Sentient Computing for Everyone

Diego López de Ipiña (1) and Sai-Lai Lo (2)
(1) Laboratory for Communications Engineering, Department of Engineering, University of Cambridge, Cambridge, UK
(2) AT&T Laboratories Cambridge, 24a Trumpington Street, Cambridge, UK

Key words: Context-Aware Computing, Location-Aware Computing, CORBA, Mobility

Abstract: Sentient Computing gives perception to computing systems so that they can detect, interpret and respond to changing aspects of user contexts. The location attribute of a user's context is of special interest because it makes human-computer interactions more natural. In the last few years, several sophisticated indoor location technologies, which can track user whereabouts, have been developed. However, they are yet to be widely adopted because of their high cost and complexities in deployment, configuration and maintenance. This paper describes a novel vision-based software location system, known as TRIP, whose low-cost, off-the-shelf hardware requirements and easy deployment features overcome other systems' limitations. Nevertheless, in order to foster the deployment of "sentient spaces" that bring services to users wherever they are or about to move to, a location system must also be accompanied by the middleware to facilitate user-bound software service activation, migration and deactivation. LocALE addresses this issue by providing a CORBA-based solution that deals with heterogeneous object lifecycle and location control. Some distributed applications combining TRIP's and LocALE's capabilities are presented to demonstrate that Sentient Computing can be made readily available for everyone and everywhere.

1. INTRODUCTION

Ubiquitous Computing [16] envisions physical spaces, such as offices, meeting rooms or homes, to be augmented with fully integrated computing devices. It aims to make services provided by these devices as commonly

available as electricity. With Ubiquitous Computing, personal computers are no longer the focus of attention. Instead, the devices in the physical space provide their services unobtrusively and may not require the direct commands from the user.

Sentient Computing [5] is our approach to make Ubiquitous Computing a reality. It stems from the basic idea that to make computerised services pervasive in our life, the devices providing such services must be given *perception*, i.e. the capability to *see* or *hear* what entities are around them; what these entities are doing; where they are and when something is happening. Only then it is possible for the underlying computing system to undertake suitable actions to match the end user expectations. For example, when a user enters his office, depending on the time and day of the week, the sentient environment could automatically initiate the playback of a specific kind of music. If a colleague later on comes into the room to discuss some project ideas, the music volume could be automatically adjusted to facilitate communication. In order to do so, Sentient Computing must collect data from a variety of sensors in the environment and act upon these stimuli according to previously specified user preferences.

Location-Aware Computing [9], whose behaviour is determined by the position of objects in the environment, represents an interesting subset of the Sentient Computing paradigm since location is often an essential attribute of the context. This has motivated the development of several indoor location systems featuring different location granularity, ranging from room-scale resolution such as the infrared-based Active Badge [15], to the more accurate 3D resolution offered by systems such as the ultrasound-based Active Bat [4] or the radio-based 3D-iD [17]. These systems require people or objects to wear an electronic tag that transmits a unique identifier via infrared, ultrasound or radio to a network of sensors on the walls or ceilings. A *Location Server* then polls and analyses the data from the sensors and makes the information available to applications. These systems share several drawbacks. The tags they use need battery power and are expensive. The infrastructure required, a network of sensors, is complex and expensive to install and maintain. These factors may limit their practical use outside research laboratories. We have devised an alternative sensor technology, called TRIP, that provides a better trade-off between the price and flexibility of the technology and the accuracy of the location data provided. TRIP is a vision-based sensor technology that recognises and locates printed 2D circular barcode tags using inexpensive low-resolution CCD cameras.

In order to promote the use of the Sentient Computing paradigm in our living spaces, it is not sufficient to make computing systems aware of the user's presence, movement or precise location, it is important that these systems are provided with the capability to automatically activate services

on behalf of the user when the appropriate contextual conditions are met. Furthermore, it is desirable to enable user-bound services to follow the user as he moves through the physical space, and to deactivate such services when the user leaves the space. In other words, what is needed is a middleware infrastructure that gives sentient applications the control over a computing object's lifecycle and location in a network. The LocALE framework is our middleware solution to this need. Sections 2 and 3 explore the TRIP technology and the distributed system infrastructure built on top of it. LocALE is introduced in section 4. Section 5 describes some applications combining TRIP and LocALE. Section 6 summarizes some related research. This is followed by the conclusions in section 7.

