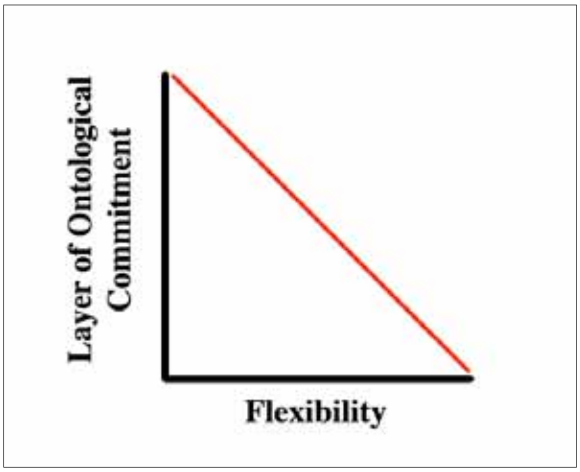


Fig.2. Flexibility increase in relation to less layers of ontological commitment1.

2.2 Flexibility Through Composition

Suitable flexibility, therefore, requires a component based AI architecture. Each component embodying a suitable solution (S) to a specific subset of IIE problem domains (N). The result of this is that multiple AI solutions can cover the IIE problem domain set (M) as required. This maximises flexibility and maintains efficiency, in turn allowing temporal complexity to be minimised.

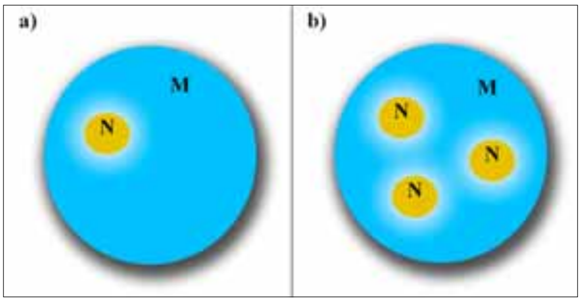


Fig.3. IIE problem domain (M) and the set of sub-domains (N) covered by an associated AI solution (S). (a) For some S. (b) For many S.

2.3 Conclusion

To conclude, the level of flexibility required to power AmI in IIEs, can not be obtained within a single AI representation and processing model (and most certainly not with a temporal complexity low enough to provide a near real time quality of service). Thus we must explore the use of a component based AI architecture.

A summary (in terms of set theory) is given below, with reference to Fig.3. :

Given that :

- The set of possible problem domains within IIEs = M
- An individual AI solution = S
- The set of IIE problem domains covered by an associated individual S = N

And :

- For any S, N  M (N is a subset of M)
- The only possible S where N = M is a universal problem solver
- No universal problem solver exist

Then :

- No S exists where N = M (i.e. there is no individual solution that solves all IIE problems)
- For any S, N  M (N is a proper subset of M). See Fig.3.(a).

  i.e.:
  if x  N
  then x  M and there exists some y
  where y  M and not y  N

Therefore :

- The only way to solve all IIE problems is to use multiple S, whose combined N cover a desired subset of M. This is shown in Fig.3.(b).

## 3. AI UBIQUITY

By definition IIE's are formed from a set of electronic devices that are interconnected in a way that permits communication (forming virtual environments). In the same way that these devices are distributed throughout a physical environment, soft resources are distributed throughout the virtual environments. To realise the AmI vision, intelligence must be made ubiquitous through the virtual environments, allowing these soft resources to operate in a truly distributed fashion.

3.1 Component Services

Resources are made accessible by soft interfaces known as services. Distributed services (DSs) are seen as components or building blocks, from which more complex applications can be constructed. That is to say, distributed services embody functionality as a result of decomposing composite applications.

Taking full advantage of this object oriented view, we can label distributed services as either atomic (irreducible) or complex (reducible i.e. composed of other distributed services), these are known as ADS and CDS respectively. Fig.4. Shows how a CDS can be composed of both ADSs and other CDSs. Recursive decomposition can occur until an application is reduced to a set of atomic distributed services. The recursion is necessary to break apart complex components.

While component membership can be accounted for in a hierarchical model, the interaction of member components can be seen as a graph. The rules of graph theory can help greatly in ensuring the stability of such application compositions. For example "minimum spanning tree"2 or "shortest path"3 algorithms can be used to optimise applications based on certain bias and

1 The diagram shows a linear correlation, this is given to show the general trend. An actual graph with correctly calculated values would not necessarily yield a linear correlation. This is mainly due to the variation in effect on flexibility that each different layer of ontological commitment would have.

