ESPRIT SYSTEMS

www.esprit-systems.de

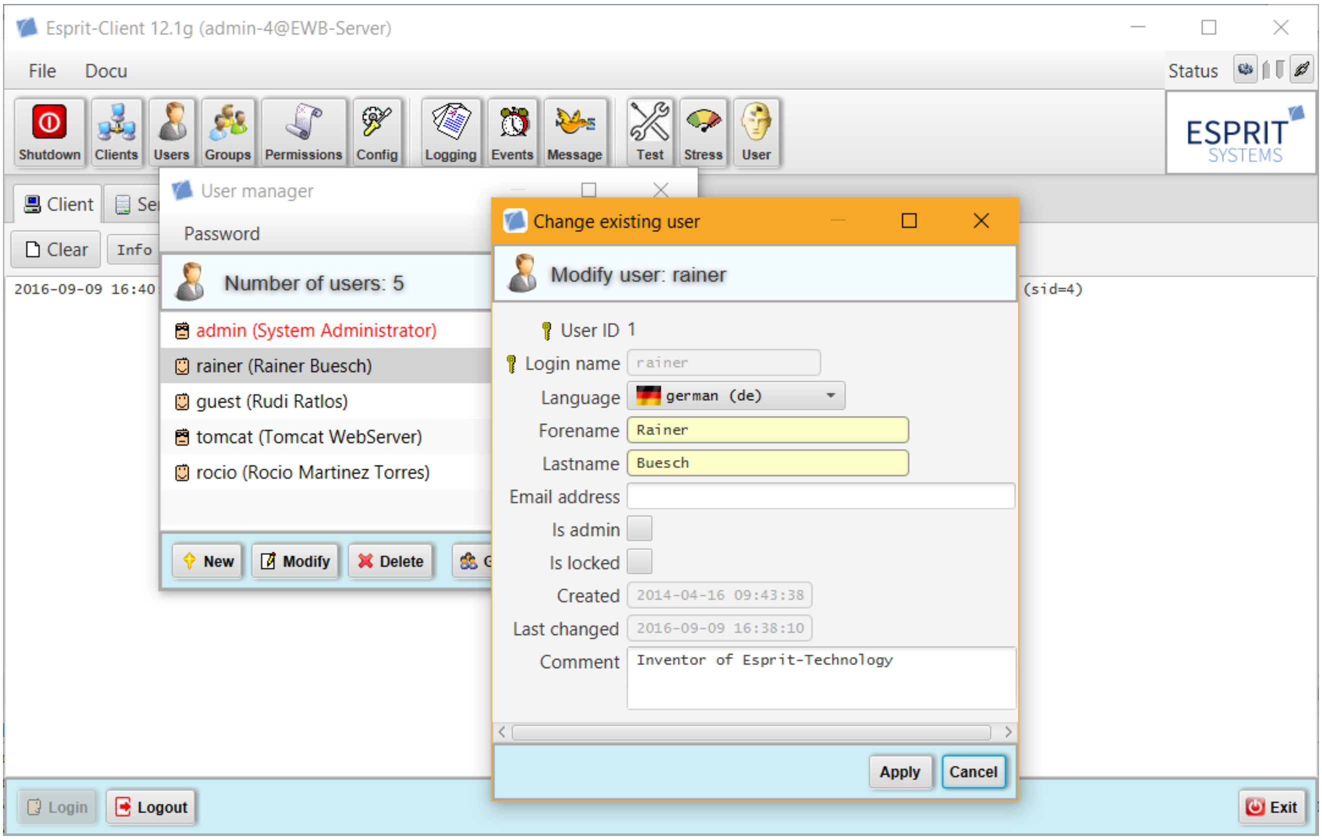
Rainer Buesch, Stuttgart Sept. 2016

Esprit Client/Server Technology

When we talk about "progress in networking" then we talk about Esprit-technology. This statement was made by the chief developer and architect of the INCA-Project at **BGR** (*Federal Institute for Geosciences and Natural Resources* in Hannover, Germany). This institute of the German government uses Esprit-software for more than 10 years for high challenging scientific applications. They use it because it has unique features, which no other software provides.

What's unique though?

In contrast to other request/response based client server systems the Esprit software fully realizes the Model-View-Controller principle over net. In practice this means: an Esprit-server is able to notify its clients instantaneously about any change that might have happened on its data. Thus, **all clients always have a consistent up to date view of the servers data** – without even issuing requests by their own. This technical concept opens a huge area of new possible applications. Client/server-computing has practically been reinvented with Esprit. It allows to realize client/server systems in a completely different design and with yet unknown dynamics. It's based on the newest Java Technology and it sets new standards in flexibility, performance, robustness, maintainability and costs.