2. TRIP: A DOWNLOADABLE LOCATION SENSOR

TRIP (Target Recognition using Image Processing) [8] is a novel vision-based sensor system that uses a combination of visual markers (2-D *ringcodes*) and conventional video cameras to identify tagged objects in the field of view. Relatively low CPU demanding image processing and computer vision algorithms are applied to video frames to obtain the identifier (*TRIPcode*) and pose (3D location and orientation) of the targets relative to the viewing camera.

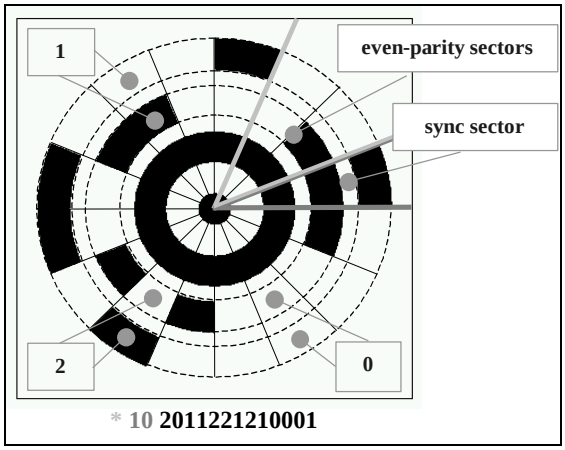

Figure 1. TRIPcode representing number 1,160,407

TRIP constitutes a very cheap and versatile sensor technology. Its 2-D ringcodes are printable, and can therefore be attached even to low-cost items, such as books or office stationeries (e.g. stapler). A TRIPcode (see Figure 1), read in counter-clockwise fashion from its synchronisation sector, represents

a ternary number in the range $1\text{-}3^{13}$ (1,594,323). Only off-the-shelf hardware is required, i.e. low-resolution CCD cameras and CPU processing power.

The TRIP software will be made publicly downloadable in due course. Users with a web-cam attached to their PCs can easily install and run this software sensor and hence provide visual awareness to their computers.

When the Target Recognition Algorithm is applied to the images, a set of image processing stages is executed in order to determine the identities and positions (x-y coordinates in the image) where TRIPcodes are spotted. [8] fully describes the mathematics involved. In addition, a Pose Extraction Algorithm is employed. Given a single view of a target, this method determines the translation vector (t_x, t_y, t_z) and rotation angles (α, β, γ) that define the rigid body transformation between the camera coordinate system and the target-centred coordinate system. The method applied is based on the POSE_FROM_CIRCLE method given in [3].

The C++ implementation of the Target Recognition Algorithm processes 15 640x480 pixels frames per second on an 800 MHz Pentium III. When the target recognition and pose estimation are simultaneously undertaken, the performance achieved is about 12Hz. TRIPcodes are recognised as long as the target's plane orientation angles are less than 70º and its image occupies at least 20x20 pixels. The 3D location of the centre of a target relative to the viewing camera is obtained with less than 5% error. An even error parity check (see Figure 1) is used to prevent the identification of false positives.

3. TRIP: A DISTRIBUTED SENSOR SYSTEM

An event-based distributed architecture has been devised around TRIP in order to manipulate and distribute the sensor data provided by this technology. The functionality of the TRIP system, i.e. its target recognition and pose extraction, has been encapsulated by a CORBA [11] component, named *TRIParser*. This component offers a UNIX `pipe`-like interface that enables applications to connect distributed *Frame Grabber* components, which supply images from cameras, to TRIParsers. A TRIParser may consume images supplied by one or more *Frame Grabbers*. TRIP processing results are, by default, asynchronously communicated in an event-form to a CORBA Notification Channel [10], which is associated with each TRIParser. The Notification Channel serves as a de-coupler between the frame analysers and the components interested in the results.

