Integration Frameworks for Large Scale Cognitive Vision Systems – An Evaluative Study

S. Wrede, C. Bauckhage and G. Sagerer
Bielefeld University, Faculty of Technology,
P.O. Box 100131, 33501 Bielefeld, Germany
{swrede,cbauckha}@techfak.uni-bielefeld.de

W. Ponweiser and M. Vincze
Vienna University of Technology, ACIN
Gusshausstrasse 27-29/E376, 1040 Wien, Austria
{ponweiser,vincze}@acin.tuwien.ac.at

Abstract

Owing to the ever growing complexity of present day computer vision systems, system architecture has become an emerging topic in vision research. Systems that integrate numerous modules and algorithms of different I/O and time scale behavior require sound and reliable concepts for interprocess communication. Consequently, topics and methods known from software and systems engineering are becoming increasingly important. Especially framework technologies for system integration are required. This contribution results from a cooperation between two multinational projects on cognitive vision. It discusses functional and non-functional requirements in cognitive vision and compares and assesses existing solutions.

1. Motivation and Background

Apart from traditional research on algorithms for low and high level computer vision, the last 20 years have seen considerable efforts on integrating such algorithms into larger systems. During the last decade, vision system research further advanced to the ambition of developing *cognitive computer vision systems* (CVS) [2, 4]. These are systems which not only involve computer vision but also employ machine learning and resoning in order to extend prior knowledge or to verify the consistency of results as well as to manage several computational modules [6].

Obviously, if fast online capabilities for cognitive vision are desired, the complexity of the involved subtasks requires to distribute computations over several machines. Looking at the design of distributed systems, however, often reveals additional degrees of complexity. Distributed architectures usually consist of many components, connectors, patterns and various rules for the connections among building blocks [15]. In order to ensure architectural soundness, integration frameworks thus are mandatory.

Throughout the past decade, several frameworks for vision systems have been proposed [17, 7, 11, 5, 9, 10]. All these frameworks were tailored to certain project specific requirements and thus are of limited generality. However, there are common needs in traditional as well as in cognitive computer vision that can easily be identified (e.g., efficient handling of image data). These of course provide general criteria that can guide a comparison of frameworks.

Practical experience shows that if large scale vision systems are being implemented (in the worst but nowadays somewhat usual case by teams of researchers from different institutes located in different countries), one has to consider not only domain specific requirements but always will face problems of *programming in the large*. Therefore, non-functional requirements have to be taken into account, too. Common examples encountered in practice are reusability, scalability, or transparency. As a consequence, techniques and approaches from software engineering should be cared for when designing vision systems.

Due to the increasing importance of this topic and since only few and less comparative reports [3, 13] are available, an evaluative study was recently requested by a european research network. As it is carried out in collaboration among several research groups with a background in integrated vision systems, experience with various frameworks gathered from different projects and applications is being accounted for. Methodologies and results will be reported here. In order to explain our approach, the next section will outline functional and non functional requirements for (cognitive) vision integration frameworks. The resulting classification scheme shall be discussed in section 3 and exemplary results will be presented afterwards. Finally, a conclusion will close this contribution.

2. Identification of Requirements

Based on experience with integrated systems (cf. e.g. [1, 8, 11]) this section reviews functional and non-functional requirements that have been identified as the most important for the construction of such systems.

0-7695-2128-2/04 $20.00 (C) 2004 IEEE

2.1. Functional Requirements

Cognitive vision systems have to process image data and abstract representations derived therefrom. Hence, *Support for User Defined Datatypes* and *Suitability for Binary Data Transfer* are required. Among the core features of CVS presented in the previous section are reasoning and learning. The reasoning property is a pure functional one. There is no specific requirement for the framework resulting thereof.

Learning and adaption in a CVS requires some kind of memory so *Data Management Facilities* must be provided. Since adaption and attention control induce dynamics into the system, *Dynamic (Re-)Configuration* of architectural building blocks is desired. According to Christensen [2], CVS should be embodied and consequently requires at least a hybrid architecture (*Independence of Architectural Styles*) to enable sensory bottom-up as well as actuatory top-down processing. In terms of *Programmatic Coordination*, a CVS can thus be viewed as a reactive system whose behavior is constrained by the task.

Following the argument of Christensen and Crowley [3] we believe that *Evaluation Support* is essential, e.g. the possibility to experiment with recorded data flow.