2016-09-09 16:40

(sid=4)

Number of users: 5

- admin (System Administrator)
- rainer (Rainer Buesch)
- guest (Rudi Ratlos)
- tomcat (Tomcat WebServer)
- rocio (Rocio Martinez Torres)

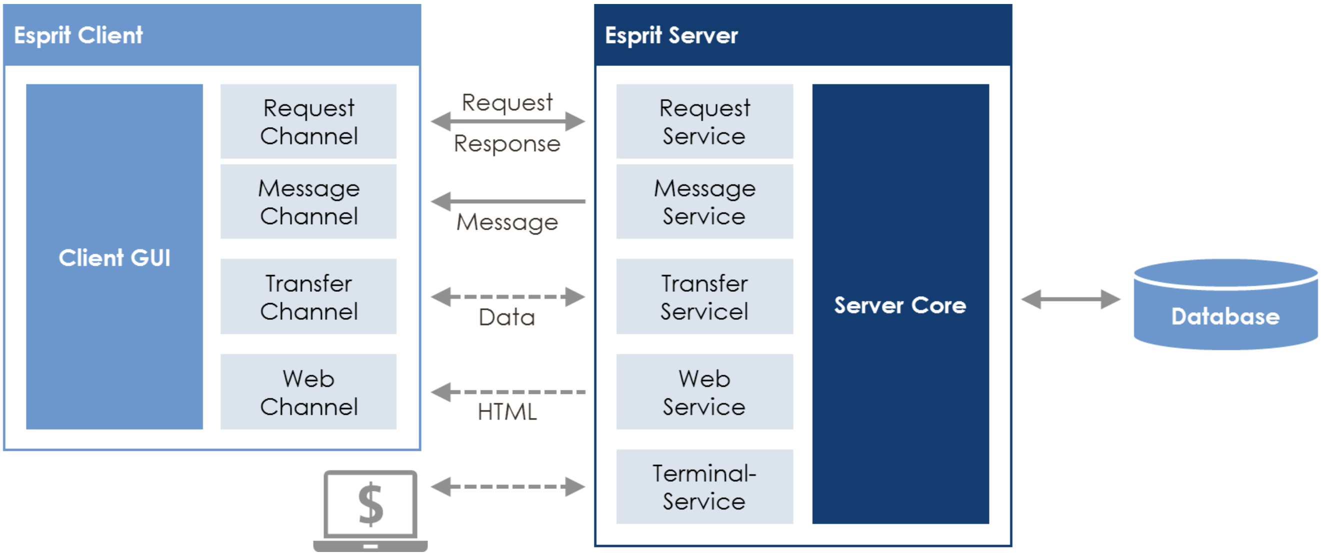
Modify user: rainer

User ID	1
Login name	rainer
Language	german (de)
Forename	Rainer
Lastname	Buesch
Email address	
Is admin	□
Is locked	□
Created	2014-04-16 09:43:38
Last changed	2016-09-09 16:38:10
Comment	Inventor of Esprit-Technology

The client user interface may be implemented in any GUI-technology. This picture shows the administration client for the Esprit-Server in a *JavaFX* implementation.

The Esprit Client/Server Concept

The concept is simple yet powerful: an Esprit-client connects to its server using a standing double channel connection. The first channel is just a request/response channel as usual. The second channel however is an instant-message channel, on which the client is listening for server-messages. A message can be anything: just an update of data that have changed on the server, but also a command that the server wants the client to execute. Thus, the server is able to control its clients.



- **Request/Response** channel for synchronous request/response
- **Instant-Message** channel for receiving asynchronous messages of the server
- **Transfer channel** (optional) for bidirectional transfer of mass-data
- **Web channel** (optional) for downloading HTML documents or software-updates
- **Terminal channel** (optional) for administering the server from a local terminal

Network Programming

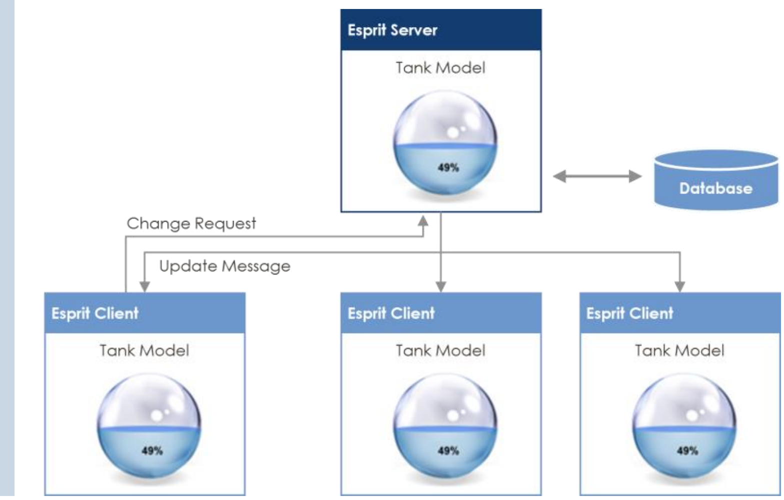
All client/server communication is done via the Esprit **SPI** (**S**erver **P**rogramming **I**nterface). Using this is as simple as programming against a standard Java API with the important difference, that method-calls are actually executed remotely on the server. This technique makes the network completely invisible to the user.
SPI supports **synchronous** as well as **asynchronous** remote method calls. In the latter case the client does not block but instead receives the response as a server-message. In this way, one client may run many asynchronous jobs on the server in parallel.

SPI is simple yet powerful. It's extremely robust, easy to use and easy to extend to whatever client/server functionality you need.

