A Reusable Dynamic Framework for Cognitive Vision Systems

Wolfgang Ponweiser[1], Gerald Umgeher[2], Markus Vincze[1]
[1] Automation and Control Institute, Vienna University of Technology, Vienna, Austria
[2] Profactor Productionstechnik GmbH, Steyr, Austria

1 Introduction

This workshop confirms the increasing demand on multi-functional vision systems. Applications in the area of mobile robots, artificial surveillance systems as well as cognitive systems must be able to interpret the observed scene. These systems pose new challenges on the sensor system:

- The system is *on-line* with the observed scene. Therefore the system has to *react* in the temporal order of activities observed and with appropriate time delay (*data-driven*).
- The system has to solve several tasks *concurrently*, tasks must be executed on *restricted resources*, tasks are *complex* in terms of the methods and combinations applied and require *control and integration* of (mostly) *probabilistic* methods to fulfil the demands of cognitive processes. Therefore the system has to be *task-driven*.
- Additionally all practical aspects of software engineering such as *scalability*, *ease of use* and *software reuse* have to be taken into account.

To comply with all these requirements two measures are proposed. The underlaying software concept is founded on a component based approach [2], which is faster than classical agent systems to guarantee reactivity and enables easier distribution than conventional object oriented approaches whose scalability ends at the one computer border. To fulfil all of the above requirements the "Service Principle" is introduced, which is derived from the CORBAservices ([8]). It is based on a "Yellow Pages" dynamic look-up directory that makes it possible to select on-line the best available service according to a formalised performance characterisation. The resulting architecture is therefore *dynamically* adapted to suit optimally the current required tasks and the changing environment. Section 2 presents details and the framework test environment and a first experimental evaluation are presented in Section 3

2 A Framework based on the Service Principle

To enable easy component distribution in a task-driven environment a component has to provide one or more interfaces to make its functionality accessible. Using this interface other components can initiate one or more tasks (= services). The components itself can be regarded as black boxes with their interfaces describing all their abilities. The interfaces are called service definitions and the component acts as the service provider. This achieves modularity and, hence, enables reuse, ease of use and scalability.

Any component can use the services of other components. In this case the component acts as the service requester. In summary, every component presents its abilities by providing services and every component makes use of the abilities of other components by requesting services.

The service descriptions form the complete interface for a component and hide the implementation. A special component, called the service list (implemented based on the CORBA trader service), provides the mechanism for any components to find out about the services that other components provide. This mechanism operates as follows (see Fig. 1): Every component registers its services offered in this service list. Using the service list as a yellow pages directory for services, a component can find on-line the services it requires and can then establish communications with the relevant component(s).