A TRIParser pushes *TRIPevents* (see Figure 2) to its associated Notification Channel. The parties interested in a given TRIParser's location data subscribe to its associated channel and convey a set of constraints that specify the events they are interested in receiving. The Notification Channel

undertakes consumer registration, event filtering and communication on behalf of its representing TRIParser. The filter constraint language supported is the constraint language defined by the OMG Trading Service. Hierarchical interconnections of these Notification Channels can be created in order to ensure the efficient and scalable dissemination of the TRIParser output. The TRIParser also provides a synchronous invocation interface (`processFrame`) that analyses a submitted frame and returns the location data inferred from it. Hence, applications can interact with a TRIParser either synchronously or asynchronously. However, for efficiency purposes, an application should establish direct communication between a *Frame Grabber* and a TRIParser, and register as an event consumer to the parser's Notification Channel.

```
struct TRIPevent {
  string TRIPcode; // code ternary representation
  string cameraID; // capturing camera identifier
  paramsEllipse params; // bull's-eye's outer ellipse parameters (x,y,a,b, θ)
  targetPosition pose;  // translation vector from camera origin to target centre
  targetOrientation angels; // (α, β, γ)
};
```

Figure 2. TRIPevent contents

3.1 The TRIP Directory Server

A TRIP Directory Server (TDS) has been created with the purpose of regulating the TRIPcode granting process and establishing mappings between real-world objects and TRIPcodes. This component ensures the efficient utilisation of the TRIPcode address space and its classification into categories. The TDS offers CORBA interfaces for the creation, modification, deletion and retrieval of both TRIPcodes and their categories.

3.2 The Sentient Information Framework

The Sentient Information Framework (SIF) defines an application construction model to streamline sentient application development. SIF isolates context capture and abstraction from application semantics. At the same time, it provides efficient mechanisms for context communication. Its main function is to transform context information into the formats demanded by the applications. SIF is not constrained to TRIP since it is sensor-technology independent. In this framework, components are categorised into 3 types (see Figure 3): (1) *Context Generators* (CGs), (2) *Context Channels* (CCs) and (3) *Context Abstractors* (CAs). *Context Generators* encapsulate

sensors, such as TRIP, and the software that extracts information from them. They are sources of sensorial events. *Context Channels*, in our case implemented as CORBA Notification Channels, receive events and, after consumer specified filtering, pass the events to the consumers. *Context Abstractors* achieve the separation of concerns between context sensing and application semantics. They consume the raw sentient data provided by CGs (e.g. TRIPcode 1234 spotted), interpret its contents (e.g. user Diego spotted) and augment it (e.g. Diego's login is dipina) to produce enhanced contextual events that can be directly used by applications. They can also *correlate* other CAs' or CGs' outcomes to generate the required sentient data. For a more detailed description of SIF refer to [6].

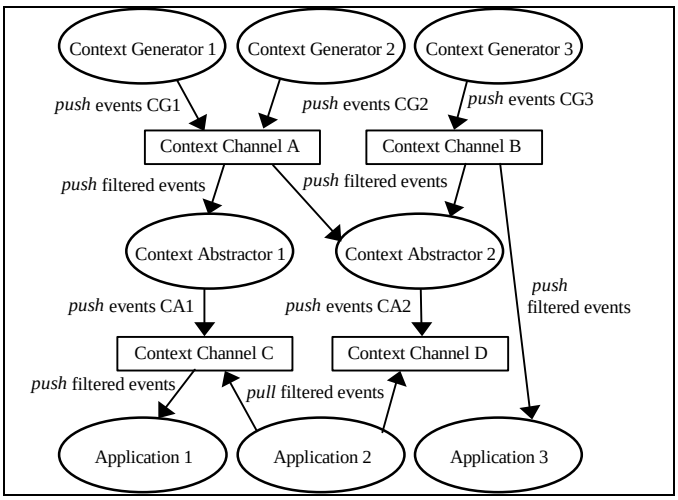

Figure 3. The SIF Architecture

4. LOCALE: MIDDLEWARE FOR OBJECT LIFECYCLE AND LOCATION CONTROL

In order to make sentient computing available for everyone and everywhere, we have, so far, described a new cost-effective and easily deployable location sensor technology, TRIP, and a sentient application construction model, SIF, for the efficient manipulation and dissemination of sensorial data. However, one question still has to be answered, i.e. once a sentient system receives a high level event (e.g. Diego enters his office), how can the system effectively trigger the required action, i.e. activates, deactivates or migrates a user-bound software service?

Our experience with sentient application development has pointed us to the need for an infrastructure to streamline the lifecycle control of user-associated services. Otherwise, every time a new sentient application is developed, the distributed components involved in its operation have to be manually started. This makes sentient application deployment very cumbersome. Moreover, user-related services are often bound to user locations, e.g. the activation of an MP3 player must take place in one of the PCs in a room where a user presence is detected. Therefore, it is desirable to control both the lifecycle and location of user-bound services by a middleware designed for this purpose.

