Implementation of Integrated Systems and Resources for Information and Computing

Claus-Peter Rückemann
Leibniz Universität Hannover,
Westfälische Wilhelms-Universität Münster (WWU),
North-German Supercomputing Alliance (HLRN), Germany
Email: ruckema@uni-muenster.de

***Abstract*—This paper presents the practical results from the real-life implementation of interactive complex systems in specialised High End Computing environments. The successful implementation has been made possible using a new concept of higher-level data structures for dynamical applications and configuration of the resources. The discussion shows how an implementation of integrated information systems, compute and storage resources has been achieved. The implementation uses techniques ensuring to create a flexible way for communication with complex information and computing systems. Besides Inter-Process Communication these are mainly Object Envelopes for object handling and Compute Envelopes for computation objects. These algorithms provide means for generic data processing and flexible information exchange. Targeting mission critical environments, the interfaces can embed instruction information, validation and verification methods. The application covers challenges of collaborative implementation, legal, and security issues with these processes. The focus is on integrating information and computing systems, Distributed and High Performance Computing (HPC) resources, for use in natural sciences disciplines and scientific information systems. Implementing higher-level data structure frameworks for dynamical applications and resources configuration has led to scalable and modular integrated public / commercial information system components.**

Keywords–Integrated Systems; Information Systems; Computing Systems; Geosciences; High Performance Computing

I. INTRODUCTION

In High Performance Computing (HPC) supercomputers, that means computer systems at the upper performance limit of current technical capabilities for computing, are employed to solve challenging scientific problems. In consequence there is no general or common architecture and configuration for HPC resources as in the lower parts of the performance pyramid.

Within the last decades a large number of implementations of information systems, computing and storage systems and other resources have been created. Nearly all of these implementations lack features for extending information systems with the various resources available. Thus, the integration could be opening advances using larger resources, interactive processing, and reducing time consumption for assigned tasks. Most of these applications and resources are very complex, standalone systems and used that way, neglecting that for many sophisticated use cases conjoint applications are desirable.

The next generation of systems neccessary for providing profound means for communication and computation will have to gather methods evolved by active interdisciplinary interchange, grown with the requirements of the last decades: The information and computing disciplines need means for in praxi collaboration from disciplines, structural sciences, computer science, computing science, and information science. Examples are computing intensive interactive environmental monitoring and information systems or simulation supported dynamical geoinformation systems. In this manner an efficient development and operation can be put into practice for making interactive use of systems with tenthousands of thousands of nodes, ten to hundred thousands of compute cores and hundred thousands to millions of system parts like memory bars and hard disk drives. Methodological sciences means sciences of developing methods for using resources and techniques for gathering new scientific insights. For years, “methodological sciences” or more precise “methodological techniques” have been commonly propagated to solve the problems of High End Computing. It has been commonly experienced that this is not true as there are no applicatory results regarding the essential requirements of complex and integrated high end systems. The available “methods and techniques” is not what we can use for supercomputing where every application and system architecture is different. Up to now this difference is implicit with common tender and operation strategy for various efficiency and economical reasons.

The experiences with integrated systems have been compiled in various projects over the last years [1], [2]. Legal issues and object security have been discussed at the CYBERLAWS 2010 and 2011 conferences. The architecture of the framework and suitable components used, have been tested by implementing various integrated systems. The following sections show components of an integrated geoscientific information and computing systems developed in one of these case studies that can be used for environmental monitoring or feeding expert systems. None of the participating

 ISBN: 978-1-61208-161-8

industry and scientific parties can or will create one single application from the components discussed here. The goal is to enable the necessary operation, nevertheless these are separate components. For the last years practical solutions to various requirements for communication requests have been implemented in a number of projects and case studies using various resources [3], [4], [5], [6], [7]. The most important communication facilities for integrated information and computing systems are:

- Communication requests with applications (example: Inter-Process Communication, IPC).
- Storage object requests (framework example: Object Envelopes, OEN).
- Compute requests (framework example: Compute Envelopes, CEN).