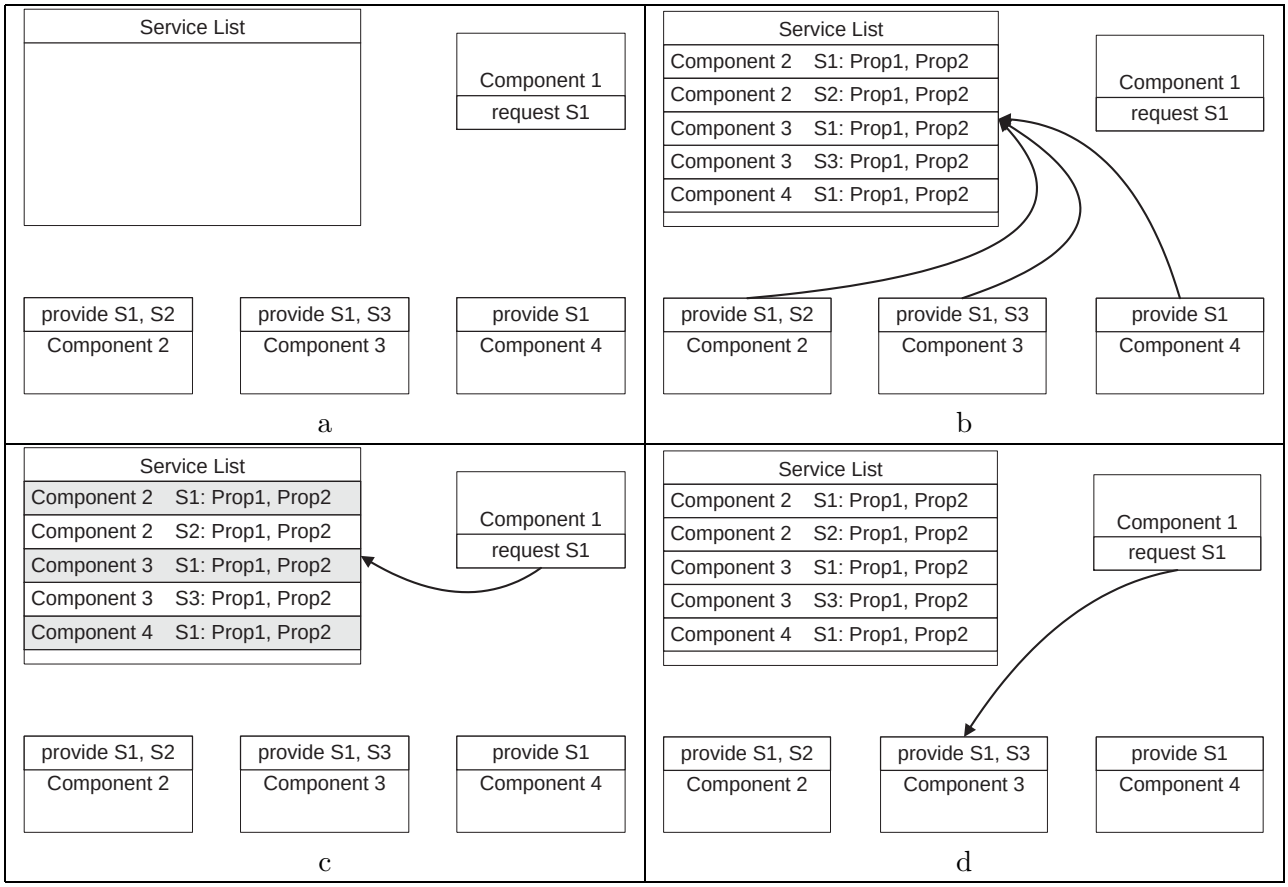

Figure 1: a) The system before any services are offered. b) The service providers offered their services with current properties. c) The service requester asks for services with specific properties. d) The service requester establishes a link to the selected service provider.

A key feature of this approach is the capability to select on-line between competing services offered of similar type based on service properties. Each entry in the service list contains a formalised characterisation of the service offered. The characterisation includes items such as the expected success rate (like points on a Receiver Operating Characteristic [9]), spacial accuracy as well as timing and resource requirements. These entries can be updated dynamically by the components to adapt to the present situation of the scene and the system context. This mechanism enables the selection of the component that provides the best service instantiation at a given instance of system operation.

A component can select required sub-tasks

- between different service types: There is a fixed number of predefined services which enables the component to select an appropriate combination of sub-tasks.
- between different service offers: The service list contains an entry for every component providing this service. The selection can than rely on the performance characterisation of the service offer.
- between different operation modes of a service instance: Still one service realization can be set to different modes of operation by adjusting some internal algorithm properties. Still this properties are hidden from the service requester by a sequence of property descriptions of one service in the service list.

The result is a distributed component set-up. All links between components are made on-line according to the best fit of the offered service characterisation to the current task. With the ability to change the characterisation on-line a perfect adaptation to the current system and environment state can be achieved.

2.1 Resource Management

Typical resources, which can be consumed, are CPU-processing time, memory, realisable views (of active camera systems) and other required hardware. In all these examples the resources are restricted and it is necessary for a generic system to coordinate and optimise their usage.