The LocALE (Location-Aware Lifecycle Environment) framework [7] defines a simple mechanism for managing the lifecycle and location of distributed CORBA objects residing on a network. In addition, it provides load-balancing and fault-tolerance features to the objects whose lifecycle it manages. The emphasis of its design is placed on providing a suitable interface for third party *object-location controllers*. These controllers can intelligently direct when and where components are activated or moved based on the inputs on user whereabouts and the location, load and capabilities of computing resources. LocALE offers an object-lifecycle infrastructure, simplifying sentient application deployment and enabling CPU intensive systems, such as TRIP, to reuse the spare resources available in a network.

4.1 LocALE Architecture

The LocALE middleware has a 3½-tier architecture (Figure 4). It consists of client applications and the following 3 types of components:

– The *Lifecycle Manager* (LCManager) provides object-type independent lifecycle control interfaces and mediates the lifecycle operations over any CORBA object residing in a location domain. A location domain is a group of machines on a LAN that are physically located in the same place, such as a building, a floor or a room. Every object creation, movement or deletion request is routed through this component. This permits the LCManager to cache the current location of every object in a domain and thus act as a forwarding agent that redirects client requests after object references held by clients are broken due to either object movement or failure. In the latter case, the LCManager tries to recover the failed object, and, in case of success, returns the new object reference.
– *Lifecycle Server(s)* (LCServer) are containers of LocALE-enabled objects that are subjected to lifecycle control. Lifecycle operations invoked on an LCManager are delegated to suitable LCServers. They subsume standard strongly typed factory functionalities by providing a type specific

creation method with hard-coded types. Furthermore, they also assist their local objects on their migration to other LCServers. They can be started either manually or by the local LCManager after a creation or migration request arrives at a location where the required LCServer type does not exist. In either case, they register with the manager by passing on their physical location (host), their CORBA object reference (or IOR) and data specific to the type of objects they handle.

– *Type-specific Proxy Factories* are placed in between clients and LCManagers. They save client applications from dealing with the generic object creation interface offered by an LCManager. Using type-specific interfaces makes client code far shorter and simpler to understand. With the generic version, if a mismatch in the type of a constructor argument occurs, the client would only receive an error at run-time rather than at compile time. *Type-specific Proxy Factories* offer type specific object-creation methods with the same argument types and semantics as the LCServers they represent. They map type specific requests to the generic format demanded by LCManagers. They are on demand activated by the local LCManager.

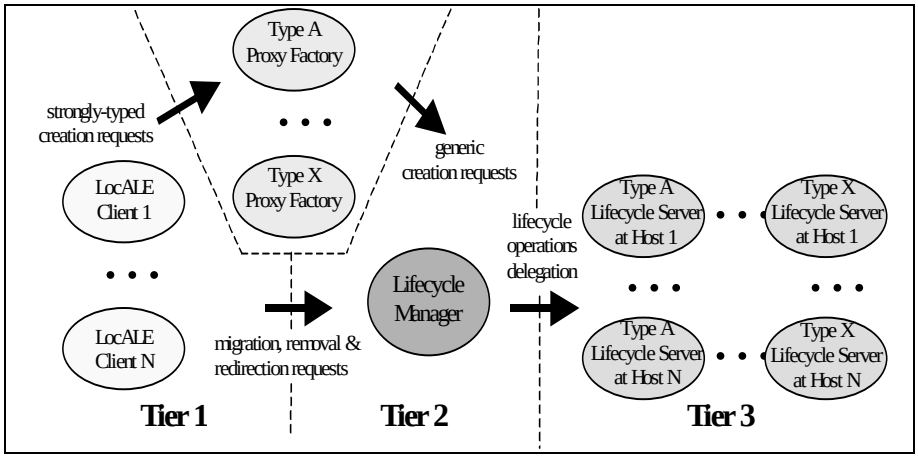

Figure 4. LocALE 3½-tier architecture

The LocALE architecture guarantees compile-time type safety of clients with respect to the lifecycle control of their services. Clients find, through the LCManager, object references to type-specific proxy factories and issue through them object creation requests. Object migration and deletion requests are directly invoked on the LCManager because they are object type independent. The LCManager then delegates incoming lifecycle operations to the appropriate LCServers.