The Esprit Client/Server Concept

Model-View-Controller (MVC) over Net

The ***M**odel-**V**iew-**C**ontroller* principle is a commonly used pattern in local application user interfaces. Thanks to the *instant-message service* of the *Esprit*-server this pattern is now available **over net**, opening enormous possibilities.
Let's consider for instance a *tank-model*, which exists exactly once on the server. When a ***C**ontroller*-client changes the ***M**odel*, then the *Esprit*-server sends an update-message to all clients, which will update their ***V**iew* instantaneously - without issuing any additional request by their own.
In this way, server and client achieve an **ever consistent view of data** with an excellent performance, because only delta-information is passed. The old way of *client-polling* is a no-go in the *Esprit*-world.



This ability makes the Esprit-system ideal for everything that has to do with observation, like vehicle tracking , flow-control and measuring. The BGR-geologists for instance simulate below-earth movements, using the Esprit system for controlling and observing huge simulation processes.

Remote Property Binding

A server can maintain any type of data *(Objects, Lists, Maps)* in so called *server-properties*. Any change in these data will be notified to all clients that have accessed that particular server-property. Thus, the client view is kept up to date fully automatically. Such properties are easy to bind to GUI-components. This concept harmonizes perfectly with the property-binding mechanisms recently introduced by *JavaFX*.

Down-Requests and Client-Commands

An Esprit-server can direct a so called *down-request* to a client – just changing the role of who is the requestor and who is the responder. Down-requests are used in order to get information from the client at any time. But they can also be used to give orders to the client what to do. For instance a server might tell the client: I've got some results for you, just come and download them - or whatever you can imagine...

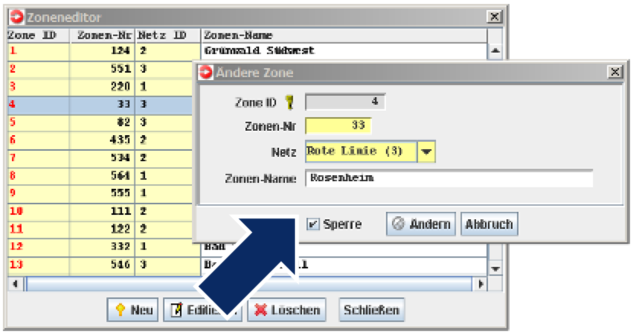
The Esprit Client/Server Concept

Server Resource-Locks

A server typically provides client-access to different types of data, i. g. files or database records. But if several clients want to change that data at the same time, then the server needs to synchronize this access in order to prevent data corruption. The Esprit-server uses so called resource-locks for this purpose. When for instance a client holds an *EXLUSIVE* lock on a file, then no other client will be able to change it.

Locks are **leased**, which means they need to be retriggered periodically in order to stay alive. If this does not happen in time by whatever reason, then the server will release the lock automatically.

A client may also acquire a *SHARED* lock, which means that the file cannot be deleted by another client while locked. The concept of resource-locks is a very flexible synchronization mechanism which applies to any kind of data.



Zoneneditor

Zone ID	Zonen-Nr	Netz ID	Zonen-Name
1	124	2	Grünwald Südwest
2	551	3	
3	220	1	
4	33	3	
5	82	3	
6	435	2	
7	534	2	
8	564	1	
9	555	1	
10	111	2	
11	122	2	
12	332	1	Bad
13	546	3	B

Ändere Zone

Zone ID: 4

Zonen-Nr: 33

Netz: Rote Linie (3)

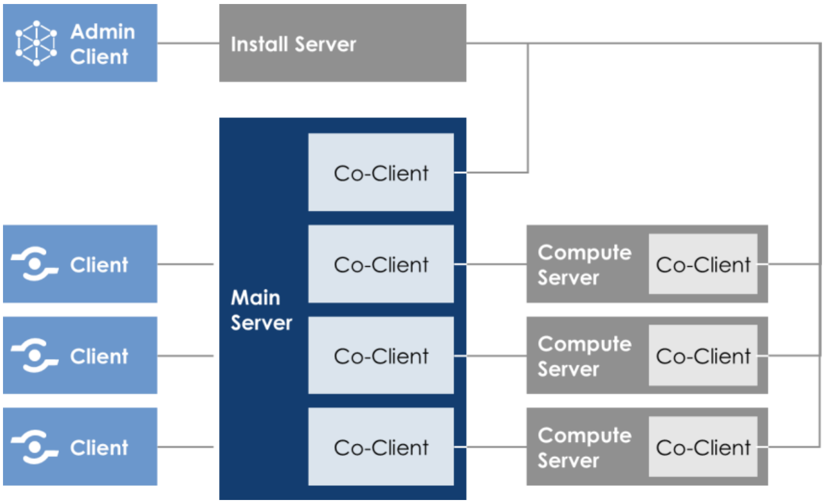
Zonen-Name: Rosenheim

☑ Sperre

This picture shows an automatically generated input-form for a database record. Clicking the lock-checkbox prevents other users to edit the same record.

Server-Networks

An Esprit-server may be by itself a client of one or several other Esprit-servers (*co-client*). So, servers can be interconnected in complex networks. Requests may be passed from server to server where they execute different codes on each of them, thus collecting data before being sent back to the requesting client. Inter-server-connections are fail-safe: interruptions - due to network failures - repare themselves as soon as the network is available again.



This picture shows a real configuration with interconnected Esprit-servers. The install server is used to keep the software of all involved systems automatically up to date.