To coordinate these types of resources the system is using the ability to dynamically change properties of the service description entries in the Service List. Hence a component offering a service in the service list is able to adapt its description (performance, accuracy, etc., see above) according to the available amount of required resources. (E.g., a service may not be useable anymore because all the available resources are consumed.) A crowding out principle is used to specify resource availability. Every service requester calculates a utility of a requested service using the service offer properties described above. This utility is than used to occupy the required amount of the restricted resource. If the resource is exhausted the service with the lowest utility is skipped. This method ensures a distributed resource allocation and can be easily extended by a priority mechanism.

The most restricted resource of active multi-camera systems interacting with complex scenarios are views, which - in opposite to processing power and memory - will not increase by technical improvements over the next years. Therefore, handling the resource view, which is also vital for every vision component to allow for robust and improved performance (see also [3, 4, 5]), is of main interest.

To coordinate the resource view an extension of the Service List is developed. This application is capable to select the best available sensor/vision system according to the service requested. The optimal view plus its tolerances are determined according to the regions of interest specified by the service requester, the view constraints specified by the service provider and the physical constraints of the active camera system. The camera that can provide the view point most similar to the optimal specification and which is not already assigned to a service with inconsistent view-requests is selected. After assignment the service provider is capable to control the view-point of the selected camera.

2.2 Assuring Reactive Control

The sophisticated coordination requirements are very related to distributed artificial intelligence and agent theory [6, 7]. There are many different coordination mechanisms (summarized in [1]). In an open agent-based framework, all components/agents negotiate until a system configuration is reached that fulfils the task according to some optimisation criteria. The negotiation phase is time consuming and termination within a fixed time cannot be assured. Hence, an additional measure has to be taken to obtain a reactive system that responds to events in the real world.

The solution proposes a hierarchy of components, where components of one level can only request services of components at all lower levels. The hierarchy avoids a free and possibly cyclic negotiation of services, which is the most time consuming phase and which could end, in the extreme, in a deadlock. If the levels of the hierarchy are strictly ordered with the abstraction of image data, selection of services required is not limited but restricted and unidirectional. There remains the capability to select among multiple competing services for a specific task.

3 Experiments and Conclusion

The EU project ActIPret has the goal to interpret activities of persons handling tools, e.g, placing a CD in a CD-player. Obviously several vision functionalities are necessary for this cognitive task. As already mentioned in the introduction it is a perfect example to study the requirements of such systems on the SW-architecture and to test the abilities of the framework developed.

The system integrates a User-HMI, synthesis components, pre-reasoning components, attention components (where the main vision processes are placed), controller and data-server components (see Fig. 2). In a first version one instance of all these components has been realized. To enable an optimal behaviour depending on the current vision data (data-driven) and the context (task-driven) several different components for one service based on different methods are planned.