4.2 Location-constrained Lifecycle Control

One of the main advantages of CORBA [11] is that it offers location transparency to applications, i.e. it makes method invocations as simple on remote objects as on local objects. LocALE leverages on CORBA's location transparency but it is able to control *where* services are located. It can also set the constraints under which these services can later be re-located. We think these functions may bring important benefits. For example, load-balancing systems may wish to initiate new service instances on the hosts of the same LAN with the lowest processing load. Follow-me sentient applications may want to move objects tied to a user's physical location to the nearest host with the required capabilities. LocALE allows applications to control the location of CORBA objects. It extends conventional object factories with additional distributed object construction semantics. The following two attributes are appended to the object creation interfaces offered by the proxy factories:

– *Location* attribute (`LocSpec`): specifies *"where"* a service should be instantiated (or moved to). The formats available for location specification are: (1) `hostDN(hostName)`, (2) `roomID(room)` and (3) `hostGroup(listHostName)`. `hostDN` and `roomID` are used when the application wants to create an object on a specific host or in a specific room. By setting `hostName` or `room` attribute to `"ANY"` in either of these `LocSpecs`, the programmer can instantiate location-independent services. The `hostGroup` format is useful to specify an arbitrary set of hosts where an object may be created, e.g. hosts in a room with audio capabilities.
– *LifeCycle contraints* (`LCconstraints`) attribute: determines whether the object created should be considered as `RECOVERABLE` and/or `MOVABLE` within the location scope in which it is created, i.e. on a host, in a room or a host group. A recoverable object is either a stateless or a persistent object whose state can be restored after failure.

With respect to the ability to create objects based on the location constraints supplied, the LCManager provides a similar function as a conventional Trader. The LCManager matches client object creation specifications against the registered object Lifecycle Server types and locations, and delegates the operation to a suitable LCServer.

4.3 LocALE Implementation

The LCManager has been implemented in C++ using the CORBA 2.3 C++ ORB omniORB [1]. Its implementation makes extensive use of some advanced features provided by the CORBA 2.3's Portable Object Adapter (POA). Among these, it relies on the POA defined standard mechanism to

generate LOCATION_FORWARD reply messages. These messages are sent by LCManagers to client ORBs to indicate the new location of an object where previously issued requests have to be redirected. A detailed description on the implementation of the LocALE middleware is available in [7].

5. TRIP-ENABLED SENTIENT APPLICATIONS

This section describes three sentient applications[1] that combine TRIP sensing with LocALE's object lifecycle and location management. They demonstrate how our work streamlines the deployment of sentient systems and make it a cost-effective process.

5.1 The LCE Sentient Library

This system augments a conventional library catalogue system with sentient features. Apart from the typical functionalities expected in a book cataloguing system, the LCE Sentient Library offers contextual information about the books in our lab. Details on the last seen book location and its proximity to other fixed items in the environment are offered. This application is an illustration of TRIP's versatility for tracking any tag-able entities in the environment.

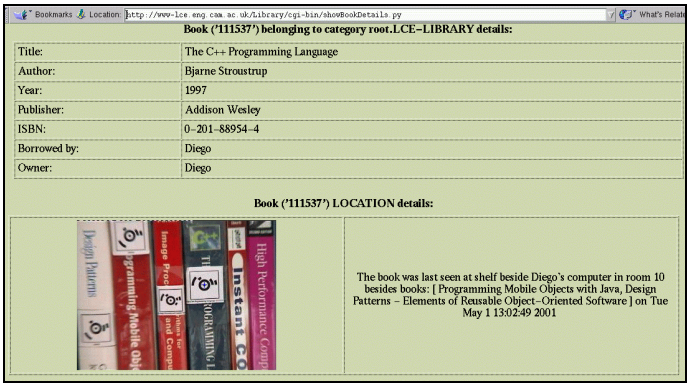

Figure 5. LCE Sentient Library snapshot

Every location where a book may be located in our lab has been tagged with a *location-category* TRIPcode. Similarly, *book-category* TRIPcodes have been attached to book spines. Periodically, the LCE librarian wanders

[1] The first two applications are available at: http://www-lce.eng.cam.ac.uk/~dl231/#Demos.

around our lab with a video camera recording TRIP tagged locations and books. The system automatically updates the book database by the off-line processing of the book-sighting video. This process involves the co-operation of a Video Frame Grabber, which provides access to the video frames, a TRIParser, which analyses those frames, and the TRIP Directory Server, where book details and contextual data are recorded. The Video Frame Grabber and TRIParser components are automatically activated by LocALE.