The Esprit Client/Server Concept

Transactional Data Transfer

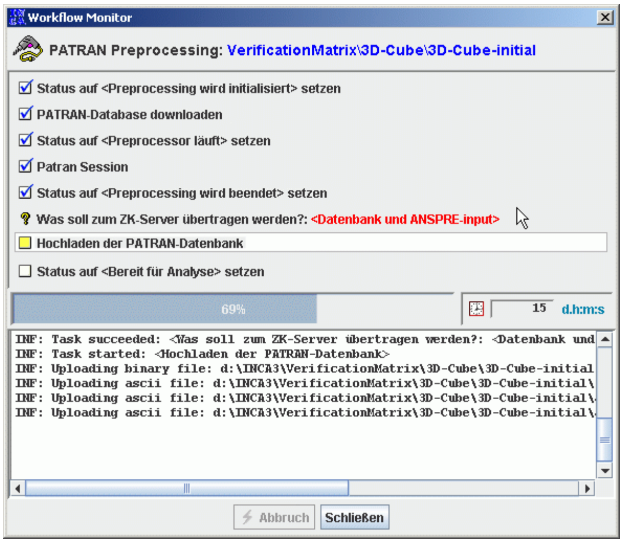
The Esprit-server supports bidirectional exchange of mass-data (huge files or file-sets) between client and server. A data transfer may happen synchronously or asynchronously and it is transactional, which means: *"all or nothing"*. The server uses a special port for transfers so that it will not enter in conflict with normal client-server operations. An additional stream-download service allows for down-streaming and processing data on the fly which results in even better performance than just copying files over the network.

Async-Task Framework and Workflows

One of the most powerful features of the Esprit-technology is its ability to handle all kinds of asynchronous tasks. A set of such tasks composes a so called *workflow*, which defines a procedure to be executed in a series. A workflow's execution is being monitored as shown in the picture. You can observe how the executed steps are check-marked and how the progress bar is moving.
A workflow may contain a mixture of tasks that execute locally at the client or remotely on the server. It may include just everything that can "run", such as OS-processes, file-transfers, stream-downloads etc.
Although executing asynchronously a workflow can interact with the user whenever it needs additional information or a user's-decision. In the picture you see a step (?-icon) where the user was asked for a decision which dynamically changed further processing by inserting an additional task.

A sample workflow could be:

- Start a CAD system on the client in order to create input data
- Transfer the input data to the central main-server
- Transfer the input data further to a compute-server and start a simulation there
- As simulation results appear transfer them to the main-server
- Download result data from the main-server to the client
- Start a CAD analysis tool on the client for displaying the simulation results



PATRAN Preprocessing: VerificationMatrix\3D-Cube\3D-Cube-initial

☑ Status auf <Preprocessing wird initialisiert> setzen

☑ PATRAN-Database downloaden

☑ Status auf <Preprocessor läuft> setzen

☑ Patran Session

☑ Status auf <Preprocessing wird beendet> setzen

? Was soll zum ZK-Server übertragen werden?: <Datenbank und ANSPRE-input>

□ Hochladen der PATRAN-Datenbank

□ Status auf <Bereit für Analyse> setzen

69%

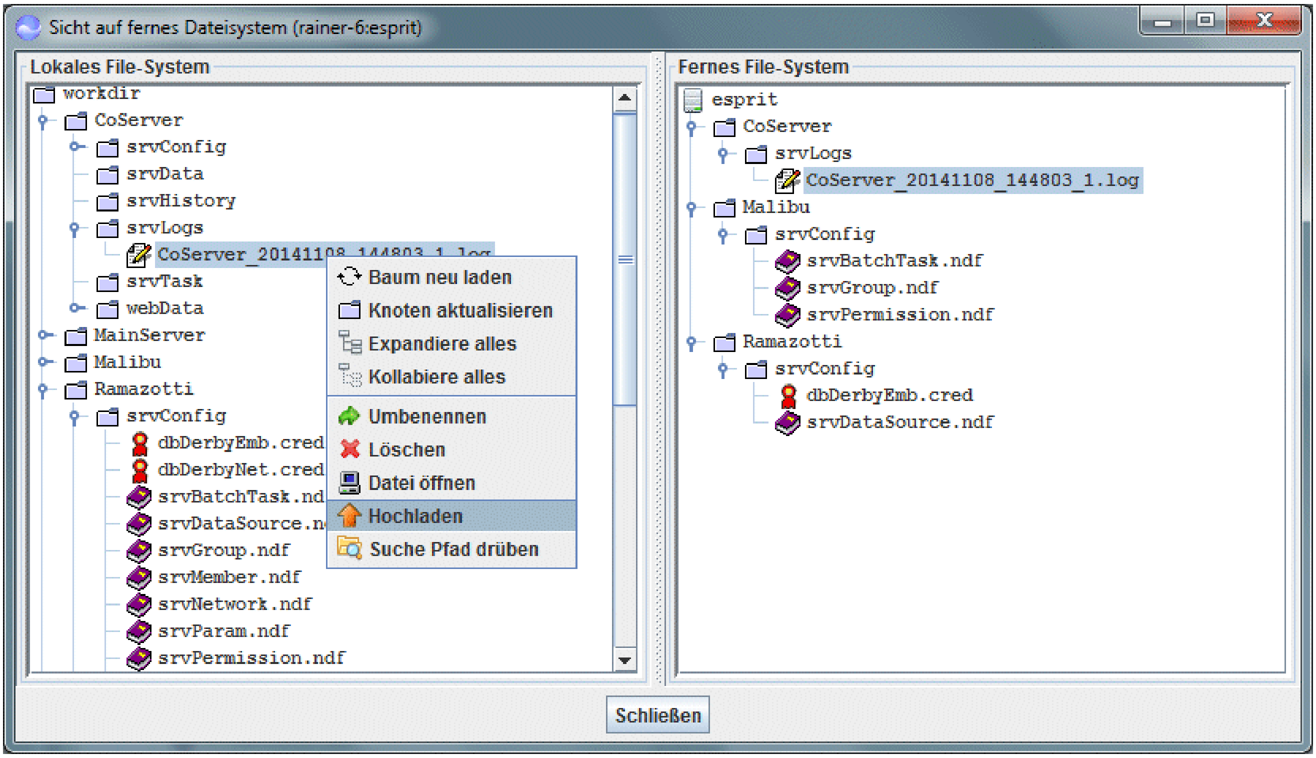
15 d.h:m:s

```
INF: Task succeeded: <Was soll zum ZK-Server übertragen werden?: <Datenbank und
INF: Task started: <Hochladen der PATRAN-Datenbank>
INF: Uploading binary file: d:\INCA3\VerificationMatrix\3D-Cube\3D-Cube-initial
INF: Uploading ascii file: d:\INCA3\VerificationMatrix\3D-Cube\3D-Cube-initial\
INF: Uploading ascii file: d:\INCA3\VerificationMatrix\3D-Cube\3D-Cube-initial\
INF: Uploading ascii file: d:\INCA3\VerificationMatrix\3D-Cube\3D-Cube-initial\
```