2 For example Prim's, Kruskal's algorithms. (Cormen 2001)

3 For example Dijkstra's, Bellman-ford, A* algorithms. (Cormen 2001)

costs (network traffic, trust, pay-per-use, etc.). Graphs such as these allow us to view data inputs and outputs as graph source and sinks respectively.

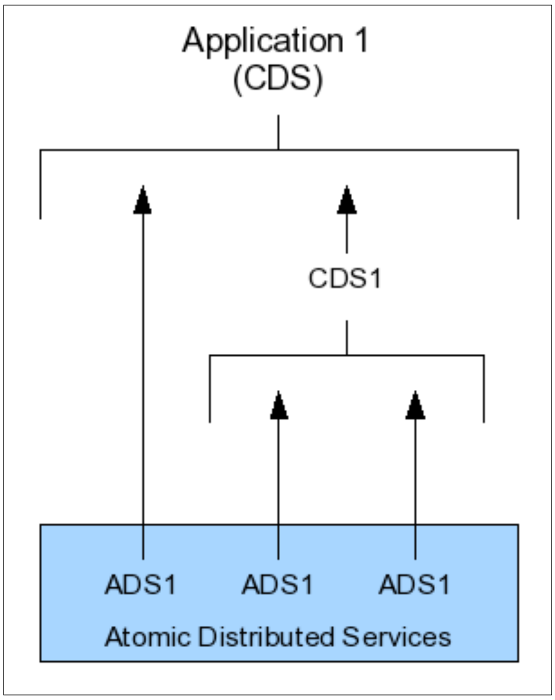


Fig.4. Composition of applications from both ADS and CDS components.

3.2 Distribution Through The Network Ether

Current network topologies are engineered to permit data communication in academic or industrial environments1. So far the proliferation of technology into home and SOHO (Small Office / Home Office) environments has failed to produce a more suitable way of managing device interconnection.

A new breed of interconnection is required that will natively support the specialist requirements of IIE technology. This project has developed the models necessary for this to become a reality and embodied them into a middleware solution called Nexus2.

Nexus forms an abstract topology which is layered over traditional network technology (an overlay network, see Fig.6b). The topology is hierarchical, providing environments within environments. This was primarily designed to allow information and communication to be localised, taking into account spatial proximity of participants. Communication and information diffuse through the hierarchy in a distributed breadth first pattern3, Fig.5. Illustrates this. A breadth first diffusion pattern is used in favour of a depth first pattern in order to guarantee that communication is kept as local as possible. Bias and costs can be factored into this to optimise a diffusion pattern based on purpose (is speed, bandwidth, cost or locality a priority?).

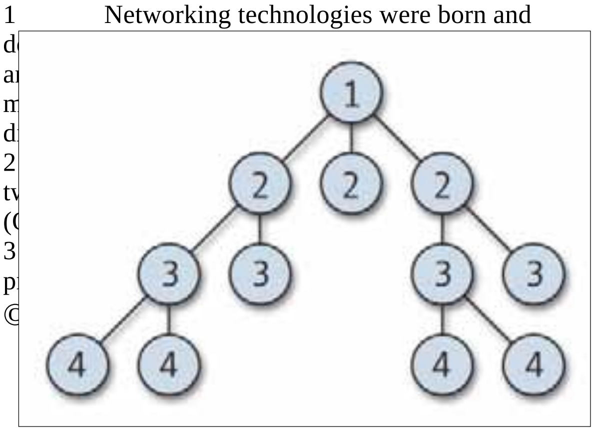


Fig.5. Distributed breadth first routing pattern.