Based on these components an integrated solution has been built, for use with local HPC resources supported by distributed information and compute resources. From the point of view of resources providers and integrators of HPC resources it would make very little sense to describe the application components here. Applications details have been published for several components before. For the core issues the conceptional results are by far the most important.

This paper is organised as follows. Section two presents motivation, challenges and complexity with the implementation. Sections three and four describe the prerequisites and basic resources configuration for the implementation. Sections five and six show the components and dependencies for integrated systems and resources. Section seven discusses the time dependence of the integrated solutions. Section eight describes the system implementation, and Section nine does the evaluation. Sections ten and eleven summarise the lessons learned, conclusions, and future work.

II. Motivation

With the implementation use cases for Information Systems the suitability of Distributed and High Performance Computing resources have been studied. These use cases have focus on event triggered communication, dynamical cartography, compute and storage resources. The goal has been, to bring together the features and the experiences for an integrated information and computing system. An example that has been implemented is a spatial information system with hundreds of thousands of ad-hoc simulations of interest. Within these interactive systems as many “next informations of interest” as possible can be dynamically calculated in parallel, near real-time, in order to be of any practical use. In the following passages we will show an environmental component exactly using this implementation for many thousands of points of interest.

Due to the complexity of integrated information and computing systems, we have applied meta-instructions and signatures for algorithms and interfaces. For these cases, envelopes and IPC has been used to provide a unique event and process triggered interface for event, computing, and storage access.

III. System prerequisites

For implementation and testing a suitable system architecture and hardware had been neccessary. A single local system had to fulfill the following minimal criteria:

- Capacity for more than 5000 subjobs per job.
- At least one compute core available per subjob.
- Interactive batch system.
- No distributed compute and storage resources.
- Fast separate InfiniBand networks for compute and IO.
- Highly performant parallel filesystem.
- Available for being fully configurable.

A system provided being fully configurable means especially configuration of hardware, network, operating system, middleware, scheduling, batch system. At this size this normally involves a time interval of at least three to six months.

It should be obvious that there are not many installations of some reasonable size and complexity that could be provided, configured and operated that way if in parallel to normal operation and production.

The available HPC and distributed resources at ZIV and HLRN as well as commercially provided High End Computing installations have been sufficient to fulfill all the necessary criteria.

IV. Basic resources configuration

Elementary operating system components on the resources involved are: AIX, Solaris, and various Linux distributions (SLES, Scientific Linux). Elementary middlewares, protocols, and accounting systems used for the integrated components are: Globus Toolkit, SGAS, DGAS. Unicore, SAGA, SOAP, and many others can be integrated, too. For communication and parallelisation MPI (Open-MPI [8], MPI from SGI, Intel, HP, IBM), OpenMP, MPICH, MVAPICH and other methods have been used along with IPC regarding to the type of operation and optimisation of algorithms needed. For the scheduling and batch systems the resources used Moab, Torque, LoadLeveler, and SGE.

All these “tools” are only middleware components, protocols, interfaces or isolated applications. They are certainly used on the system resources but they cannot integrate anything, not on the disciplines/application level, not on the services level, not on the resources level. So we want to concentrate on the important high-level issues for the further advanced view of components.

V. Components

Using the following concepts, we can, mostly for any system, implement:

- Application communication via IPC.
- Application triggering on events.
- Storage object requests based on envelopes.

 ISBN: 978-1-61208-161-8

- Compute requests based on envelopes.

For demonstration and studies flexible and open Active Source Information System components have been used for maximum transparency. This allows OO-support (object, element) on application level as well as multi-system support. Listing 1 shows a simple example for application communication with framework-internal and external applications (Inter-Process Communication, IPC).

```
1 catch {
2  send {rasmol #1} "$what"
3 }
```

Listing 1. Application communication (IPC).

This is self-descriptive Tcl syntax. In this case the IPC `send` is starting a molecular graphics visualisation tool and catching messages for further analysis by the components.