The Esprit Client/Server Concept

Observing remote File Systems

“Observing” is Esprit's favorite feature and one thing you often want to observe is a remote file system.
The picture shows a client-side file-tree next to a file-tree on the server. You easily can find out differences and exchange files to get both sides consistent. User access to server side files is controlled by so called *path-permissions*. A user can only do what he is permitted to do.
Depending on his path-permissions a user may only perform certain actions or even only see a part of the tree. This system is ideal for maintaining central project-trees, where users with different roles are acting on.



The Esprit Client/Server Concept

Team Communication

The built in instant-message-service of the Esprit-server makes it possible that users can communicate directly with each other. This is extremely useful when for example developers work in a team on the same project. Users may chat among each others or maintain a common task-list. The team leader may send popup-messages to particular users or a group of users – you can realize all types of communication you need in order to increase the team productivity.

Server Background Processes

The Esprit-server supports scheduled background task execution (similar to UNIX cron). Customers use this feature for nightly database updates or periodic data extraction. Like everything else within Esprit such background tasks are observable for clients.

Persistency using Neutral Data Format (NDF)

The Neutral Data Format **(NDF)** is a file-format for storing and archiving any kind of textual data. In contrast to XML it is *really* human readable and extremely compact with only a minimal overhead of meta data. It supports all data structures known in information science such as lists, tables, arrays and objects. Esprit provides a high performance *writer* and a *parser* for this format. Especially the parser is highly tuned for processing mass-data and beats XML-parsing by far as proven in a university-study.

```
@TABLE Groups {
       [gid groupName        displayName       comment ]
        1   "guests"         "Gäste"           “Has no permissions";
        2   "employees"      "Mitarbeiter"     "Has normal permissions";
        3   "administrators" "Administratoren" "Has all permissions";
}
```

This is an example for a table structure in NDF. This format is extremely compact, well readable and can be parsed significantly faster than XML. In the Esprit-world this format has fully replaced XML.