Several benefits also emerged from such a topology, namely :

- Passive location resolution.
- Ease of owner/occupant resolution for resources and environments.
  i.e.
  If :
  resource x resides in space y,
  and :
  principle z owns y,
  then :
  z also owns x
- The virtual nexus environments can form surrogates for physical environments, with spatial relations implied. For example room1 exists within floor1, of building1 (Fig.6a.).
- The topology is easily related to by human spatial cognition. Including areas of space that are labelled despite having no physical partitioning (for example “the corner of the room”, or “that place on the desk”). This same principle also applies to conceptual environments which have no (or loosely bounded) physical presence. For instance a “mobile” (wi-fi / bluetooth / etc.) network which a human may conceive as an environment bounded by the effective irradiation range from an access point (or other node in an ad-hoc topology). Note however, the difference between this and the concept of logical grouping (which is also supported by Nexus and discussed later in this paper – see section 3.4).
- The hierarchy scales both up and down while maintaining quality of service due to distributed management. In a deployment scenario for example, a network provider could provide a root Nexus at the telephony exchange. The consequence of this is a network resolvable right up to town level (indeed nothing exists to stop it scaling up again to national or international level) and right down to John Smiths bed-side table.

It should be noted that IIEs, following a classification suggested by (Russell 1995), have the following properties :

1. Inaccessible – It is not guaranteed that an environments state can be retrieved that is complete, accurate and up-to-date.

1 Networking technologies were born and
d
a
m
d
2
tv
((
3
p
©

2. Non-Deterministic – There is no guarantee a state will result from a certain action or set of actions.
3. Dynamic – The environment changes as a result of many interactions by many different entities.
4. Continuous – There is not a fixed, finite number of actions and percepts.

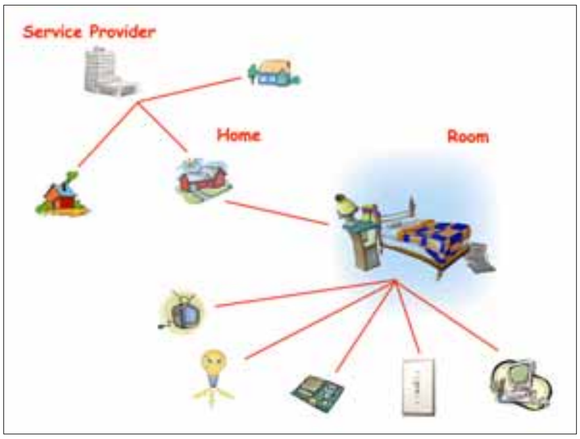


Fig.6a. Hierarchical arrangement of physical world.

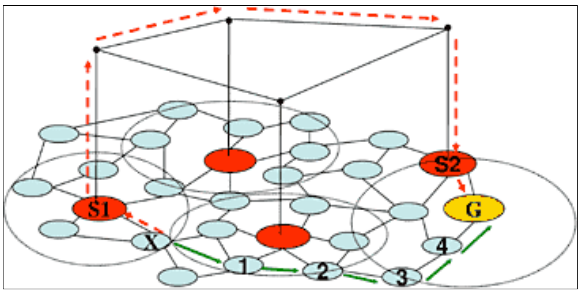


Fig.6b. An overlay network (copyright Intel corporation 2006).

<u>3.3 Agents As Network Peers (Entities)</u>

Nexus allows soft entities to have identity within the network. This identity allows the entity to be individually addressable and to have certain rights and restrictions associated with it. This is essential to permit such entities to operate in IIEs properly. Like humans in physical environments, these entities are mobile across Nexus (virtual) environments. This permits an extremely flexible model for the purposes that soft entities can fulfil.

Although the word "agent" is shrouded in ambiguity, it is the most appropriate term for labelling these entities. According to (Ferber 1999) an agent is a physical or virtual entity which:

- is capable of action in an environment,
- can communicate directly or indirectly with other agents,
- is driven by a set of tendencies (in the form of individual objective or of a satisfaction/survival function which it tries to optimise),
- possesses resources of its own,
- is capable of perceiving its environment,
- has only a partial representation of this environment (and perhaps none at all)
- possesses skills and can offer services
- may be able to reproduce itself
- has behaviour tending towards satisfying its objective(s), taking account of the resources and skills available to it and depending of its perception, its representation and the communication it receives.

Nexus agents fulfil this description with a taxonomy1 to describe/classify them in more detail (Table 1).

<u>TABLE 1 – Nexus Agent Order Software Taxonomy</u>

| Property | Description | Possible Values |
|---|---|---|
| Family | Taxon to group agents by purpose. | Service<br>User[2] |
| Genus | Taxon to group agents by common behavioural characteristics | Reactive<br>Deliberative<br>Social |
| Realm | Scope (extent) of intended agent perception (and possible mobility). | Universe<br>World |
| Period | Intended life span (with period sub-divisions being epoch's). | Transient<br>Volatile<br>Persistent |

<u>Genus.</u> This taxon allows sentient (relating to "able to perceive" not "able to feel") agents to be grouped by common cognitive behavioural characteristics. An overview is given in Fig.7.