Listing 2 shows an example of how the communication triggering can be linked to application components.

```
1 text 450.0 535.0 -tags {itemtext relictrotatex} -fill
   yellow -text "Rotate␣x" -justify center
2 ...
3 $w bind relictrotatex <Button-1> {sendAllRasMol {rotate
   x 10}}
4 $w bind relictballsandsticks <Button-1> {sendAllRasMol
   {spacefill 100}}
5 $w bind relictwhitebg <Button-1> {sendAllRasMol {set
   background white}}
6 $w bind relictzoom100 <Button-1> {sendAllRasMol {zoom
   100}}
```

Listing 2. Application component triggering.

Tcl language objects like `text` carry tag names (`relictrotatex`) and dynamical events like `Button` events are dynamically assigned and a user defined subroutine `sendAllRasMol` is executed, triggering parallel visualisation. Storage object requests for distributed resources can be done via OEN. Listing 3 shows a small example for a generic OEN file.

```
1  <ObjectEnvelope><!-- ObjectEnvelope (OEN)-->
2  <Object>
3  <Filename>GIS_Case_Study_20090804.jpg</Filename>
4  <Md5sum>...</Md5sum>
5  <Sha1sum>...</Sha1sum>
6  <DateCreated>2010-08-01:221114</DateCreated>
7  <DateModified>2010-08-01:222029</DateModified>
8  <ID>...</ID><CertificateID>...</CertificateID>
9  <Signature>...</Signature>
10 <Content><ContentData>...</ContentData></Content>
11 </Object>
12 </ObjectEnvelope>
```

Listing 3. Storage object request (OEN).

OEN are containing element structures for handling and embedding data and information, like `Filename` and `Content`. An end-user public client application may be implemented via a browser plugin, based on appropriate services. With OEN instructions embedded in envelopes, for example as XML-based element structure representation, content can be handled as content-stream or as content-reference. Algorithms can respect any meta-data for objects and handle different object and file formats while staying transparent and portable. Using the content features the original documents can stay unmodified. The way this will have to be implemented for different use cases depends on the situation, and in many cases on the size and number of data objects. but the hierarchical structured meta data is uniform and easily parsable. Further it supports signed object elements (`Signature`), validation and verification via Public Key Infrastructure (PKI) and is usable with sources and binaries like Active Source. Compute requests for distributed resources are handled via CEN interfaces. Listing 4 shows a generic CEN file with embedded compute instructions. Content can be handled as content-stream or as content-reference (`Content`, `ContentReference`). Compute instruction sets are self-descriptive and can be preconfigured to the local compute environment.

```
1  <ComputeEnvelope><!-- ComputeEnvelope (CEN)-->
2  <Instruction>
3  <Filename>Processing_Batch_GIS612.pbs</Filename>
4  <Md5sum>...</Md5sum>
5  <Sha1sum>...</Sha1sum>
6  <Sha512sum>...</Sha512sum>
7  <DateCreated>2010-08-01:201057</DateCreated>
8  <DateModified>2010-08-01:211804</DateModified>
9  <ID>...</ID>
10 <CertificateID>...</CertificateID>
11 <Signature>...</Signature>
12 <Content><DataReference>https://doi...</DataReference><
    /Content>
13 <Script><Pbs>
14 <Shell>#!/bin/bash</Shell>
15 <JobName>#PBS -N myjob</JobName>
16 <Oe>#PBS -j oe</Oe>
17 <Walltime>#PBS -l walltime=00:10:00</Walltime>
18 <NodesPpn>#PBS -l nodes=8:ppn=4</NodesPpn>
19 <Feature>#PBS -l feature=ice</Feature>
20 <Partition>#PBS -l partition=hannover</Partition>
21 <Accesspolicy>#PBS -l naccesspolicy=singlejob</
    Accesspolicy>
22 <Module>module load mpt</Module>
23 <Cd>cd $PBS_O_WORKDIR</Cd>
24 <Np>np=$(cat $PBS_NODEFILE | wc -l)</Np>
25 <Exec>mpiexec_mpt -np $np ./dyna.out 2>&1</Exec>
26 </Pbs></Script>
27 </Instruction>
28 </ComputeEnvelope>
```

Listing 4. Compute request (CEN).

