An XML Based Framework for Cognitive Vision Architectures

Sebastian Wrede, Jannik Fritsch, Christian Bauckhage and Gerhard Sagerer
Bielefeld University, Faculty of Technology,
P.O. Box 100131, 33501 Bielefeld, Germany
Email: \{swrede,jannik,cbauckha,sagerer\}@techfak.uni-bielefeld.de

Abstract

Distributed processing and memory structures are very important aspects of cognitive vision systems. Both issues not only require sophisticated conceptual designs but also pose problems of software and systems engineering. In this paper, we describe a general XML based solution to these problems. Practical experiences are reported to underline its suitability.

1. Introduction

In vision systems, a variety of image processing functions needs to be carried out. To achieve fast processing in complex systems the distribution of algorithms over several nodes is necessary. Usually this is not supported by vision libraries like OpenCV, whereas middleware approaches like CORBA are too generic to provide a simple solution for distributing vision algorithms. Consequently, specialised frameworks have been developed, e.g., [13] that often focus on single aspects like control or communication.

Given that memory and distribution are fundamental requirements for cognitive vision systems [4], supporting software frameworks for their integration and construction are needed. We propose a generalised solution for cognitive vision system engineering by way of an active memory infrastructure. Concerning the realisation of such an active memory, there are two major issues: a suitable communication framework allowing to distribute the different algorithms over several computers as well as information storage and data organisation for the active memory itself.

Note that the development of complex cognitive vision architectures with many independent researchers involved is not only a matter of conceptual design but also a system engineering task. Therefore we take into account not only functional but also non-functional requirements from modern software engineering. Furthermore, earlier experiences with multi-modal system integration [1] or robot companions [9], and as a basis thereof our communication system DACS [7] have strongly influenced our design decisions.

The solution we present in this paper emerged from a cognitive vision project aiming at the development of a visual active memory. The demonstrator realised within this project serves as personal assistant in an office scenario that memorises visually recognised objects and actions [2]. The scene is observed by a helmet-mounted camera, being part of a mobile AR-gear [15]. The detection of relevant objects like keyboard, cups, etc. is realised using a robust object detection approach introduced by Viola and Jones [17] in conjunction with a tracking algorithm based on hyper-planes [10]. Action recognition is done using the trajectory of the hand in the scene [8].

The proposed holistic integration approach not only enables us to build a system with support for distribution and memory. It also helps to efficiently and transparently organise the construction process for research projects on integrated prototypes. This facilitates an agile software development process in the academic domain which has been proven useful in software industry.

The next section outlines several important non-functional requirements that have been identified; section 3 explains our integration concept. Practical experiences using this approach are described in section 4. Finally the paper is summarised and an outlook is given.

2. System Engineering Requirements

As integration is a complex task and requires sophisticated knowledge of middleware like, e.g., CORBA, vision researchers tend to concentrate on the development of single algorithms. As important criteria for system construction, frequent integration cycles can only be achieved when module developers are able to easily build components with the integration framework. Therefore Simplicity, Usability and Standards Compliance are critical.

Often specifications change frequently in a research project. Thus an important feature is the Flexibility of the integration framework. The impact of interface changes on an existing system architecture should be minimal to avoid the versioning problem known from CORBA [5].
Another important requirement that benefits directly from high usability and flexibility is Rapid Prototyping. As consequence, iterative development should not only be supported for single modules but also for the integrated system. Wrong directions in system evolution can more easily be identified if integration is performed on a regular basis starting at an early project stage.

For large scale software systems, software engineering research has shown that decoupling of modules is important. Thus the framework should support Low Coupling of modules. If supported, this facilitates not only independent operation of components but also minimal impact of local changes on the whole system. With a framework that adheres to low coupling, Debugging and Evaluation of a running system architecture can easier be supported. One way to achieve this is the ability to replay previously recorded data to substitute architectural components. This also supports an evaluation based on a common corpus of data at the various architectural levels of a cognitive vision system.

3. Memory Infrastructure

Keeping the above goals in mind, this section explains our integration approach. After introducing the concept of our active memory [18], the information storage part is addressed by a description of the Memory Interface.

The remainder of this section describes XCF, the XML enabled Communication Framework that is used to connect external computational modules to one or more instances of an active memory or directly to other system components. The XCF framework allows module developers to easily exchange XML messages and referenced binary data types. Together, XCF and the memory interface constitute the Memory Infrastructure.

Conceptual Architecture Figure 1 provides an overview of the current system architecture and the different processes, e.g., action recognition, that are connected to the active memory as described in the introduction.

Figure 1. Visual active memory architecture.

The first consideration about the concept of an active memory regards the atomic information that can be collected in a storage. We term this entity a Memory Element. These elements are XML documents [3] that follow an object-oriented paradigm validated with inheritance-based XML scheme specifications [3]. Conceptual Domains separate the memory elements through different hierarchically organised semantic categories that store for example image data (i.e., patches cropped from images) as well as more abstract descriptions of observed objects, events or categories.

All memory elements are collected in a persistent storage that relies on a Repository style architecture [14]. Module developers can access this repository through the so called Memory Interface API. This interface serves as a facade to the memory functionality and ensures consistency in terms of transactions and parallel access. Furthermore, Intrinsic Processes are closely coupled to the memory content. They work typically on the common metadata of memory elements, e.g. time information of hypotheses. The active repository contains a virtual machine for event-triggered execution of these processes within atomic transactions. Extrinsic Memory Processes, e.g. consistency validation [11], are the main computational modules of the cognitive system. In contrast to existing approaches these components are working asynchronously on memory instances and are therefore only loosely coupled which allows for flexible algorithms and architectures.

