On a multi-agent analysis system in a multi-media information and selection system

by
Hans-Jürgen Hoffmann

FG Programmiersprachen & Übersetzer
Technische Hochschule Darmstadt
Alexanderstraße 10, D-64283 Darmstadt
Phone +49-6151-163410
hoffmann@pu.informatik.th-darmstadt.de

ABSTRACT

Multi-media applications, interactively used, require more advanced analysis techniques than traditional single-media applications and/or multi-media applications only providing multi-media presentation, resp. We propose equally entitled analysis components for user interactions to provide better task adaptation, openness for evolving analysis methods and ease of testing/validation. Background and derived provisions for incremental analysis are investigated. A multi-agent approach built from agents cooperating over a common data structure is discussed in the context of design and prototypical use of an information and selection system as can be found in electronic product catalogue applications. The common data structure is organized in form of a blackboard[*1)].

1. Application context

Most of the multi-media applications in real, productive use today are providing only a user-driven presentation style and are programmed based on a presentation and interaction library. *Electronic product catalogues* ([7], [8]) as in work today for multi-media product information presentation and, possibly, product selections are one instance of this kind of applications. EPKfix-DA, part of the EPKfix-project [2], follows a different approach in two aspects, one during design, the other in run-time control of prototypical presentation.

First, during design, a specification of a catalogue is interactively and incrementally built-up derived from repeated informal interviews linked to a prototypical presentation of what has already been specified. The outcome of an interview, that is, an informal textual (part of a) specification and the accompanying interactions with the prototypical presentation have to be analyzed and, by stepwise progress and in each step as far as possible, transformed into a formal specification.

Second, user interactions at run-time are multi-sensorial and event driven. In a design scenario with linkage to a prototypical specification, as considered in EPKfix-DA, the same assumption is valid. It is not only a mouse with two or three keys and/or a touch screen that is intentionally manipulated by a user/interviewer. There are independent distance sensors and other event sources, especially timing clocks, all together causing interaction events which have to be handled by the supporting system in an integrated manner.

To have an idea what an *interview* looks like, think of a meeting between a commissioner of an electronic product catalogue and a person educated in how to acquire information of all aspects of such a catalogue with the help of and by interaction with a (prototypically) presented (informal and) formal catalogue specification. The

[*1)] The project is sponsored by German Government under 01-IS 520 C

Initial cycle	**Following cycles**	**Last cycle**
- Input of (informal) outcome of first interviewing session. - **Analysis**. - Refinement of initialized design blackboard; specification of catalogue. - Output of both, informal and inferred, formal prototype of catalogue.	- Presentation of specification as far as determined (informal **and** formal parts). - Request to solve open specification details. - Interaction with presented draft of catalogue. - Input . . . - **Analysis**. - Refinement. . . - Output . . .	- Presentation . . . - Request for final confirmation. - Interaction (without changing specification). - Input . . . - **Analysis**. - Triggering of transfer of (completed) specification.

Fig 1: Cycles in the (incremental) design process

interviewer may be guided by hints given by the design tool EPKfix-DA in order to control the design process.

It is not the topic of this paper to describe the application context of EPKfix (and EPKfix-DA as an offspring thereof) in full. The group in Darmstadt concentrates its work on (i) the architectural aspects of multi-agent analysis at design time, as described in section 2 and 4, (ii) domain modeling including modeling of analysis aspects, covered in section 3, and (iii) providing an implementation of the central design data base used during the transformation process from an informal to a formal specification. As during design in the scenario sketched above a prototypical presentation of the product catalogue under design is provided, analysis has to cover aspects of multi-media interaction, too.

Fig. 1 characterizes the interviewing process for building up a specification of an electronic product catalogue. Cycles may be separated in design sessions.