In this case standard PBS batch instructions like `walltime` and `nodes` are used. The way this will have to be implemented for different use cases depends on the situation, an in many cases on the size and number of data objects. An important benefit of content-reference with high performant distributed or multicore resources is that references can be processed in parallel on these architectures. The number of physical parallel resources and the transfer capacities inside the network are limiting factors.

VI. Integrated systems with coupled resources

Figure 1 shows the applied integration of the information and communication systems with coupled computation resources, namely compute resources and storage resources. For integrating the features of information and communication systems with powerful compute resources and storage,

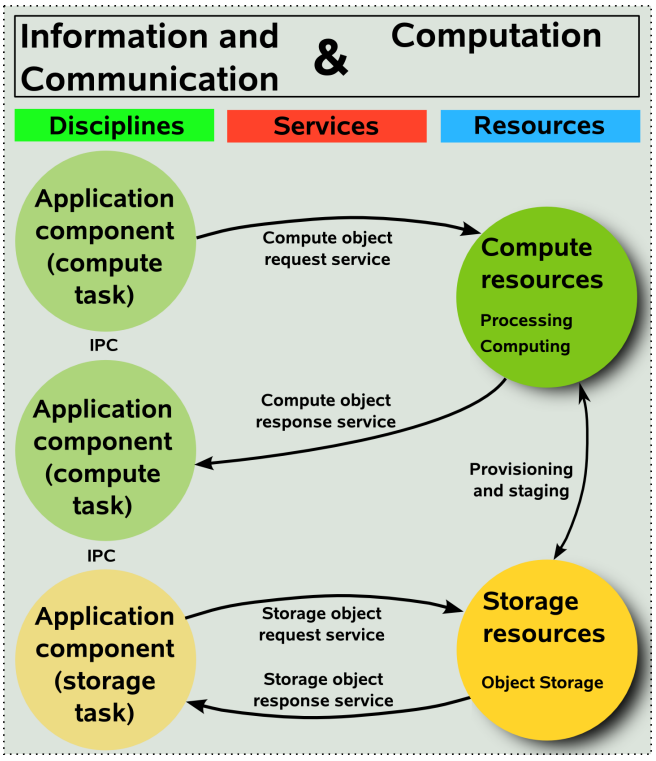

Figure 1. Integrated systems and resources.

it has been necessary to implement interfaces and software applications being able to efficiently use the benefits of High End Computing resources.

Following the results of the long-term case studies [9] three columns namely disciplines (as geosciences), services (as middleware and compute services), resources (computing and storage) had to be figured out for this scenario.

The discipline column shows application components with the state for a compute task and an application component with state for a storage task. Local tasks, ordinary communication between the applications without the need for external computing power, can as usual be done using a local service, for example using Inter-Process Communication (IPC).

Using services, requests can be sent to the configured compute object request service for compute intensive tasks. Results delivered from the computation are delivered for the compute object response service, giving the desired information back the one of the application components. Compute Envelopes (CEN) can be used for exchange of the compute requests.

The resources column does provide compute resources for processing and computing as well as storage resources for object storage. Commonly these resources are separated for backend use with high end applications customised on the compute resources.

Application components may trigger storage tasks using a storage object request service. Data objects are handled by the service and delivered to the storage resources. Request for retrieval from the storage are handled by the storage object response service. Object Envelopes (OEN) can be used for exchange of the object requests.

For enabling overall scalable integrated systems, mostly for large and voluminous data, the computing and storage resources can communicate for using stored data from within compute tasks and for provisioning and staging of data.