A web interface allows LCE members to (1) browse all the book categories, (2) list books in a category, (3) obtain book details and (4) perform keyword-based search of books. Figure 5 shows the result of a book search with this web interface.

5.2 Active TRIPboard

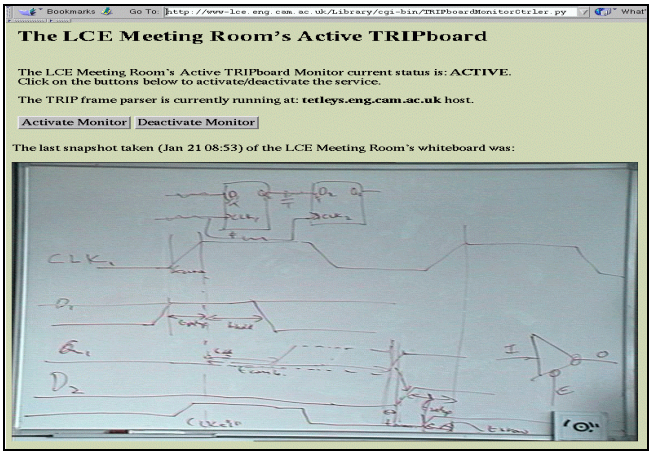

Figure 6. Active TRIPboard snapshot

This application augments a conventional whiteboard with interactive commands issued by placing TRIPcodes in the field of view of a camera pointing at the whiteboard. Some example actions that can be triggered are: (1) capture whiteboard snapshot and email its web link to people currently present in the room, (2) printout a copy of the whiteboard snapshot for each person in the room. The application components: a Frame Grabber and a TRIParser are activated via LocALE either when a person appears in the meeting room or through a web interface (see Figure 6). The Active Badge indoor location system deployed in our lab is used to determine person presence. In future, the meeting room will be populated with sufficient cameras to cover all the field of views of the room, and so permit people

wearing TRIPcode tags to be detected. Through LocALE, the TRIParser is activated in a load-balancing way by randomly picking one of the hosts in a candidate list. If the TRIParser fails, the application recovers transparently because LocALE's fault-tolerance support will recreate the parser on one of the available host.

5.4 Follow-me Audio

This application provides mobile users with music from the nearest speakers wherever they are. The source of music is an MP3 jukebox server whose operation is controlled by showing jukebox-type TRIPcodes to web-cams attached to some of our lab's PCs. A personal software agent, associated with each lab person, listens for that person's movement events generated by a *Person Location Monitor Context Abstractor*. This component obtains people sightings from the *Context Channels* of the TRIParsers corresponding to the PC web-cams spread throughout the laboratory. From raw TRIP sighting events, it generates personnel presence and movement events. The personal agent, acting as a component migration controller, requests LocALE to migrate the audio player and MP3 decoder components to the host nearest to the user that can deliver the music. As the user wanders around the location-aware computing environment, the music follows him. The state of the system and time index into the current song persists as the components migrate. This application makes use of TRIP's tracking capabilities, SIF's context modelling features and LocALE's migration support.

6. RELATED WORK

SONY's CyberCode [12] is a visual tagging system based on a 2D square barcode that can be used to determine the 3D position and identifier of tagged objects. Several augmented reality applications have been produced based on this system, e.g. the visitor view of CyberCode tagged items in a museum is augmented with synthesised information. The geometric features of CyberCodes require higher image resolution for their accurate recognition and location than TRIP. BBC's free-d [14] location system measures the precise position and orientation of studio cameras, by using an auxiliary camera mounted on the back of a conventional moving camera pointing to circular markers, similar to TRIPcodes, placed on the ceiling of a TV recording studio. A hardware implementation of its algorithms was necessary to achieve real time processing. The system is used for virtual reality TV production. It is expensive and cumbersome to deploy.