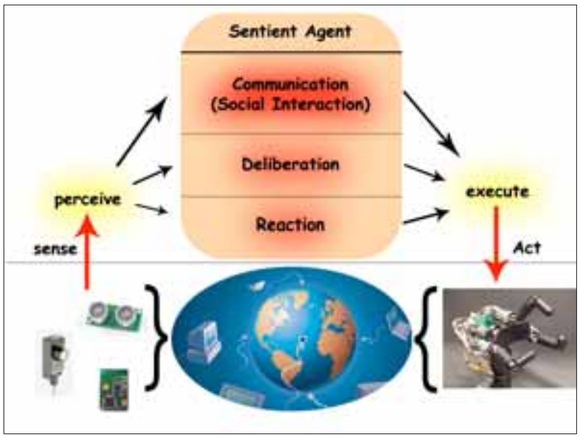


Fig.7. Nexus agent cognitive behaviour layers (Genus)

Valid values here are (currently, and as shown in Table 2) :

- Reactive : Follows "reflex" style behaviour, which is typically static.

---

1 The taxonomy is structured in a way that permits the classification to be embedded in a more general taxonomy. The higher ranked taxons (domain, kingdom, phylum, class, order) have been left for this purpose. This table can be seen as describing the "Nexus Agent" order (of the "artificial" domain, "electronic" kingdom, "software" phylum, "sentient" class).

2 The technology facade discussed in section 1.2 of this paper is an example of a Nexus user agents responsibilities.

- Deliberative : Reasoning occurs (internally) and is very dynamic (learning).
- Social : Reasoning can be aided from external sources (for example ask someone else).

TABLE 2 – Agent Genus Use of Communication

| | **Perform I/O** | **Intelligent** | **Interactive** |
|---|---|---|---|
| Reactive | Yes | -- | -- |
| Deliberative | Yes | Yes | -- |
| Social | Yes | Yes | Yes |

Realm. Scope of intended agent perception and, consequently, the extent to which an agent can be mobile.

Valid values here are :

- Universe : The set of all Nexus worlds that exist. This set is conceptual and not necessarily tangible due to the potential size that scalability permits.
- World : A Nexus world is an identity for a logical set of hierarchically related Nexus environments (equal to the hierarchical root of such a set). The word “world' can be interchanged with the word “Nexus” to describe an environment, where the implication is that a world has child environments (forming a set). For example an individual house can be seen as a world, comprising several child environments such as rooms.

Period. The intended life-span (amount of time in an “alive” state) that an agent is designed to have before expiring (entering a “dead” state).

Valid values here are :

- Transient : A short life-span that implies the agent will only exist temporarily (within its intended scope). Typically service agents will have this time frame, some user agents may require a transient period (for example “guest” user agents).
- Volatile : The agent will exist for as long as it takes to complete its purpose (non perpetual), this time-frame. Typically service agents.
- Persistent : The agent is intended to exist perpetually. This will typically be a user agent, but it may desirable to have some service agents with this period (high responsibility “super” agents).

3.4 Groups

Groups of association between communicating agents allow a medium for social interaction. These groups allow a mechanism for addressing participants that have a common interest. Component services (see section 3.1) also use the group abstraction to enable the composition of applications (CDS).

The group abstraction is designed to enable a secure (groups can have security locks to accept/refuse entry or to “kick” a peer that is to be no longer included) communication environment that is independent of the Nexus hierarchical topology (although security of the Nexus topology is never broken). These social communication environments provide an appropriate medium for interactive groups. It is perfectly acceptable for a single peer to be a member of several groups at once.

## 4. COMPONENT BASED AI ARCHITECTURE

“Knowledge provides the reasoning behind agent action”

The component based AI architecture relies on a knowledge abstraction (KIDAM), which in turn enables a trinity of inter-related entities. This trinity forms the Knowledge Modelling Language (KML).

This architecture allows agents to be producers and consumers of knowledge, the result of which is intelligent reasoning with subsequent action.

4.1 Knowledge-Information-Data Abstract Model (KIDAM)

KIDAM describes the structural (Table 3) and behavioural (Table 4) levels at which knowledge can be handled. This allows things to commit to appropriate semantics for knowledge handling based on what they intend to do with the knowledge. This is necessary to enable the successful creation, use, storage and exchange of knowledge.