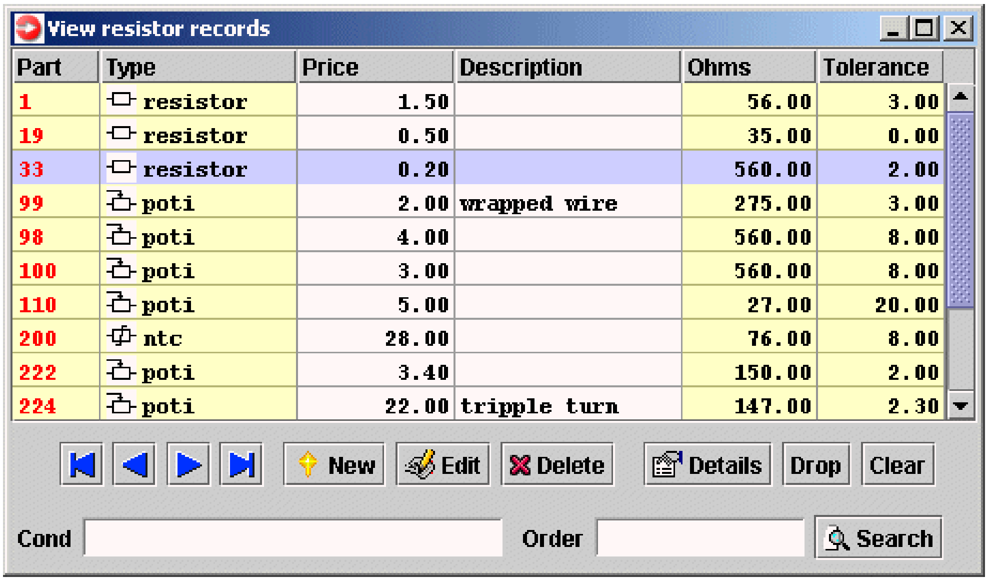
The Esprit Client/Server Concept

Persistency using DBObjects

Every software developer knows how costly it is to map relational database records to Java objects. Esprit's DBObjects do this automatically for you; a DBObject is is a Java-class which represents a database record. It carries the information of the record and knows by itself how to read/write its values from/to the database. DBObjects make the database persistency transparent to the user, who does not need not know anything about SQL any more.

DBObject classes are automatically created by a special compiler, which reads the meta information from the database system. So they are consistent with the actual database by construction. Whenever a database table has been changed, you just need a recompile and everything is consistent again within a few seconds. DBObjects are building blocks for constructing composed persistent objects with higher complexity.

An Esprit-server manages an arbitrary amount of connections to databases even from different vendors. Because DBObjects are compatible to all of them, you can easily exchange data between them – of course with arbitrary logic in between.



Part	Type	Price	Description	Ohms	Tolerance
1	resistor	1.50		56.00	3.00
19	resistor	0.50		35.00	0.00
33	resistor	0.20		560.00	2.00
99	poti	2.00	wrapped wire	275.00	3.00
98	poti	4.00		560.00	8.00
100	poti	3.00		560.00	8.00
110	poti	5.00		27.00	20.00
200	ntc	28.00		76.00	8.00
222	poti	3.40		150.00	2.00
224	poti	22.00	tripple turn	147.00	2.30

Automatic Client-Installation with Esprit-AppStore

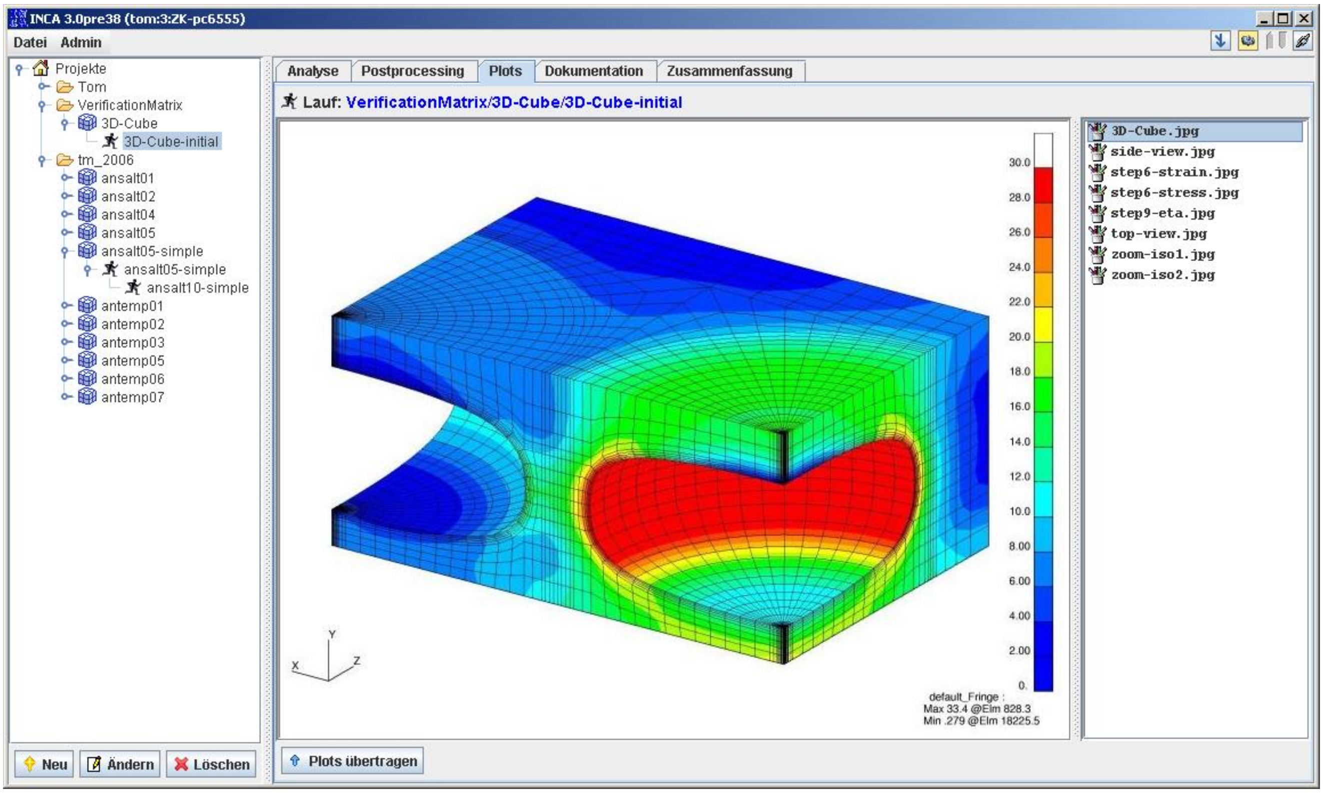
Esprit is **not** a web technology, rather a good old rich-client-technology. The latter one has significant advantages over web technologies, such as better load balance between client and server and effectively unlimited complexity at best possible performance. The only disadvantage has been until now that you needed to reinstall the client software whenever the server software has been upgraded. This disadvantage has completely disappeared due to the *Esprit-AppStore*, which is doing this job automatically under the hood. An Esprit-server keeps its clients up to date in all aspects – even with software-installation.