2. Organization of analysis components

In simple applications (e.g. compilers) *sequential organization* of analysis components is sufficient. A scanner assembles *tokens* the structure of which is recovered by a parser building a *tree* to be decorated by *attributes* and - possibly - transformed to an optimized analysis result. There is not much difference in situations where this process is repeated in a *hierarchical organization* (e.g., communication systems with bottom-level signal analysis, all forms of intermediate analysis, and an application-oriented top-level unit analysis).This organization form is feasible only for linear input. Incrementality cannot be achieved.

Object-oriented approaches allow an encapsulated structure of *data items* - or *objects* - (i.e., tokens, tree nodes, attributes, signals, ..., units) and their individual behavior supporting an analysis process in a form more flexible in object identification and in organization of analysis steps; incremental approaches are feasible. This is state-of-the-art in non-interactive and interactive situations as typically found in the so-called graphical interfaces of computers with a two-dimensional screen for direct manipulation. Object-orientation offers a wide scope of unstructured actions in objects; a scope; however, not necessarily well adapted to specific analysis tasks occurring in multi-media environments and under independently timed events as a clear process structure is missing. Class hierarchies and inheritance facilities introduce only static structuring, not dynamic process structuring. This is a drawback. Introduction of an additional analysis technique possibly requires modifications to a lot of class descriptions and/or inheritance paths without giving guidance for modifications in the process structures; so openness is not achieved as desired for our application context.

Blackboard approaches ([4], [10]), maintaining advantages if combined with object-orientation in objects to be analyzed, extract the analysis processes and their dynamic interrelations from individual objects and assign them to equally entitled analysis components, or *agents*, viewing and operating on objects in a task-specific, open manner. The agents are best specified in an individual rule-based form. The necessary coordination between agents is provided by a general, not application-dependent mechanism of access synchronization and object evolution. This approach is especially appropriate for event-driven analysis requirements and for incremental analysis as it allows repeated rule application without a predefined ordering.

The analysis system in EPKfix-DA is organized as shown in fig. 2.

There are agents (i) providing different mappings to the outside environment, viz., the interactive *Specification Editor*, the (partially integrated) *Prototyping Server*, the *Test/validation Requester*, the *Documentation Provider*, the (internal*) Object Viewer*; there are agents (ii)providing specific analysis methods, e.g., *Item Identification* derived