TABLE 3 – Structural KIDAM

| **Layer** | **Purpose** | **Description** |
|---|---|---|
| Knowledge | Processing | Committing to a structure of representation (defined by a knowledge language – see section 4.2) |
| Information | Index/exchange | Allows operations independent of what the information means (only commits to the base knowledge tuple structure – see section 4.2). |
| Data | Transport, non-volatile storage | There is no structure imposed. It is simply treated as raw byte data for non-volatile storage, or transport across a communications medium. |

TABLE 4 – Behavioural KIDAM

| **Layer** | **Precision** | **Description** |
|---|---|---|
| Knowledge | Abstract | Descriptive information. |
| Information | Formalised | Qualifies data by giving context. |

| Layer | Precision | Description |
|---|---|---|
| Data | Precise | Specific value that forms a surrogate. |

We can present knowledge as an n-tier model that consists of three layers (Fig.8).

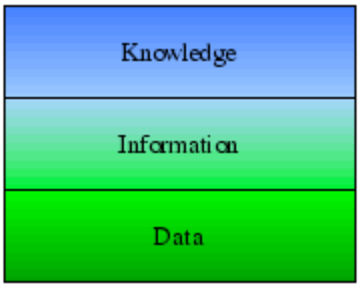


Fig.8. The 3-tier KIDAM stack.

4.2 Knowledge Modelling Language (KML)

The knowledge modelling1 language is composed of three concepts :

- Tuples : Knowledge is captured as a "knowledge tuple", which is an n-tuple structure whose labelled and unordered components (order does not matter due to the labelling) can be complex (other n-tuple structures, for example the triples used in RDF) or atomic (a key-value mapping, where the key is the label). The basic structure of a knowledge tuple is shown in Table 5, this acts as a base format that can be expanded on by a specific language.

TABLE 5 – Knowledge Tuple Base Format

| Component | Required | Description |
|---|---|---|
| ID[2] | Y | Unique identity for this knowledge tuple |
| Language | Y | The language who's semantics this knowledge obeys. |
| Content | Y | An n-tuple complex component that is language specific (defined by language) |

- Languages : Languages define all the information needed to successfully represent and process knowledge in a certain way. Details of this kind of information are given in Table 6.

TABLE 6 – Knowledge Language Description Format

| Property | Description |
|---|---|
| Name | Name of the language |
| Role | The role that this language takes :<br>• Surrogate[3]<br>• Set of ontological commitments[3]<br>• Fragmentary theory of intelligent reasoning[3]<br>• Medium for efficient computation[3]<br>• Medium for human expression[3] |
| Vocabulary / Semantics | Allowable/required values of a knowledge tuple (the content part of the Knowledge tuples), and their semantics (sanctioned / recommended inferences). |
| Motivation | A scenario that illustrates a problem and how the language solves the problem. The scenario helps you understand the more abstract description of the pattern that follows. |
| Applicability | What are the situations in which the language can be applied? How can these situations be recognised? |
| Structure | A graphical representation of this language. |
| Participants | Producers / Intermediates / Consumers of this language. |
| Collaborations | How the participants collaborate to carry out their responsibilities. |
| Consequences | How does the language support its objectives? What are the trade off's and results of using the language? |
| Implementation | What pitfalls, hints, or techniques should you be aware of when implementing the language? |

- Renderers : A Knowledge renderer processes knowledge tuples according to the semantics defined by the appropriate language. In effect, a renderer is a language implementation. These renderers exist as lightweight plug-in based processors which can offer processing functionality accessed by an Information Communication Language (ICL).

The relations between the trinity is shown in Fig.9.

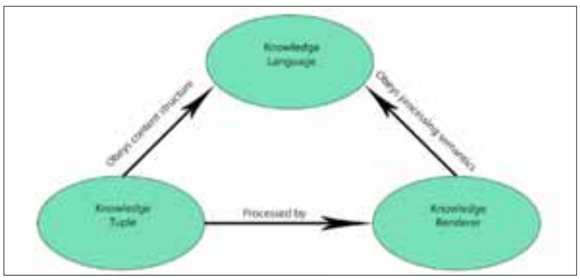


Fig.9. The relations between the KML trinity.