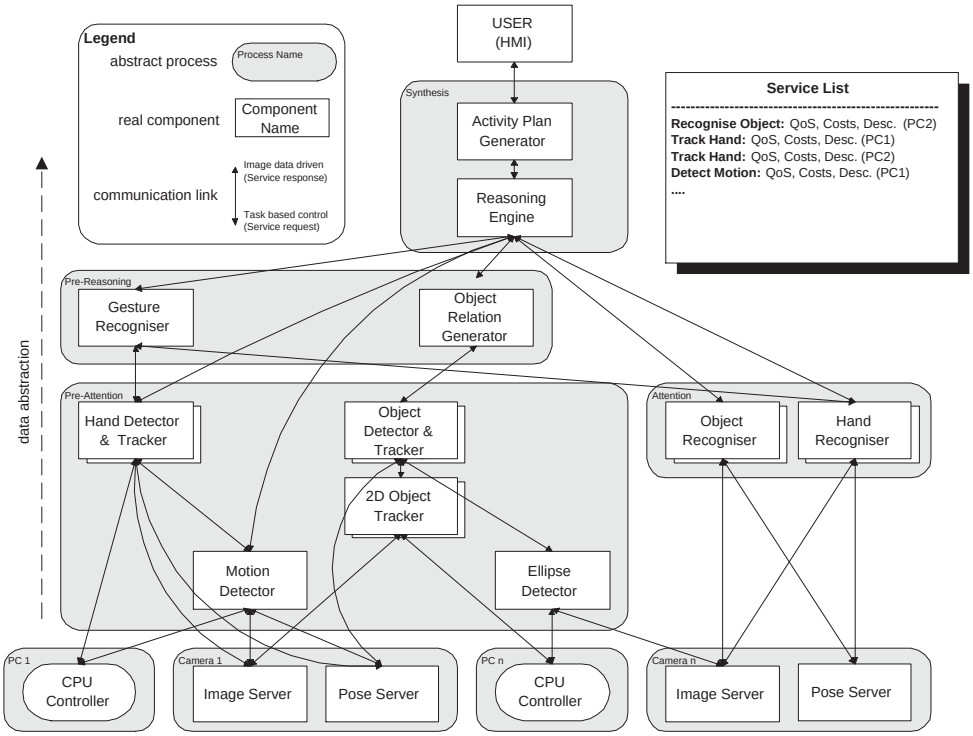

Figure 2: Example set-up for a Cognitive Vision project

The first running systems provide evidence that the SW-framework presented is able to comply to the requirements. Modules, expressed in the requirements of 'scalability' and 'software-reuse', have been developed at five partners and integrated within two days. Contradicting task-driven and reactive (data-driven) system behaviours could be combined. A generic method to handle restricted resources has been realised and tested on the resource "views available". It enables to select the services with the best views from a list of dynamic view requests.

Especially the selection of service offers enables a component programmer to optimize the system in a manageable complexity in combination with the internally used method(s). Hence, the framework is a fundament to port methods now used for specific vision or interpretation tasks (e.g., neural networks, fuzzy logic or probabilistic methods or Bayesian networks) to the software architectural level. Future work attempts to study the interaction between components and cognitive aspects such as on-line learning of components that best cooperate.

References

[1] H. Nwana, L. Lee, N. Jennings, "Coordination in Software Agent Systems"; Lecture Notes in Artificial Intelligence, Vol. 1198, January 1997.

[2] Szyperski C., Pfister C.; Workshop on Component-Oriented Programming, Summary; In Mhlhuser M. (ed.) Special Issues in Object-Oriented Programming - ECOOP96 Workshop Reader. dpunkt Verlag, Heidelberg

[3] J.A. Fayman, O. Sudarsky, and E. Rivlin: "Zoom Tracking", In Proc. IEEE Int. Conf. on Robotics and Automation (ICRA '98), pages 2783 - 2789, May 1998.

[4] T. Lindeberg, K. Brunnstrm and J.O. Eklundh: "Active Detection and Classifiaction of Junctions by Foveating with a Head-Eye System Guided by Scale-Space Primal Sketch". In ECCV 92, page 701-709, May 1992.

[5] C. Capurro, F. Panerai and G. Sandini: "Dynamic Vergence", IEEE IROS 96, pages 1241-1249.

[6] M. Wooldridge, N. Jennings, "Intelligent Agents: Theory and Practice"; Knowledge Engineering Review 10(2), 1995.

[7] Y. Shoham, "What we talk about when we talk about software agents"; IEEE Intelligent Systems 14(2), pp 28-31, March 1999.

[8] Object Management Group (OMG), "CORBAservices: Common Object Service Specification"; March 1995

[9] T. Kanungo and R. Haralick: "Receiver Operating Curves and Optimal Bayesian Operating Points"; In Proc. IEEE Int. Conf. on Image Processing, pp. 256-9, Vol. 3, Washington D.C., 1995,