CHAIMS-project: overview of ongoing and future tasks and open questions

Dorothea Beringer
Version: 8/19/98

Tasks

Task	Who	Priority Version	Deadline	Status
For now				
Transportation demo, 1.1	*wp*	*1.1, high*	*EDSC-Demo-days*	*done*
Parameters: hierarchical setting Describing SETPARAM and INVOKE, GETPARAM and EXTRACT, “hierarchical” setting of method-parameters, getting default-values for parameters. Updating CORBA-protocol and implementing it in compiler and wrapper. Make a demo that effectively uses this hierarchical setting of attributes (e.g. using transportation demo).	lm	high 1.2	September	description: done
ClientId: adding clientId to the INVOKE, SETUP, EXTRACT, SET/GETPARAM, TERMINATE for megamodules in compiler, CORBA-protocol and CORBA-wrapper	lm	high 1.2	September	
Basic CHAIMS types in the language and the protocols*:* *Describe language semantic and syntax, describe protocol, describe mechanisms in compiler, adjust repository description and example. Add all basic types to ASN.1. Implement it in compiler (e.g. converter-functions) and wrapper (if necessary).*	*wp*	*high 1.2*		*Date added to ASN1-gentype (C++)* *done*
Enhancing repository *with comments, parameter and method names for user and compiler. Adjusting our repository example and putting it on the web.*	*wp*	*high 1.2*		*done*
Shortcut EXTRACTWHENDONE*: description, example and implemention.*	*wp*	*medium*		*done, July 27*
Local megamodules: Describe concept of local megamodules, adjust repository description and make examples (keywords LOCAL, self), define and describe protocol for local megamodules, implement this protocol in compiler, make wrapper templates for it (no clientId necessary), make local IO-megamodule and CmathS (simple math-megamodule). Describe and implement shortcut INVOKEEXTRACT.		medium		protocol: db description: db
Glossary: *containing all CHAIMS-terms (and unofficial synomyms)*	*rbetc.*	*high*		*done*
Results from EXTRACT and GETPARAM as lists: New CORBA-idl for lists (RMI-protocol should already have it). Add new syntax for EXTRACT and GETPARAM to compiler. Make example of megaprogram plus demo using these features.	wp	high		in progress
Invocation-scheduler, model free optimization..., see below	lm			
Describing: Diagrams, slides for RMI-wrappers	gs	high		
Taking over IO from Kazuki, updating it for all possible types.	*lm*	*high*	*August*	*done*

Task	Who	Priority Version	Deadline	Status
Comparison operators for all basic CHAIMS types: compiler, after CmathS.		low		
CmathS: Simple arithmetic helper module as local C++ module, adding keyword NOTUNIQUE to repository, see also below	wp	low		
Reorganisation directory structure	*all*	*high*		*done*
Moving one of the transportation megamodules (second optimizer?) to Omnibroker, updating demo with having two megamodules for same tasks and taking one with better estimates.	lm	low		
Describing possible scenarios in which CHAIMS could be used: - supply-chain management	lm	medium		
Masterclient: add generation of mixed clients to compiler, describe it, implement it. Update RMI-protocol and RMI-wrapper (see seperate list of gs). Make presentable demo of a mixed megaprogram.	gs, wp	low	end of september	gentypes:db nearly done
Version management: preventing chaos in our different versions of compiler code, wrapper code, examples, demos, ...				ad hoc solutions and decisions
Omnibroker: installing it for everybody to use	lm	high		
Later				
TCP/IP: wrapper, protocol, compiler, example	ns?			
Error-checking in parser, based on enhanced repository information.				
Run-time error-checking in CSRT.				
Parameter names versus descriptive names in datablobs: description, taking out condition that parameter names are equal to descriptive names in blobs in CORBA wrapper. Adding descriptive names to literals in CHAIMS megaprogram (changes in Syntax description, Compiler, megamodule examples).		medium		description done, db
Second demo-suite of megamodules with megaprograms.	wp?			
I/O as local Java helper module (after masterclient)				
Wrapper for DCE	wp?	verylow		
Completing CLAM: Make complex megaprogram examples, and enhance language as necessary. ==> Apart from the current CHAIMS-language, we could have a more sophisticated, yet still purely compositonal language, that is a superset of CHAIMS. See also open points.	ns, lm?			
CHAIMS-protocols: Generic description, protocol-specific descriptions.	*db*			*done*
CMathC: Complex math helper module: takes a function/rule and executes that function/rule for a set of parameters (function, rules just include operations already implemented in Simple math helper module.				
Wizard for composition				
Wizard for wrapping				
Overview results, interactive megaprogramming				
WWW...				
Overview-pages	*lm*	*medium*	*ongoing*	*ongoing task*