As motivated above, features for the construction of parallel distributed software systems are especially important. Distributed engineering attributes capture this requirements. First of all this includes the *Level of Transparency* [16] provided by the integration framework. As far as possible we require *(Explicit) Interface Specifications* because experience shows that this reduces the risk of interface mismatches. Possible sources for errors are further reduced if the framework allows for *(Active) System Introspection* [3] and *Error / Exception Handling*. These features directly support the *Robustness* of the framework.

Furthermore, the *Framework Performance* and *Scalability* are of course important integration framework features but since they always depend on specific application scenarios, they cannot be rated generally.

2.2. Non-Functional Requirements

In system integration, non-functional requirements are often neglected and considered to be less important. However, experience reveals that they are crucial for project success. The following demands either represent best practices resulting from previous projects of our groups or describe insights gathered from software engineering research.

As integration is a complex task and requires profound knowledge of middleware like, e.g. CORBA, many vision researchers admittedly concentrate on the development of single modules. Consequently, the goal of frequent system integrations could be achieved more often if module developers were able to easily intergrate their components. Thus, *Usability* and simplicity in terms of a flat *Learning Curve* are basic essentials of an integration framework.

In long-term research projects aiming at integrated demonstrators, specifications or datatypes may change frequently. Since the resulting modules should be flexible in terms of changeability, *Ease of Modification* is also important. For example, the impact of interface changes on an existing system architecture should be minimal to avoid a versioning problem as known from CORBA [14].

Another requirement directly related to usability and ease of modification is *Rapid Prototyping*. The ability to use a framework not only for prototyping of single modules but also of a whole system supports iterative development. Thus wrong directions in system evolution can more easily be identified, especially if integration is performed on a regular basis starting at an early project stage.

As reusability of exisiting software modules increases productivity, the *Integration of Legacy Code* or legacy modules, e.g. for object recognition, must be enabled. Ways of integrating existing modules broadly range from dynamically loadable plugins or object-oriented wrapper facades for exisiting C libraries to a closed framework that does not allow any external dependencies due to constraints enforcing a parallel control architecture.

Framework Sustainability and *Framework Maturity* provide strategic criteria if one has to choose between different framework alternatives. Sustainability characterizes the current and expected project activity. In either case, closed or open source framework development, an active development community will ensure support and further bug fixing. The maturity level of a framework characterizes in how many projects an integration software has been used and which level of stability it has reached.

Finally, the *Available Documentation* is of invaluable importance. Especially in frameworks resulting from research projects this feature often is neglected. By documentation we do not mean only an appendix in a corresponding PhD thesis. It rather necessitates an up-to-date reference manual and a complete API documentation. For successful reuse of an existing technology this is mission critical.

2.3. Additional Features

So far, we only considered criteria for which a qualitative rating is possible. Some features, however, will improve the framework's applicability in system integration but cannot be benchmarked. These are called additional features.

The *License Type*, *Supported Architectures* and *Operating System(s)* as well as available *Language Bindings* are criteria that help assessing the suitability of a framework for a specific purpose and environment.

The assessment of a communication framework is completed by considering which *Communication Patterns* (e.g. streams, event channels, etc.) it provides and by regarding its *External Dependencies* and *Standards Compliance*.

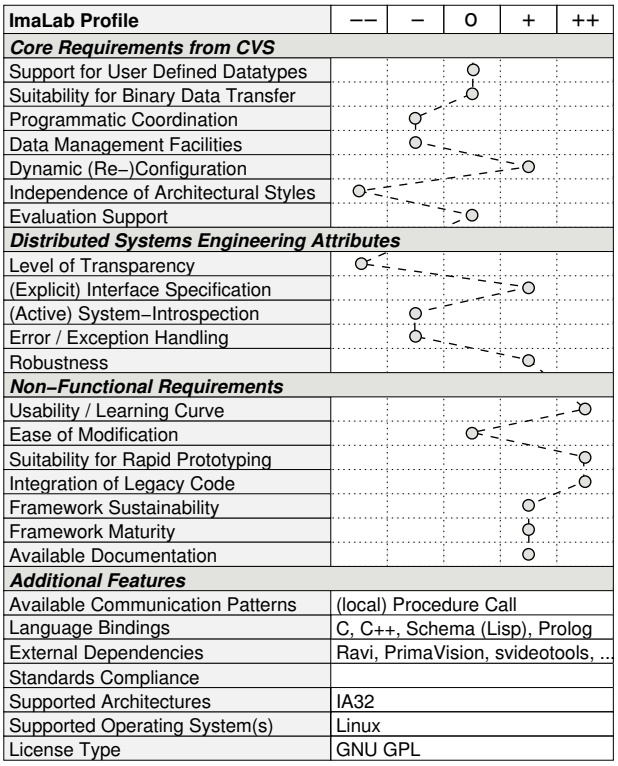


Figure 1. Graphical evaluation scheme.