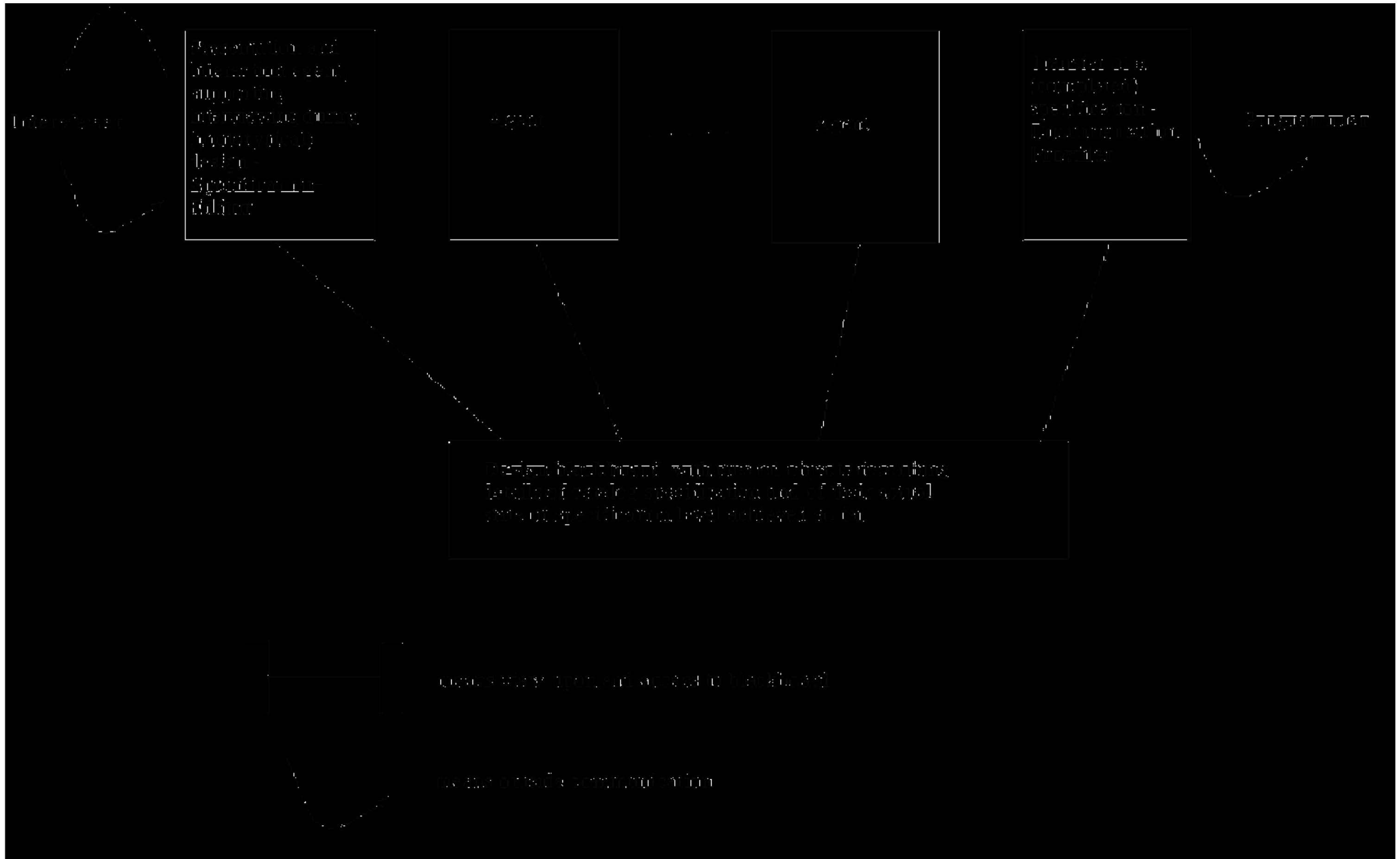
Interviewer
Presentation and Interaction agent, supporting interviewing during (prototypical) design - Specification-Editor
Agent
Agent
Translation of a (completed) specification - Documentation Producer
Programmer
Design blackboard with created objects describing details of catalog specification and of their actual state of specification level achieved so far
means view upon and access to blackboard
means outside communication

Fig. 2: Organization of analysis components in EPK-fix-DA (just a few agents shown)

from user/interviewer interactions, *Catchword Extractor* and *Catchword Linker* in informal specification steps, classical *Token* and *Structure Analyzers* in formal specification steps, *Consistency* and *Completeness Checkers*; and (iii) there are „creative“ agents, like an *Attribute Inference* agent or a *Specification Process Controller*. This list is not complete. Openness in the analysis architecture, one of our design goals, is provided already in not trying to have an a priori completed list of analysis components.

Characteristics of some agents are the following:

Specification Editor/Prototyping Server

All communication about (parts of) the specification is treated by the *Specification Editor*. It is partially integrated with the *Prototyping Server* in so far as it presents the (parts of the) informal and/or formal specification already refined allowing interactions in the considered interviewing cycle with the presented (draft of the) catalogue for further, refining specification details. The editor creates new objects - in general in an intermediate level (see section 3) -
and, as a consequence, invokes other analysis agents. The *Prototyping Server* is, essentially, an interpreter of the already available parts of the specification which are formal.

Item Identification

Items are elementary building blocks of a catalogue. *Item Identification* maps feedback from (formal specification parts in) the prototypical presentation to objects created on the blackboard.