These services are so far using a loosely coupled parallel computing, parallelised on the application component level. Each single task can itself contain scalable and loosely to highly parallel computing jobs running on the available compute resources. MPI and OpenMP can be used here. The CEN Envelopes are used to transfer the tasks and their description.

The user has to ensure that with using the resources the interactivity and latencies of the integrated system still result in appropriate and usable comprehensive system.

Among the compute and storage resources a provisioning and staging mechanism for data and resources requests and responses can be used. Therefore triggering of computing for storage operations and triggering of storage operations for computing are available.

VII. Time Dependence

The same reason why opening large resources for information system purposes is desirable, there is still a dependence on time consumption for interactive and batch processing. Table I shows the characteristic tasks and times that have been considered practical [9] with the current information system applications, for example with environmental monitoring and information system components.

Table I
Tasks and times regarding the overall integrated systems.

Task	*Compute / Storage Times*
In-time events requests	1–3 seconds
Interactive requests	up to 3 minutes
Data processing	1–24 hours
Processing data transfer	n days
Object storage interval	n weeks
Object archive interval	n years

The different tasks afford appropriate policies for interactive and batch use inside the resources network. Besides that, the user and the developer of the application components can use the computing and storage interfaces in order to extend the application facilities using these non-local resources.

Nevertheless for configuring the system and for implementing new operations the decisions have to be made which type of implementation would be more suitable.

Interactive request are mostly not acceptable when response time are longer than a few minutes, even for specialised information systems. HPC systems have shown a good performance for parallelisation of interactive subjobs,

 ISBN: 978-1-61208-161-8

being in the range of minutes. Whereas distributed resources are much less scalable and provide less performance due to smaller and mostly different resource architecture types and non-exclusive use. Compute times for 1 to 24 hours will force to decide about the field of operation of the system application, when assigning the tasks and events. For example those compute resources doing computation on large jobs are the computational bottleneck for interactive use. One the other hand for information system purposes, for example needing visual updates within longer intervals, like for special monitoring purposes for environmental, weather, volcano or catastrophes monitoring and using remote sensing, this scenario is very appropriate. Storage resources and object management can reduce the upload and staging times for objects that can be used more than once. Service providers are confronted with the fact that highly performant storage with reliable and long time interval archiving facilities will be needed at a reasonable price.

VIII. Implemented system

The system implemented integrates the component features described from the projects and case studies. Figure 2 shows the implementation of the integrated systems and resources. The components were taken from the GEXI case studies and the well known actmap components [9], [10]. These components handle information like spatial and remote sensing data, can be used for dynamical cartography and trigger events, provide IPC and network communication, and integrate elements from remote compute and storage resources as available with existing compute resources [5], [6], [7]. Processing and computing tasks can for example consist of raytracing, seismic stacking, image transformation and calculation, pattern recognition, database requests, and post processing. The modularisation for development and operation of advanced HPC and application resources and services can improve the multidisciplinary cooperation. The complexity of operation and usage policies is reduced.

A. Application components

The integrated system is built in three main columns, application components in use with scientific tasks for various disciplines, meaning the conventional scientific desktop and information system environment, services, and resources. These columns are well understood from the Grid-GIS house framework. In opposite to the conventional isolated usage scheme, interaction and communication is not restricted to happen inside the disciplines and resources columns only. Non isolated usage can speed up the development of new components and the modification of existing components in complex environments. The workflows with the application scenarios (Figure 2) are:

a) Application communication.
b) Storage task.
c) Compute task.

These tasks can consists of a request, triggered by some event, and a response, when the resources operation is finished. The response can contain data with the status or not, in case that for example an object has been stored on the resources. Based on this algorithm, task definition can be reasonably portable, transparent, extendable, flexible, and scalable.

B. Application communication

A) Request: The internal and framework-external application is triggered from within the framework components (rasmol is used in the example). From within an actmap component a task to an application component is triggered. IPC calls are used with data and information defined for the event.

Response: A framework-external application is started (rasmol locally on the desktop). The external application can further be triggered from the applications available.