3. Evaluation Scheme

Dealing with framework assessment, a common evaluation scheme did not exist so far. We thus propose a scheme based on the requirements identified above. For each criterion, we apply a five step scale abstractly denoted as {- -, -, o, +, ++}. As an example for the meaning of these units, consider the assessment of the feature *Support for user defined datatypes*: In a framework with only a fixed set of data types that can be exchanged between architectural building blocks, support for user defined data structures is almost *impossible* (- -). A *difficult* (-) rating would be assigned, if for communication purposes artificial data types must be decomposed into the native datatypes provided by the framework. If the required coding effort is equivalent to a definition of a class in OOP including serialisation, this would be a reason to rate the framework as being *neutral* (o) regarding the data structure support criterion. If standard serialisation methods are provided, the feature is *supported* (+). An *automatic* (++) case would require no explicit communication related coding at all.

Details on how the other functional and non-functional criteria are mapped to the five step scale can be found in [12]. A caveat is that especially for the non-functional criteria the assessment is based on the subjective experiences of the authors. Thus, although our assessment applies software engineering principles as far as possible, we would very much appreciate discussions on these important group of criteria for integration frameworks.

4. Results

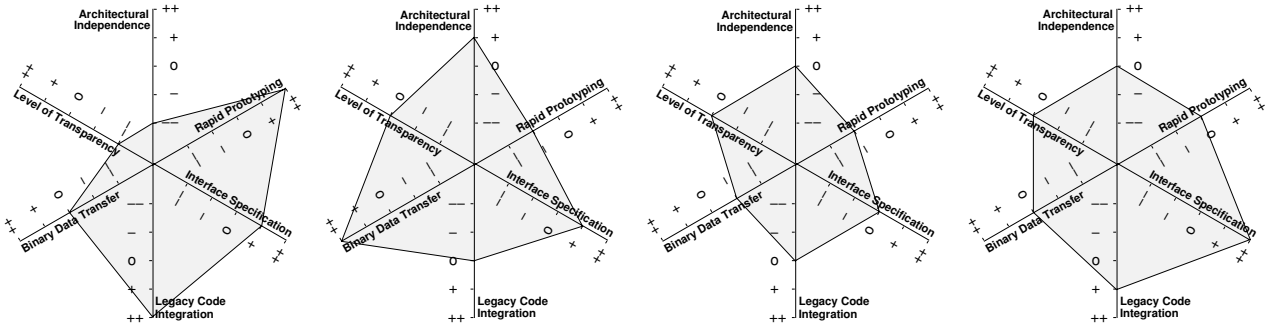
One goal of our evaluation is to identify frameworks that support the development of cognitive vision system. As an example, Fig. 1 shows the detailed results for the ImaLab framework. Complete results for the other frameworks considered in our study can be found in [12]. Based on these results, individual frameworks can be assigned to four different classes and the strengths and weaknesses of these classes can be compared in general. Typical representatives of these classes and their most discriminative features are shown in Fig. 2. Although *Vision Libraries* like VXL, IUE or openCV can serve as a basis for the implementation of computer vision modules, they are not suited for the integration of a large scale vision system. Consequently, mere library collections are not considered here.

Visual Image Processing Environments like ImaLab first of all provide facilities to quickly create pipe-and-filter style vision systems allowing for for rapid prototyping. Often they come along with easy to use graphical user interfaces. However, this class of frameworks normally lacks support for distributed architectures which prevents them from being used for large scale vision systems.

Classical *Middleware* approaches constitute another group; a typical example is the CORBA implementation TAO. On the one hand, middleware approaches are very generic and powerful. On the other hand they suffer from this genericity because it complicates their use. Moreover, they do not support requirements specific for the cognitive vision domain. Frameworks like SmartSoft or OROCOS are designed for the *Robotics Domain*. Since modern robotics also deals with machine vision, the frameworks of this group fulfill some of the requirements identified in section 2. They often allow for distribution and provide transparency. However, in robotics, real-time constraints are a major topic which introduces additional complexity in using these frameworks. Furthermore, their integration facilities are often closely coupled to components for robot control.