Task	Who	Priority Version	Deadline	Status
ATTENTION: wherever above is said “describe”, this means putting it also up on the web!!!	all	always		
Password for protecting note pages.	*lm*			
Add section “Architecture” to web-page with project description: description from papers, diagrams	*lm*	*high*		*done*
Description of CHAIMS 2.0, Syntax in EBNF, ...	*db*	*high*		*done*
User-manual				
Provider-manual				
Reading				
Compare CLAM to ISL of HADAS, http://www.dsg.technion.ac.il				
Workflow-management and CHAIMS...				
Publications				
Preparing DEXA	db		August	

Scheduling, various subtasks:

- integrate preprocessor, make examples with preprocessor
- enhance wrappers (default: not zero but averages) and megamodule examples to return decent estimate values
- write about prepropressor and general stuff of scheduling (see papers below), publish paper
- verify, if making interpreter is possible in current CHAIMS infrastructure
- investiage and discuss advantages/disadvantages of writing interpreter in Java
- read about scheduling/optimization (and especially model free optimization), write about it in paper
- make and discuss design for interpreter and its scheduler
- implement interpreter, make examples
- write about interpreter in paper, and publish paper

CmathS:

Functions offered by CmathS: see description in CHAIMS/Code/Helpermodules/CmathS
How to wrap CmathS: take updated CORBA-C++ wrapper, take out clientID, change the part of parameter marshalling so method despatching is done type-dependant
Additions to Repository: new keyword NOTUNIQUE for repetitive parameters of methods. Ex: METHOD CmathS.Min(IN par NOTUNIQUE, RES res UNIQUE); UNIQUE can be omitted as it is default

Open points, missing parts, flaws, ...

Open Point	What	Variants	Decision
COMPILER			
Interpreter	Is an interpreter possible with current architecture?		
LANGUAGE			
EXTRACT	Returning assoc. list instead of just one value, having parameters that denote the values to be returned.	Would involve the introduction of associative lists in the megaprogramming language.	
ESTIMATE (methodname)	should return a triple: - time (in seconds) - fee for one invocation (in $) - volume of return-data (in kbytes)		*to be done with scheduler integration*
recognizing end of commands in megaprogram	Right now the parser assumes that a command is ended by a newline-character, which is not true for longer commands written over several lines.	- reeintroducing ; - parser deals with logical structure	
CHAIMS-language: callId's	Copying of callId's should be explicitely forbidden.		
Appealing front-end	Our current front-end (CHAIMS-language) is quite akward - it claims to be on a high-level, yet the megaprogrammer mainly does silly coding.	1) Short-cuts, e.g. EXTRACTWHENDONE, automatice generation of estimate code.... 2) Graphical user-interface (maybe easy to create with nowadays graphical tools) that generates input for CHAIMS-compiler. Compare also to workflow-management systems like changengine of HP. 3) A real high-level language... yet graphical input could be clearer.	*only long-term objective, if at all*
CHAIMS - procedural language?	Right now, CHAIMS is procedural language. Is that okay? See also minutes weekly meeting 6/24/98.	- graphical language describing data-dependancies - query-language	
CHAIMS types	Having an identifier rather than a string? or both?		
Hardcoded values	Should we allow hardcoding of values also for opaque types? How would the language look like for this?		No.
CHAIMS-language	The CHAIMS-language has to be simple, and purely comositional. Yet it needs not be the same as the CHAIMs-protocol, though it should contain the CHAIMS-protocol as subset. What is the language we really need for writing useful programs? Is a purely compositional language really feasible? Differences to architecture languages, coordination languages?	Example better name: CAMIC: **c**omposition language for **a**utonomous **m**egamodules **i**n **C**HAIMS	
Repetitive arguments	E.g. for math module: minimum of three arguments (used to find minimum of three times of estimates, there is no other easy way to write that in CLAM).	- lists in CLAM - or keywork REP in repository for repetitive arguments	