Memory Interface Since we focus on great flexibility, relational databases as persistence solution for XML data are infeasible. Instead, the native Berkeley DB XML database [6] provides the core component of the memory interface. In contrast to classical DBMS this embedded DB library concentrates on core features like support for transactions and multi-threaded environments. Since it is no complete server application and exhibits a small footprint, this library provides an efficient basis for the implementation of our active memory. On this basis our realisation of the active memory not only allows for plain XML documents, but also for...
mixed XML and binary data like cropped image patches that are not stored in XML but can still be referenced as memory content. Through the use of XPath [3] statements developers of memory processes can easily specify queries in a declarative manner.

Additionally, the basic insert, select and update methods as well as the memory events are connected to the intrinsic and extrinsic memory processes. Through a subscription model for distributed event listeners and the processes which can be attached to trigger listeners the memory indeed becomes active. Whenever a basic method is called, the memory instance notifies all processes that have been registered for that kind of action and the type of involved memory element. The execution of registered intrinsic memory processes is controlled by a runtime environment where processes like forgetting or information fusion can be realised [18]. Finally, the memory interface encapsulates complexity and serves as an API for inserting and accessing memory content or register trigger listeners.

XML enabled Communication Framework One very important part of a framework for distributed cognitive vision systems is the ability to distribute modules and corresponding algorithms across different machines in order to guarantee fast system reactions. However, most cognitive vision researchers are no middleware experts prohibiting the native use of, e.g., CORBA. Thus we developed XCF, enabling researchers to easily build distributed object oriented systems that are fast enough for online demonstrators.

As technical basis of our framework the Internet Communication Engine (ICE) [12] was chosen. ICE offers similar functionality as CORBA, but with a much more lightweight approach. The ICE core manages communication tasks using an efficient protocol, provides a flexible mechanism for multi-threaded servers and additional functionality that supports scalability. Similar to CORBA, ICE also relies on pre-compiled proxy objects. Unlike CORBA, there is no explicit dynamic invocation interface in ICE.

The XCF API encapsulates the middleware complexity and the additional features introduced by our approach. XCF itself uses ICE for low level communication tasks and is written in C++. It features a pattern based design as well as communication semantics like publisher-subscriber that allow one-to-many communication, (non-)blocking remote method invocation with query and send semantics as well as event channels. All XCF objects and exposed methods can be dynamically registered at runtime.

XCF conforms at least to the following transparency levels which are important for communication frameworks [16]: Access transparency is provided by the XCF core, where the implemented dispatcher service realises location transparency. Concurrent access of multiple clients on one server is transparently handled by a specific worker thread pool. Additionally, the use of monitor threads provides migration transparency for computational modules.

Similar to the memory interface data storage described previously, data exchange between different modules is based on XML. Since interfaces are specified using XML schema, runtime type safety can be ensured. Additionally, high performance native datatype transmission for large binary data is possible that can be referenced from XML messages. System introspection is directly supported that helps in debugging and monitoring a running distributed system.

4. Practical Experiences

With a first implementation of the memory infrastructure framework, our integrated demonstrator was developed. Experience shows that the use of XML helped in defining datatypes which are suitable for every involved project partner. XML objects as native datatypes allow for a flexible
schema evolution and seperation of concerns within a system architecture. Additionally, generic software modules can much easier be realised within this concept.

Figure 3 shows the overall throughput of the different communication patterns when transmitting XML object hypotheses across a network by blocking RMIs with send and query semantics as well as the results for a data stream that is received by three subscribers in parallel. The latency amounts to approx. 0.2–0.3ms. Optional schema validation takes ~1ms for the object hypothesis shown in Fig. 2. Evaluation of the native datatype transmission shows that the performance is also sufficient for image data streams.

A detailed evaluation with respect to the performance characteristics of memory instances can be found in [18].

Concerning the goals mentioned in section 2 we think that Usability of the memory infrastructure is high. This emerges from two design decisions: On the one hand the component developer only has to deal with a small number of classes in a simple API and on the other hand many technologies used are standards based. Using XML technologies throughout the whole framework, we meet the Flexibility requirement. This results in changeability, easier adaptation and integration of modules. Additionally, openness of distributed memory instances allows developers to retrieve information in a standard fashion using declarative queries.

Low Coupling is reached through combination of active memory instances and XCF. The memory instances itself decouple the memory processes and serve as an information mediator while the XCF framework provides location and access transparency for components. This leads to easy exchange of components and high robustness against component failure. Both techniques enable Rapid Prototyping.

Debugging and Evaluation is supported by simulation of components using XCF for replaying recorded memory data with a so called module simulator. When the memory content has time information associated (as shown in Fig. 2) whole architectural layers can be replaced by this simulation tool as if they were online available. Development and evaluation of different algorithms or system configurations on comparable data has been much easier with this feature.

5. Conclusion and Outlook

The successful integration of our demonstrator and the online ability of the resulting system architecture in our collaborative cognitive vision project VAMPIRE is a first proof of concept for the proposed framework. Nevertheless we are currently working intensively on the implementation of event channels and the validation of interface specifications. Altogether we are confident that the steps we have taken so far lead in the right direction towards a generic integration framework for cognitive vision systems that focusses on high usability, memory and distribution.

Acknowledgements

This work was supported by the VAMPIRE project of the European Union (IST-2001-34401).

References