C. Storage task

B) Request: From within an actmap component a storage task is triggered. The stored OEN definition is used to transmit the task to the services. The services do the validation, configuration checks, create the data instructions and initiate the execution of the object request and processing for the resources.

Response: The processing output is transmitted to the services for element creation and the element (in this example a photo image) is integrated into the actmap component.

D. Compute task

C) Request: From within an actmap component a compute task is triggered. The stored CEN definition is used to transmit the task to the services. The services do the validation (configuration checks, create the compute instructions and initiate the execution of the compute request and compute job for the resources.

Response: The processing output is transmitted to the services for element creation and the element (in this example a remote sensing image and vector object) is integrated into the actmap component.

IX. Evaluation

The target has been to integrate application communication, computing, and storage resources for handling computing requests and content for distributed storage within one system architecture. The technical details of the components have been discussed in several publications and used in applications publically available. The case study has demonstrated that existing information systems and resources can be easily integrated using envelope interfaces in order to achieve a flexible computing and storage access. As the goal has been to demonstrate the principle and for the modular system components used and due to the previous

 ISBN: 978-1-61208-161-8

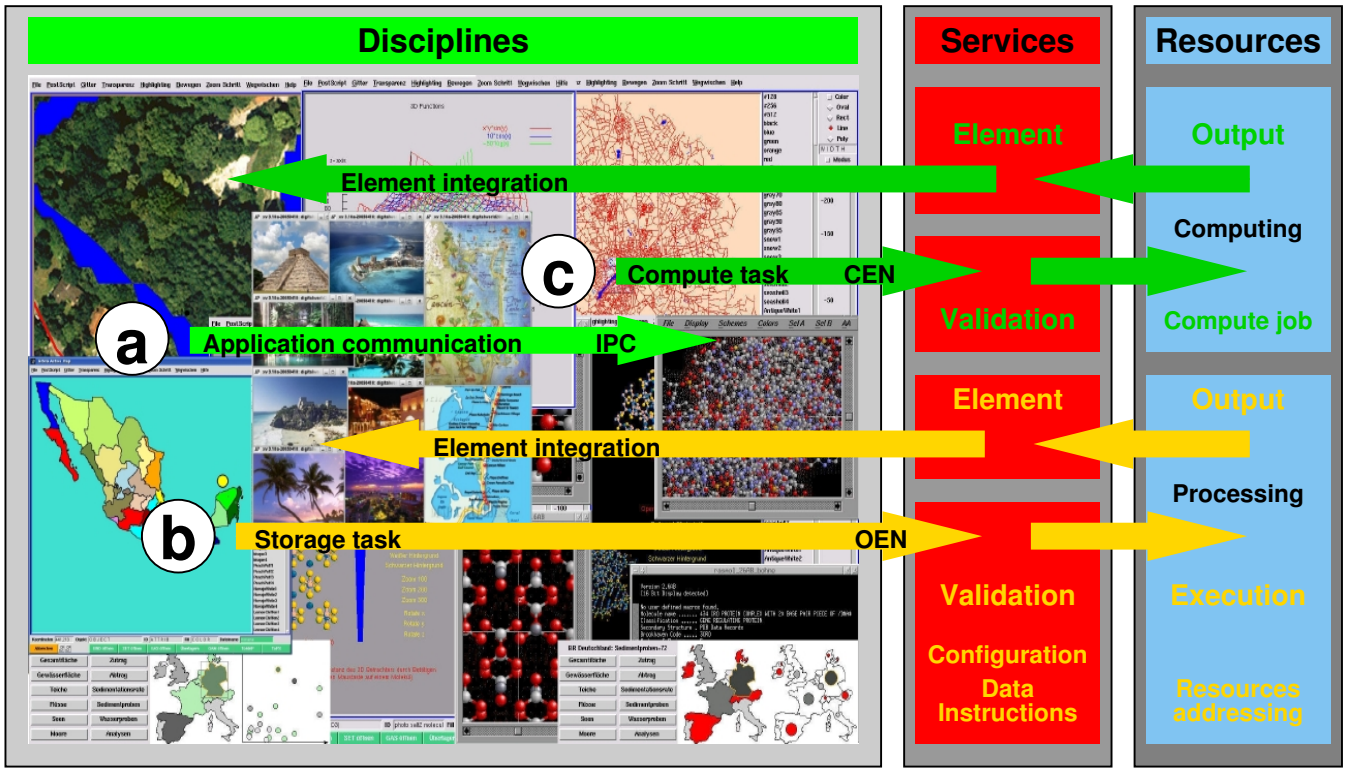

Figure 2. Implementation of the different tasks with integrated systems and resources.

experiences, the services necessary for integration afforded minimal scripting work.

With modern information and computing systems object management is a major challenge for software and hardware infrastructure. Resulting from the case studies with information systems and compute resources, signed objects embedded in OEN can provide a flexible solution.

The primary benefits shown from the case studies of this implementation are:

- Build a defined interface between dedicated information system components and computing system components.
- Uniform algorithm for using environment components.
- Integration of information and computing systems.
- Speed-up the development of new components and the modification of existing components in complex environments.
- Portable, transparent, extendable, flexible, and scalable.
- Hierarchical structured meta data, easily parsable.
- OO-support (object, element) on application level.
- Multi-system support.
- Support for signed object elements, validation and verification via PKI.
- Usable with sources and binaries like Active Source.
- Portable algorithms in between different object and file formats, respecting meta-data for objects.