Catchword Extractor/Catchword Linker

The task of the *Catchword Extractor* agent is to recognize catchwords in the informal feedback coming from the specification editor. The *Catchword Linker* resolves sequences of recognized catchwords for appropriate refinement steps in the object structure representing the catalogue specifications. References [5] and [1] identify the range of progress in catchword extraction which we have in mind.

Consistency Checker

Creation and refinement of specification objects (see section 3), especially on the highest, final level, are constrained by consistency rules. The *Consistency Checker* is a (simple-minded) maintenance agent for constraints; *simple-minded* in the sense of not trying to solve arising conflicts, but only diagnosing them. Constraints being satisfied are triggering object creation and/or attribute refinement. So, *Consistency Checker* shows (some) characteristics of a „creative agent“, too.

Attribute Inference

Attributes, i.e., the constituents of application-oriented objects (see section 3) forming and characterizing a product catalogue under design are the material for an inferring activity of the *Attribute Inference* agent with the goal to complete the specification of the catalogue without interaction. Its main directive is to perform the transformation from an informal specification to a formal one. So, appropriate initialization of attributes is one of its tasks. Another important contribution to the transformation process is abstraction from in the first instance unrelated application-oriented objects to condensed, more abstract objects.

Specification Process Controller

The *Specification Process Controller* realizes strategies for querying/highlighting incomplete specifications of a product catalogue. It is triggered (i) by observing inferred attributes, (ii) by events originated in interactions with the prototype specification, and (iii) by rules covered in scripts for handling groups of such controlling steps.

All agents are able to view objects laid down in the blackboard and to follow hyper-links established between objects when they are created. Thus, *viewing* and *navigation behavior* has to be provided by all objects. Agents are allowed (depending on the specific tasks assigned to them) to modify objects. Besides access control there are no locks in the objects hindering modification and evolution; access to objects is open; in agent design a careful investigation of required access strategies has to be done and specifically fixed by *locking rules*. *Analysis results* are also laid down in special objects on the blackboard for attribute inference and/or specification process control.

To provide openness during design of a new (or modified) agent specialization in the object class hierarchy of the blackboard objects are allowed to provide means to handle the specific task by the considered agents and its „co-workers“.

Our approach applies analysis by specialized agents in a form similar to but extending work by Sánchez et al. [9].

Requirements for structuring the working domain of EPKfix-DA are derived from two sources: *Application context* as described in section 1 and *Analysis organization* as described in section 2.

Details of the first source, *application context*, are given in [3].

For *analysis organization* an initial domain structure is given in fig. 3. It is described in form of a class hierarchy prepared for multiple inheritance into objects of the application domain.

- SpecificationObject (abstract)
    - InformalSpecificationObject
    - IntermediateSpecificationObject (abstract)
        - IntermediateSpecificationObjectLevel1
        - IntermediateSpecificationObjectLevel2
        - .
        - .
        - .
        - IntermediateSpecificationObjectLevel*n*[*2)]
    - FinalSpecificationObject
- CatchwordSentence (abstract)
    - SingleCatchword
    - OpenCatchwordSentence
    - ClosedCatchwordSentence
- ItemIdentification (abstract)
    - PresumedItemIdentification
    - FinalItemIdentification
- ProcessControlObject (abstract)
    - SequentializedControlObject
    - PrioritizedControlObject
    - ProcessControlObjectGroup

Fig. 3: Domain structure for analysis organization in EPKfix-DA

SpecificationObjects have in common (i) status information held in instance variables, (ii) necessary links for viewing and interaction control (in the style of the Smalltalk MVC-approach) to support the *Specification Editor*, *Catchword Extractor*, *Item*

[*2)] It is not yet determined what number of levels of *IntermediateSpecificationObjects* will be modelled. It is anticipated that a partial ordering system has to be designed, not a total ordering as it can be conjectured by giving level numbers.

Identification, and the *Prototyping Server*, where interaction control is performed in considering established and, possibly, newly created objects of classes in the *ItemIdentification*- and *CatchwordSequence*-hierarchies, and (iii) necessary hypertext links used in browsing/navigating between application objects. Associated methods provide appropriate access.