1 Modelling is perhaps a misleading description as it implies a strict engineering of knowledge, but it is the most appropriate way of labelling the trinety.

2 Future work will replace this static field with a more natural way of tuple indexing, e.g. a composite key.

3 As suggested by Davis (1993)

4.3 Information Communication Language (ICL)

Operating at the information level of the structural KIDAM model, the ICL allows information to be communicated between entities. This is similar in operation to an agent communication language[1] (ACL). The semantics of this language are still being evaluated.

## 5. IIE FRAMEWORK

Tying all the parts of this work together into a logical whole that can be comprehended has been achieved by use of a framework (labelled OFFIE : Open Framework For Intelligent Environments). This framework is intended as a pluggable operating platform, the structure of which is shown in Fig.10.

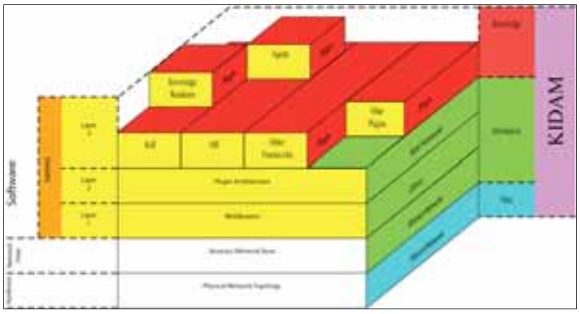


Fig.10. The OFFIE framework.

The layers that make up the framework are (from the ground up):

1. Physical Network Topology : This layer accounts for the network of devices and interconnections typically seen in a home or office environment. This is anticipated to be a TCP/IP based network (running over Ethernet, Wi-Fi, Bluetooth, etc) with security measures such as firewalls, NAT (Network Address Translation), gateways , etc.
2. Abstract Network Topology (Nexus) : This layer gives us a an abstract model (overlay network) of the physical network which is enforced by SIMPAKS Layer 1.
3. Scalable Intelligent Middle-Ware Providing Ambient Knowledge Support (SIMPAKS) : This grouping of three sub-layers represents the deployable software that enables the pervasive computing vision.

   - Layer 1 : Middle-Ware : This software layer handles communications between the peer and Nexus network. This middle-ware offers a heterogeneous physical network the ability to be populated by many intelligent peers which can co-operate in a secure way.
   - Layer 2 : Plug-in Architecture : The framework is designed to be easily configurable and extensible with a plug-in architecture. This layer forms the basis and provides management of the run-time peer environment. All software entities that exist in SIMPAKS layer 3 are plug-ins (which may in turn be frameworks that support sub-plug-ins) and are managed by this layer.
   - Layer 3 : Plug-ins : This layer contains all application plug-ins that compose a peers functionality. Two such plug-ins are the Agent Support Framework (ASF) and the Knowledge Support Framework (KSF).

## 6. CURRENT PROJECT STATUS AND FURTHER WORK

Currently the project is in the implementation phase.

Conclusions of work so far
Future language implementations
Proving the concept

## GLOSSARY

| | |
|---|---|
| ADS | Atomic Distributed Service |
| AI | Artificial Intelligence |
| AmI | Ambient Intelligence |
| CDS | Complex Distributed Service |
| DS | Distributed Service |
| IIE | Intelligent Inhabited Environment |
| Nexus | Middleware for IIEs; A Nexus : a single virtual environment (in the Nexus middleware model) |

world, universe, groups, icl, kidam, tuples, langs, renderers, offie, simpaks, asf, ksf

## REFERENCES


Clarke, A.C. (1999). Clarke's Third Law. In: Profiles of the Future: An Inquiry into the Limits of the Possible. Weidenfeld & Nicolson,

Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein (2001). Introduction to Algorithms, Chapters 24: Single-Source Shortest Paths, and 25: All-Pairs Shortest Paths, pp.580–642. MIT Press and McGraw-Hill.

Davis, R., S. Howard and P. Szolovits (1993). What Is A Knowledge Representation. In: AI Magazine, 14, 17-33.

Ferber, J. (1999). Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Addison Wesley Longman.

Oxford Dictionary Online. www.oed.com. Oxford University Press.

Russell, S. and P. Norvig (1995). Artificial intelligence: a modern approach. Prentice hall.


1 The ICL would actually be labelled an ACL if the participants were restricted to agents only.