- Original documents can stay unmodified.
- The solution is most transparent, extendable, flexible, and scalable, for security aspects and modularisation.
- Handling of cooperation and operation policies is less complex [11].
- Guaranteed data integrity and authentication derived from the cryptographic strength of current asymmetric algorithms and digital signature processes.
- Flexible meta data association for any object and data type, including check sums and time stamps.

Main drawbacks are:

- Additional complexity due to additional resources and system environment features like batch scripting (Condor [12], Moab / Torque [13]) and using verification/PKI.
- Complexity of parsing and configuration.
- Additional software clients might come handy to handle resources and generate, store and manage associated data and certificates.

The context is an important aspect, though it cannot be called “drawback” here. With closed products, e.g., when memory requirements are not transparent, it is difficult for users to specify their needs. Anyhow, testing is in many cases not the answer in productive environments. Separate measures have to be taken to otherwise minimise possible problems and ease the use of resources in productive operation.

Even in the face of the drawbacks, for information systems making standardised use of large numbers of accesses via the means of interfaces, the envelopes can provide efficient management and access, as programming interfaces can.

 ISBN: 978-1-61208-161-8

X. Lessons Learned

Integrating information system components and external resources has provided a very flexible and extensible solution for complex application scenarios. OEN and CEN, based on generic envelopes, have provided a very flexible and extensible solution for creating portable, secure objects handling and processing components with integrated information and computing systems.

The case study showed that very different kinds of object data structures and instruction sets may be handled with the envelopes, in embedded or referenced use. Meta data, signatures, check sums, and instruction information can be used and customised in various ways for flexibly implementing information and computing system components. Support for transfer and staging of data in many aspects further depends on system configuration and resources as for example physical bottlenecks cannot be eliminated by any kind of software means.

For future integrated information and computing systems an interface layer between user configuration and system configuration would be very helpful. From system side in the future we need least operation-invasive functioning operating system resources limits, e.g., for memory and a flexible limits management. Homogeneous compute and storage resources and strong standardisation efforts for the implementation could support the use of high end resources regarding economic and efficient operation and use.