The *Catchword Extractor* agent as well as the *Item Identification* need private analysis objects created from the corresponding class hierarchies. The objects are needed to maintain catchword sentences occurring in informal interview sentences and to organize item identification before accessing an application object; the latter organization supports interaction with formally based aspects of prototypical presentation.

Similarly, the *Specification Process Controller* needs private objects, mainly to allow sequential and hierarchically nested follow-up in process control scripts as given for process control groups. By these means it influences the interviewing process.

4. Incremental analysis

Electronic Product Catalogues very often show and offer a general scheme of presentation and ordering (see fig. 4). This is a revolving scheme of application.

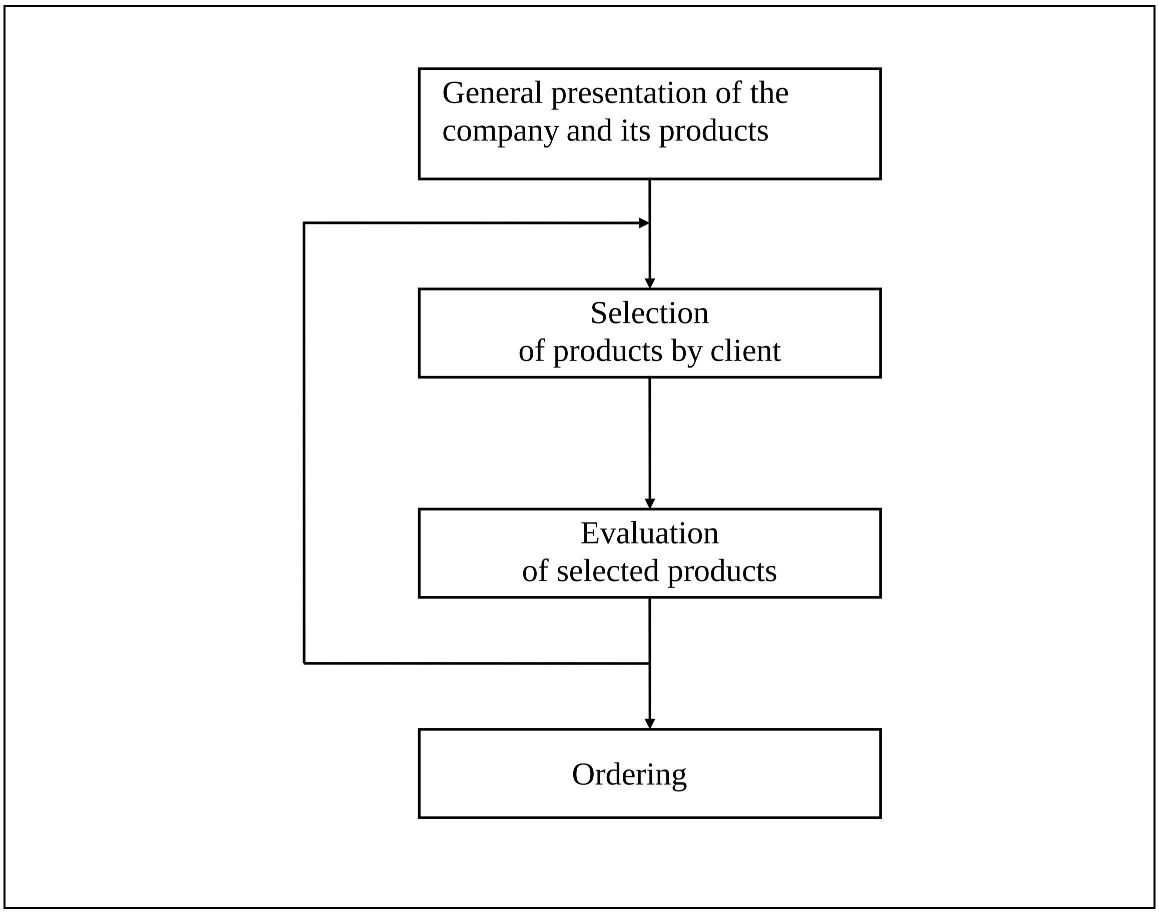
General presentation of the company and its products
Selection of products by client
Evaluation of selected products
Ordering

Fig. 4: (simplified) Scheme of presentation and ordering [6].