Custom Application Example 1

BGR
Bundesanstalt für **G**eowissenschaften und **R**ohstoffe
Federal Institute for Geosciences and Natural Resources

www.bgr.bund.de

This institute of the German government in Hannover uses Esprit-technology for executing complex finite element calculations of geologic systems, such as all kinds of below-earth-movements. Workflows are used to control distributed simulations on different interconnected compute-servers. Clients observe the simulation progress online by receiving event notifications from the server. This also includes the automatic transfer of result data to the clients, so that the user can already start analyzing them as soon as they are available.



Lauf: VerificationMatrix/3D-Cube/3D-Cube-initial

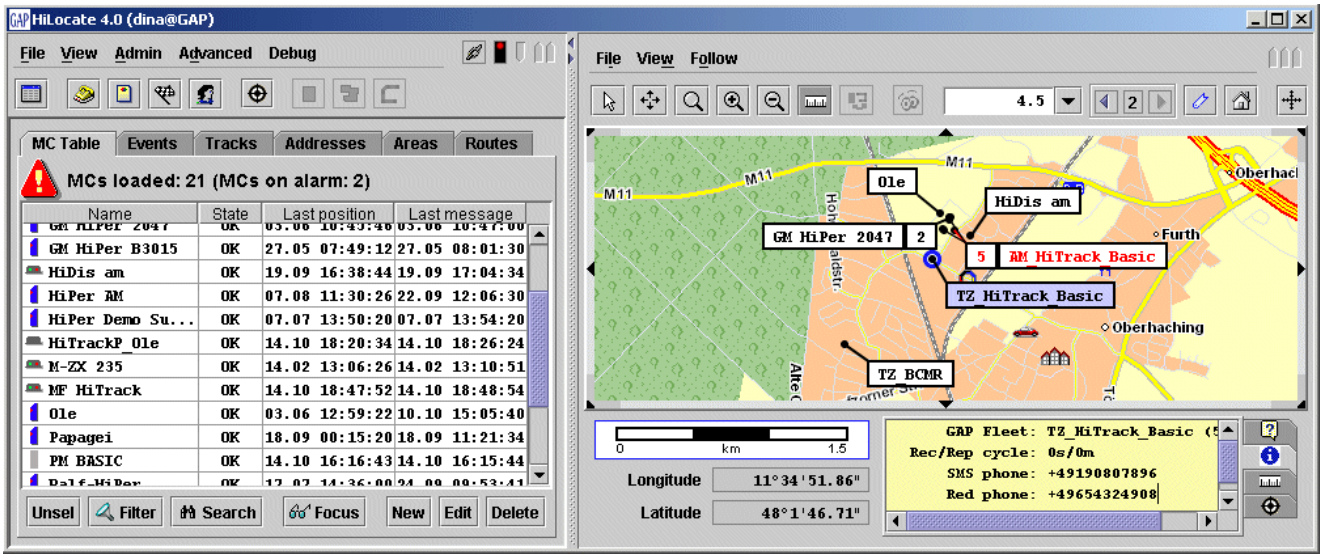
default_Fringe :
Max 33.4 @Elm 828.3
Min .279 @Elm 18225.5

Custom Application Example 2

ISA-Telematics

www.isatelematics.de

This company uses the Esprit-concept in their telematic system *HiLocate*, which allows for online tracking of vehicles on a virtual map. All vehicles periodically send their position information via SMS to a central server which in turn notifies its clients about the vehicle movement. This application has a huge set of features: so you may enter a follow-mode which dynamically fits the view to one or more vehicles. Furthermore you can define rules for vehicles such as mandatory routes or forbidden areas and you will get an alarm notification if a vehicle violates any limit. A huge performance gain was achieved by using parallel processing on client and server: by zooming to another section of the map the server will be calculating the new map-view while at the same time the client is processing the layout and placement of the flags which are used to label a single vehicle or a group of them.



MCs loaded: 21 (MCs on alarm: 2)

Name	State	Last position	Last message
GM HiPer 2047	OK	03.06 10:45:46	03.06 10:47:00
GM HiPer B3015	OK	27.05 07:49:12	27.05 08:01:30
HiDis am	OK	19.09 16:38:44	19.09 17:04:34
HiPer AM	OK	07.08 11:30:26	22.09 12:06:30
HiPer Demo Su...	OK	07.07 13:50:20	07.07 13:54:20
HiTrackP_Ole	OK	14.10 18:20:34	14.10 18:26:24
M-ZX 235	OK	14.02 13:06:26	14.02 13:10:51
MF HiTrack	OK	14.10 18:47:52	14.10 18:48:54
Ole	OK	03.06 12:59:22	10.10 15:05:40
Papagei	OK	18.09 00:15:20	18.09 11:21:34
PM BASIC	OK	14.10 16:16:43	14.10 16:15:44
Ralf-HiPer	OK	17.07 14:36:00	24.09 09:53:41