XI. Conclusion and future work

It has been demonstrated that integrated information and computing systems can be successfully built, employing a flexible and portable envelopes framework. For this implementation Object Envelopes, Compute Envelopes, and IPC have been used. For the case study Active Source components and Distributed and High Performance Computing resources provided the information system and computing environment. With integrated information and computing systems the following main results have been achieved. Local and inter-application communication can be done using IPC. Object Envelopes can be natively used for handling objects and implementing validation and verification methods for communication. Compute Envelopes can be used in order to define information system computation objects and embed instruction information. These algorithms provide means for generic data processing and flexible information exchange.

The concept used has been found to be least invasive to the information system side as well as to the resources used while being very modular and scalable. The services in between can hold most of the complexity and standardisation issues and even handle products that are meant to be commercially used or licensed. In the future we will have to integrate the features into a global framework for communication purposes and defining standardised interfaces. This implementation has demonstrated a flexible basic approach in order to begin to pave the way and show the next aspects to go on with for future integrated information and computing systems.

Acknowledgements

We are grateful to all national and international academic and industry partners in the GEXI cooperations for the innovative constructive work and to the colleagues at the Leibniz Universität Hannover, IRI, HLRN, WWU, ZIV, D-Grid for participating in fruitful case studies and the participants of the EULISP Programme for prolific discussion of scientific, legal, and technical aspects over the last years.

References

[1] C.-P. Rückemann, "Using Parallel MultiCore and HPC Systems for Dynamical Visualisation," in *Proceedings, GEOWS 2009, Cancun, Mexico*. IEEE CSP, IEEE Xplore, 2009, pp. 13–18, ISBN: 978-0-7695-3527-2, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4782685&isnumber=4782675 [accessed: 2011-02-20].

[2] C.-P. Rückemann, "Envelope Interfaces for Geoscientific Processing with High Performance Computing and Information Systems," in *Proceedings, GEOProcessing 2011, Gosier, Guadeloupe, France*. XPS, 2011, pp. 23–28, ISBN: 978-1-61208-003-1, URL: http://www.thinkmind.org/download.php?articleid=geoprocessing_2011_2_10_30030 [accessed: 2011-03-20].

[3] "D-Grid, The German Grid Initiative," 2008, URL: `http://www.d-grid.de` [accessed: 2009-11-16].

[4] ZIVGrid, "ZIV der WWU Münster – ZIVGrid," 2008, URL: `http://www.uni-muenster.de/ZIV/Server/ZIVGrid/` [accessed: 2008-12-23].

[5] "ZIVHPC, HPC Computing Resources," 2011, URL: https://www.uni-muenster.de/ZIV/Technik/ZIVHPC/index.html [accessed: 2011-07-10].

[6] "HLRN, North-German Supercomputing Alliance (Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen)," 2011, URL: http://www.hlrn.de [accessed: 2011-07-10].

[7] "ZIVSMP, SMP Computing Resources," 2011, URL: https://www.uni-muenster.de/ZIV/Technik/ZIVHPC/ZIVSMP.html [accessed: 2011-07-10].

[8] "Open-MPI," 2011, URL: http://www.open-mpi.org [accessed: 2011-07-12].

[9] "Geo Exploration and Information (GEXI)," 1996, 1999, 2010, 2011, URL: http://www.user.uni-hannover.de/cpr/x/rprojs/en/index.html#GEXI (Information) [acc.: 2011-02-20].

[10] "Applications with Active Map Software, Screenshots," 2005, URL: http://wwwmath.uni-muenster.de/cs/u/ruckema/x/sciframe/en/screenshots.html [accessed: 2011-02-20].

[11] "EULISP Lecture Notes, European Legal Informatics Study Programme, Institute for Legal Informatics, Leibniz Universität Hannover (IRI / LUH)," 2011, URL: http://www.eulisp.de [accessed: 2011-02-20].

[12] "Condor, High Throughput Computing," 2011, URL: http://www.cs.wisc.edu/condor/ [accessed: 2010-12-26].

[13] "Moab Admin Manual, Moab Users Guide," 2011, URL: http://www.clusterresources.com/products/mwm/moabdocs/index.shtml [accessed: 2011-02-20].

 ISBN: 978-1-61208-161-8