Minutes of our one-day workshop July 2, 1998

7/5/98, Dorothea Beringer

The workshop took place at Gio's house and was attended by Gio, Ron, Woody, Laurence and Dorothea.

1 Agenda

1) Where we are now:
- State of infrastructure: overview language, protocols, CORBA-wrapper, RMI-wrapper, compiler, repository, scheduler
- Existing demos
- Conceptual work on CHAIMS
- Presentation of our work
- Missing parts, flaws in current infrastructure and concepts

2) Where to go from now:
- Objectives long-term
- Objectives summer (autumn) term
- Architecture of CHAIMS system: possible/necessary changes
- Further things to do in upcoming months

3) Prioritizing work items

4) Examples for CHAIMS
- Scenarios for CHAIMS
- New suites of megamodules

5) Language issues

2 Documents

Documents we used in the discussion were:

- List of open points as found in the second part of the document "/Chaims/Planning/Tasks_1998_June".
- Slides (architecture, language, future plans, objectives) of /CHAIMS/slides/slide_collection_db.ppt.
- Printouts from protocol definitions etc.
- Hand-written notes and slides are in my office.

3 Summary results

- We should all know now where we stand, what we have, what we do not have yet, what our current concepts and terms are.
- We had a lengthy discussion about the CHAIMS-language, especially concerning what kind of types are allowed, how we want to use helper-modules like math and IO, what SETATTRIBUTES is for, and how pre-setting of parameters in the megamodule should work.
- Another major point were the CHAIMS objectives, which we slightly edited. Certain plans like Scheduler have been pushed off in favor of getting first a streamlined concept of our language and implementing these changes.

4 Detailed minutes

CHAIMS-language: Global parameters in megamodules, SETATTRIBUTES, GETPARAM

Which of the following concepts do we want in CHAIMS?

1. The parameters of methods (values in the name-value-lists of INVOKE) are **not** the same as global megamodule parameters (set by SETATTRIBUTE)
   ==> Various methods of the same megamodule may use the same parameter name for different kind of parameters (e.g. "route", meaning once one city-pair, and once a list of city-pairs).
   ==> We need both primitives GETPARAM as well as GETATTRIBUTES (GETPARAM takes as first argument the name of the method for which a specific parameter is asked for)
2. The parameters of methods and global parameters **are the same thing**. A megamodule just has parameters that are used as parameters for method invocations and that can be set either by SETATTRIBUTE or by INVOKE. All parameters have a default value that is used as long no client-specific default is set by SETATTRIBUTE, or method-specific value is given by INVOKE (see also notes on slide)
   ==> All parameters must have a unique name when they differ in type or when SETATTRIBUTE should refer to them distinctively
   ==> When a method is started with INVOKE, then the order of precedence is the following for the parameters of that method: 1) if INVOKE contains this parameter in its name-value list, that value is taken; 2) the client-specific value preset by SETATTRIBUTES is taken; 3) the general default value hard-coded in the megamodule is taken.
   ==> The primitive GETPARAM is no longer needed in the CHAIMS-language. SET-/GETATTRIBUTES is used instead also for getting default type and value information for method parameters. Because the names are unique within a megamodule, no method name as parameter to GETATTRIBUTES is needed.
   ==> The megaprogrammer can choose freely between setting parameters with INVOKE or SETATTRIBUTES.

We decided to change over to version 2.

Reasons: Megaprogrammer should have possibility to preset parameters or set them by INVOKE. This was the reason for introducing SETATTRIBUTES, and no magic and vague "global variables".

Unique names of parameters: These names of parameters are only valid within the CHAIMS-system, i.e. they are created by the megamodule provider when wrapping the megamodule and are documented in the repository. If uniqueness provides problems, a method name can become part of the names of its parameters. Consequence for repository: though we specify for each method which parameters they take, we do not have to add the name of the methods when describing parameters. Example: see revised suggestion for repository sent out by me.

Helper modules

Should modules providing IO and arithmetic be normal megamodules, i.e. remote and adressed by using the asynchronous CHAIMS-primitives (i.e. for each method call at least INVOKE, EXAMINE and EXTRACT would have to be used)? Or should we introduce synchronous invocations for such helper methods? Especially simple arithmetic functions are considered to be short, so splitting up the CALL-statement does only result in akward programming and unreadable programs. Or should we add special primitives for arithmetic, e.g. math.PLUS(...), having a CHAIMS language that will be permanently growing and needing changes to the compiler for each additional operation we need? See also hand written notes on slide.

Decision: We introduce shortcuts and the notion of local helper-modules offering synchronous methods, but we stick with the basic CHAIMS primitives.

- We introduce an additional protocol type in the repository: LOCAL (instead of ORBIX, RMI, DCE etc.). In case of the protocol LOCAL, the name of the computer will then be replaced by the keyword self. We yet have to discuss how we will implement local megamodules.