The last class identified in our study consists of two examples especially designed as *Cognitive Vision System Frameworks*: While XCF [17] focuses on data management facilities, distribution and rapid prototyping, zwork [11] mainly deals with the programmatic coordination and dynamic reconfiguration required in dealing with the control aspect of cognitive vision systems.

As it can be seen from Fig. 2, each group of frameworks concentrates on different aspects in system integration. Thus, system engineers will have to carefully identify their needs and project requirements and correspondingly



(a) ImaLab (b) The ACE ORB (c) SmartSoft (d) zwork

Figure 2. Comparison of different frameworks along their most discriminative features.

decide for a most suitable framework. A first guideline to do so is given by the proposed evaluation scheme.

5. Conclusion

This paper identified functional and non-functional requirements for the integration of cognitive computer vision systems. Taking into account experiences from previous integration projects, we presented an evaluation scheme for a fast and straightforward assessment of software frameworks for cognitive vision. Typical groups of frameworks were identified and it became apparent that by now there is no all-purpose solution for the integration of large scale vision systems. Moreover, although the development of vision systems is of increasing importance, the non-functional key criteria to successfully develop such systems seem to have been neglected so far. Consequently, we hope that this contribution will foster discussions within and across communities dealing with the design of intelligent systems.

Acknowledgements

This work has been partly supported by the EC Vision Specific Action 13-2 and the IST Projects 2001-32184 ActIPret and 2001-34401 VAMPIRE.

References

[1] C. Bauckhage, G. Fink, J. Fritsch, F. Kummert, F. Lömker, G. Sagerer, and S. Wachsmuth. An Integrated System for Cooperative Man-Machine Interaction. In *Proc. CIRA*, pages 328–333, 2001.

[2] H. Christensen. Cognitive (vision) systems. *ERCIM News*, pages 17–18, April 2003.

[3] H. I. Christensen and J. L. Crowley, editors. *Experimental Environments for Computer Vision and Image Processing*. World Scientific Publishing, 1994.

[4] J. Crowley and H. Christensen, editors. *Vision as Process*. Springer, 1995.

[5] B. Draper, G. Kutlu, E. Riseman, and A. Hanson. ISR3: Communication and Data Storage for an Unmanned Ground Vehicle. In *Proc. ICPR*, volume I, pages 833–836, 1994.

[6] European Research Network for Cognitive Computer Vision Systems, 2003. http://www.ecvision.info.

[7] G. Fink, N. Jungclaus, F. Kummert, H. Ritter, and G. Sagerer. A Distributed System for Integrated Speech and Image Understanding. In *Int. Symp. on Artificial Intelligence*, pages 117–126, 1996.

[8] J. Fritsch, M. Kleinehagenbrock, S. Lang, T. Plötz, G. A. Fink, and G. Sagerer. Multi-modal anchoring for human-robot-interaction. *Robotics and Autonomous Systems*, 43(2–3):133–147, 2003.

[9] K. Konstantinides and J. R. Rasure. The Khoros Software Development Environment For Image And Signal Processing. *IEEE Trans. on Image Processing*, 3(3):243–252, 1994.

[10] C. Lindblad, D. Wetherall, and D. L. Tennenhouse. The VuSystem: A Programming System for Visual Processing of Digital Video. In *ACM Multimedia*, pages 307–314, 1994.

[11] W. Ponweiser, G. Umgeher, and M. Vincze. A Reusable Dynamic Framework for Cognitive Vision Systems. In *Workshop on Computer Vision System Control Architectures*, 2003.

[12] W. Ponweiser, M. Vincze, S. Wrede, and C. Bauckhage. Overview of Software Frameworks for Use in Cognitive Vision Approaches. Technical report, ECVision Specific Action 13-2, 2004. http://www.ecvision.info.

[13] A. Rares, M. Reinders, and E. Hendriks. Mapping Image Analysis Problems on Multi-Agent-Systems. Technical report, TU Delft, November 1999.

[14] D. C. Schmidt and S. Vinoski. Object interconnections: CORBA and XML, part 1: Versioning, 2003. http://www.cs.wustl.edu/ schmidt/report-doc.html.

[15] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[16] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[17] S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer. An XML based Framework for Cognitive Vision Architectures. In *Proc. ICPR*, 2004.