Specification of a catalogue during design again is not necessarily a straight-forward process. The statement becomes obvious in remembering the prototypical support of interviewing by cyclic informal input, followed by analysis, feedback in an as-much-as-possible formalized form, modification and/or supplementing until a final, complete status of the specification is achieved. The blackboard approach chosen to support the analysis steps merges smoothly in the sketched spiral scheme of design. Employing independent, equally entitled agents performing the necessary analysis brings superior flexibility and situation/task adaptation in the structure of a design process.

The spiral scheme and provisions for incremental analysis postulate each other. EPKfix-DA realizes incremental analysis by provisions in *IntermediateSpecificationObjects* and the different forms of *ProcessControlObjects*.

5. Conclusions

EPKfix-DA, a project presently in the design phase, will show a powerful, object-oriented domain structure for manifold analysis techniques in the environment of a blackboard system. Provisions for incremental analysis of interactions derived from an interview of a commissioner with a person acquainted to specification of electronic product catalogues are based on prototypical presentations. The process of transformation from an informal to a formal specification finally required as a basis for efficient implementation is supported by „creative“ agents operating upon objects stored in the blackboard.

This is a report about ongoing activities to be discussed inside the EPKfix-consortium and outside in a workshop on Multimedia Software Development.

6. Acknowledgment

The author thankfully acknowledges discussions with *Patrick Closhen* and *Martin Frisch* as well as support by colleagues working as project partners for *mediatec* in Nuremberg, *FORWISS* in Erlangen, *LMU München* and *TU Dresden*.

7. Literature References

[1] H. Chen et al.: Automatic concept classification of text; Comm. ACM 37 (1994) 10, 56 - 73.

[2] EPKfix-Consortium: EPKfix; Projektantrag, 1994.

[3] M. Frisch, H.-J. Hoffmann: Electronic Product Catalogues, an approach for object-oriented modelling of the application domain (submitted for a conference), 1995.

[4] V. Jagannathan et al.: Blackboard architectures and applications; Academic Press, 1989.

[5a] G. Knorz: Ein Dialogsystem für anwendungsbezogene Eingaben in einer stichwortartigen Fachsprache; Angewandte Informatik 21 (1979) 11, 495 - 503.

[5b] G. Knorz: Metasystem für ein System zur Dialogführung, das anwendungsbezogene Eingaben in einer stichwortartigen Fachsprache akzeptiert; Notizen zum Interaktiven Programmieren, Heft 2, Feb. 1979, 5 - 15.

[6] R. Lutze: internal communication in the EPKfix-Project, 1995.

[7] P. Mertens et al.: Angebotsunterstützungssysteme für Standardprodukte; Informatik Spektrum, 17 (1994) 5, 291 - 301.

[8] NN: Hitparade - ganz individuell; IBM Nachrichten, 45 (1995) 321, 18 - 21.

[9] J.A. Sánchez et al.: HyperActive, extending an open hypermedia architecture to support agency; ACM Trans. Computer-Human Interaction 1 (1994) 4, 357 - 382.

[10a] I. Reischitz, G. Vogt: DIADES-II, an user interface design system with assessment of design- based on a blackboard architecture. In G.E. Lasker (ed.): Advances in Support Systems Research;
Intl. Inst. for Systems Research, 1990, 183 - 188.

[10b] G. Vogt: Über Struktur und Aufbau eines Entwurfssystems für Benutzungs-schnittstellen; Dissertation, TH Darmstadt, 1994.