- We introduce an additional keyword SYNCH for methods in the repository. This is used for methods from which a result can be extracted without a previous examine, because the INVOKE is not an asynchronous call that just starts a computation, but is a synchronous call that also computes and stores the result.
- We introduce the shortcut INVOKEEXTRACT, which is expanded by the compiler into either INVOKE - EXTRACT in case of methods marked as SYNCH in the repository, or INVOKE - EXAMINE - EXTRACT for all other methods. Parameters have yet to be discussed. Shorter name for the shortcut?
- We introduce the shortcut EXTRACTWHENDONE to avoid awkward programming of while-loops. It is expanded by the compiler into WHILE (ih.EXAMINE =! DONE) {}, result = EXTRACT (...), or something more sophisticated like polling all ongoing invocations. Its parameters are the same as for EXTRACT.

Types, Control- and user-data

We decided that method parameters in CHAIMS can be a blob or any of the basic CHAIMS types for parameters which are (with examples how to represent their values in a megaprogram):
- **boolean**: TRUE, FALSE
- **string**: "qweqwer"
- **integer**: 34563
- **real**: 0.678
- **date**: 1998:7:5::10:24:59

We did not yet discuss internal implementation of the various types, nor how they are represented in the CHAIMS protocols.
We **do not offer** any **arithmetic operators** over these basic types - these are offered by math helper modules and used like any other method of a megamodule by INVOKE-EXAMINE-EXTRACT or the shortcut INVOKEEXTRACT.
Yet we **do offer** simple **comparison** (<,>,=,=!,<=,>=) between values of the same basic CHAIMS type.
The type date includes date and time, and is for absolute dates (counted from 0 B.C.) as well as time or date differences. Example: a time difference of 3 hours and 5 minutes would be 0:0:0::3:5:0.

Associative lists

Right now, INVOKE, SETATTRIBUTE take an associative list of name-value pairs as parameter. EXTRACT just returns one value. We want to change that to:
- EXTRACT (and GETATTRIBUTES) return also an associative list. We yet have to decide how that should look like in the CHAIMS language.
- All parameters should somehow be a list of name-type-value triples (as we have it now inside the blobs). This is probably rather an implementation issue as I do not see how that would show in the megaprogram written in the CHAIMS-language. Can anybody remember what reasons we have discussed to have always triples? What do we need the type for? Examples?

Parameter names and names in blobs

There are two kind of names that we have to distinguish, and which need not be identical:
1. **Parameter name**: name that appears in the associative lists in the parameters of INVOKE etc. in the megaprogram, name that appears in the repository. This name is defined by the megamodule provider and is for the **megaprogrammer**, and it may be cryptic, use underscores etc. (e.g. *city_pair_list*)
2. **Descriptive name in datablob**: each element in a datablob, i.e. also the top-level one, has a descriptive name used by the IO-megamodule. This name is a description of the element and this description is used by the

user of a megaprogram. Example: *List of city pairs.* The megamodules are free what kind of name they put in the blob. Values of the same parameter name may have different descriptive name, depending on the method that filled up the value.

The current implementation of the wrappers requires that for the top-level element of a datablob these names are identical. We will have to change that, because they need not be identical.

3 Objectives of CHAIMS

All slides about objectives and long- / short-term goals of CHAIMS see: /CHAIMS/planning/objectives_june_1998.ppt

Short-term objectives:

- **Rest of 1998:**
  - Basic infrastructure (fixing most severe flaws, moving to consistent architecture, all primitives, types, associative lists with handling it, having CORBA) ==> conceptual and implementation work -- CONSOLIDATION
  - More examples (descriptions of scenarios as well as implemented demos), wrapping one (maybe two) additional suites of megamodules. ==> implementation work -- CONSOLIDATION
  - Maybe: Mixing of protocols in client (CORBA, RMI) or/and TCP/IP-three-tier architecture
- **Beginning of 1999** (depending on where we are at the end of 1998):
  - Preparing for more capable scheduler (examples with current scheduler, reading about other scheduler-problems and implementations, redesigning architecture of compiler (interpreter?), designing scheduler algorithm and architecture, writing paper about all this ==> lots of conceptual work, some implementation -- looking ahead for better Scheduler
- **Rest of DARPA-funding, 1999**: **Scheduler**

4 Open points

Complete list as well as decisions, see /Chaims/Planning/Tasks_1998_June.

5 Other work to do that we mentioned

- More reading: everybody is invited to report about papers related to CHAIMS.
- WWW: all our work and concepts should be documented there. Especially all concepts should be written down, so they are not just in our heads and get lost as time goes on.
- Documenting wrappers, repository etc also on the web.
- Glossary, terms.
- Defining CHAIMS-language version 2.0, which will include many of the features discussed above. Once we have defined it in a coherent way, we can define the CHAIMS-protocols for it, and then start implementing them.
- Maybe writing a paper about the characteristics of asynchronous invocations and scheduling.