GeorgiaTech's Context Architecture project [2] attempts to make sentient application development as simple as GUI development. For that it introduces Context Widgets that separate context sensing from context-use. It has some similarities to SIF but doesn't tackle efficient context information dissemination. Microsoft's EasyLiving [13] project tries to create reactive context-aware living spaces without the user having to wear any location tag or computing device. Their approach tracks people based on colour histograms without requiring the user to wear any marker. This has a much heavier computational demand and produces less reliable results than TRIP. AT&T's Sentient Computing project [5] aims to replace human-computer direct interaction (through mouse or keyboard) by enabling users to instead interact with their surrounding space based on the precise location and orientation provided by the Active Bat Location System [4]. This concept has been explored with several sophisticated sentient applications.

7. CONCLUSION

TRIP, a novel cost-effective and easily deployable location sensor technology, has been introduced. This sensor requires only inexpensive and off-the-shelf hardware, which makes the creation of location-aware reactive environments, even in the home, an affordable proposition. All that is required to augment a standard PC with visual awareness is a web-cam and TRIP's downloadable software.

SIF, an application construction model to ease the development of distributed sensor-driven systems has also been described.

LocALE, an object lifecycle and location control middleware that streamlines user-bound service activation, migration and deactivation and, at the same time, permits application developers to reuse the spare networked computing resources in a LAN, has been presented. LocALE complements TRIP's minimal cost and deployment complexity. It is a very useful tool for sentient applications that have to trigger user-related services in response to captured sensor data.

To validate our contributions, several TRIP-enabled applications, following the SIF model and leveraging on the LocALE infrastructure have been developed and described in this paper. The initial results are encouraging.

ACKNOWLEDGEMENTS

Diego López de Ipiña is very grateful to the Basque Government's Department of Education for the sponsorship of his PhD studies. Thanks are also due to AT&T Laboratories Cambridge for sponsoring the TRIP project.

REFERENCES

[1] AT&T Laboratories Cambridge, "omniORB C++ CORBA ORB", http://www.uk.research.att.com/omniORB/omniORB.html

[2] Dey A.K., Salber D., Futakawa M. and Abowd G. "An architecture to support context-aware applications", 12th ACM's UIST, 1999

[3] Forsyth D., Mundy J.L., Zisserman A., Coelho C., Heller A. and Rothwell C. "Invariant Descriptors for 3-D Object Recognition and Pose", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 10, pp. 971-991, October 1991

[4] Harter A., Hopper A., Steggles P., Ward A. and Webster P. "The Anatomy of a Context-Aware Application", Proceedings of MOBICOM'99, August 1999

[5] Hopper. A. "Sentient Computing", The Royal Society Clifford Paterson Lecture, AT&T Laboratories Cambridge, Technical Report 1999.12, 1999

[6] López de Ipiña, D. "Building Components for a Distributed Sentient Framework with Python and CORBA ", Proceedings of the 8th International Python Conference, January 24 - 27, 2000

[7] López de Ipiña, D. and Lo S. "LocALE: a Location-Aware Lifecycle Environment for Ubiquitous Computing", Proceedings of ICOIN-15, February 2001

[8] López de Ipiña D., "TRIP: A Distributed vision-based Sensor System", Technical Report, Laboratory for Communication Engineering, Cambridge, September 1999

[9] Nelson G. "Context-Aware and Location Systems". PhD Thesis. Cambridge University Computer Lab, UK, January 1998

[10] Object Management Group. "Notification Service Specification", June 2000

[11] Object Management Group. "The Common Object Request Broker Architecture: Architecture and Specification", October 1999

[12] Rekimoto J. and Ayatsuka Y. "CyberCode: Designing Augmented Reality Environments with Visual Tags", Designing Augmented Reality Environments (DARE 2000), 2000

[13] Shafer S, et al. "The new EasyLiving Project at Microsoft Research, Microsoft, 1999

[14] Thomas G. A., Jin J., Niblett T., Urquhart C. "A versatile camera position measurement system for virtual reality in TV production", Proceedings of IBC'97, September 1997

[15] Want R., Hopper A., Falcão A. and Gibbons J. "The Active Badge Location System", ACM Transactions on Information Systems, Vol. 10, No. 1. 91-102, January 1992

[16] Weiser M. "The Computer for the 21st Century". Scientific American, September 1992

[17] Werb J. and Lanzl C. "Designing a positioning system for finding things and people indoors", IEEE Spectrum, pp.71-78, September 1998