Open Point	What	Variants	Decision
Comparing time and fee	Finding the optimum of estimates given time as well as fee. How to ship these pairs of parameters to a optimizer module?	- Partial order in name-value list (first all times, then all fees, the megamodule knows how to match them together) - additional structure (grouping of tuples) in name-value list - getting a OPAQUE type from estimate containing all (additional variant to getting time or fee) - first calling several functions for pairs, and then function to compare these results	
	Lists in CLAM? How to make complex calculations on estimates?		
WRAPPER			
ESTIMATE, for scheduler	By default, wrappers should estimate time by summing up previous execution times, instead of returning just 0..		*to be done with scheduler integration*
OTHER PARTS			
Cost-function for scheduler	- should take into account real values for fee and volume - more sophisticated cost-function?	- Cost function for moving invokes outside of an if-block: taking into account estimates of other invokes and extracts between the possible invoke and the beginning of if... - more interpreter-like and rule-based implementation	*to be done with scheduler integration*
Repostiory, Scheduler	In order to save resources (and fees) for no longer used resources for holding results at the server side, the scheduler should terminate all invocations that are no longer accessed. Yet there is a special subclass of invocations that should not be killed automatically, because they perform tasks without providing results to the megaprogram, e.g. printing.	Flag "no_result" in repository for each method that should not be terminated automatically by scheduler.	*to be done with scheduler integration*
Look-up of method names, automatic comparison of competing services	Right now, we look up megamodule names and not method names in the repository. If two megamodules offer the same services, the megaprogrammer has to know that.	- same megamodule name for servers with same services, or additional megamodule name for servers with overlapping services - also lookup for methodnames, so same methodname can be used for different methods that do same thing - automatic generation of any code related to get estimates from two megamodules offering same services, and for choosing the better one	
Error handling 1	When an error occurs in the server or connection, there is no way to inform the client/server. Also the user never gets errors, even not from compiler.	- timeouts if connection gets lost - CORBA-protocol: returning an error-code instead of void for all CHAIMS-primitives - error-messages in parser of compiler for user to standard-out	*postponed*

Open Point	What	Variants	Decision
Error-handling 2	Primitives should be able to return an error-status. Time-outs of megamodules. Parsing errors in compiler. Error-information to user (how?).	No consistent concepts yet.	*postponed*
VARIA			
WfMS, BPMS	How do we differ from workflow and business process management systems? Where are we subset? What do we add?		

Ideas for papers

MegaGene

about the application of CHAIMS to genomics

Heterogeneity

Why, how in CHAIMS.....

Overview CHAIMS protocol:

Describes CHAIMS from the point of view of protocols (and wrappers); contains protocol, gentype, comparison to other protocols and similar systems; mainly written by db with help from everybody for the wrapper part

CHAIMS and supply-chain-management:

Scenario for using CHAIMS in that domain, example of megaprograms and megamodules, description of who is megamodule provider and who is megaprogrammer and who uses the megaprogram; Laurence and Guido will give it a start (second priority after upgrading current infrastructure to new protocols and adding master client)

Parallelism and invocation scheduling in CHAIMS:

CHAIMS potential for parallelism with sequential client, why this approach to parallelism, why invocation scheduling, differences of our approach to optimization (based on run-time estimate) in contrast to model based optimization, comparison to optimization and scheduling in compilers and workflow-management systems, description of our current pre-processing scheduler and our experiences with it (which we yet have to gather when we intergrate it and run examples with it), description of the future run-time scheduler, comparison of them with advantages and disadvantages and consequences on CHAIMS architecture.

CHAIMS as CORBA services

CLAM:

A composition only language: why, its strengths, its characteristics,....
once the language has more evolved....