Longitude 11°34'51.86"

Latitude 48°1'46.71"

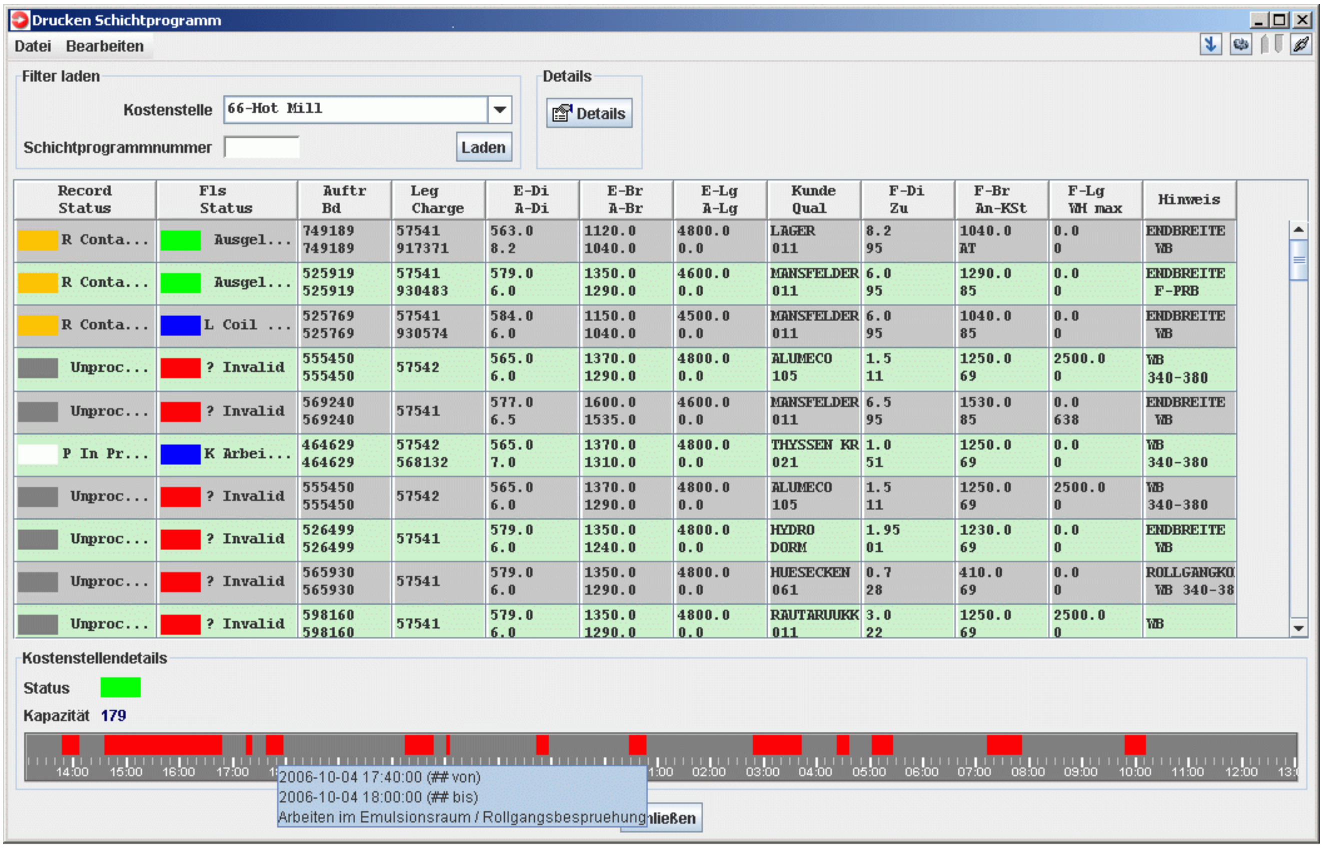
```
     GAP Fleet: TZ_HiTrack_Basic (5
Rec/Rep cycle: 0s/0m
    SMS phone: +49190807896
    Red phone: +49654324908
```

Custom Application Example 3

Hydro-Aluminiumwerke AG

www.hydro.com

This enterprise in Hamburg uses Esprit-technology to monitor production progress in their aluminium rolling mill. Every table row represents a work-place in the factory which has been modeled as an *Alive Business Object* (ABO). Such objects are kept up to date automatically as the state of the work-place changes. Another ABO is the walking timeline on the bottom which gives a life overview over the production process on the whole. The red spots indicate problems with certain production machines that require manual interaction.



Kostenstelle: 66-Hot Mill

Schichtprogrammnummer:

Record Status	Fls Status	Auftr Bd	Leg Charge	E-Di A-Di	E-Br A-Br	E-Lg A-Lg	Kunde Qual	F-Di Zu	F-Br An-KSt	F-Lg WH max	Hinweis
🟧 R Conta...	🟩 Ausgel...	749189 749189	57541 917371	563.0 8.2	1120.0 1040.0	4800.0 0.0	LAGER 011	8.2 95	1040.0 AT	0.0 0	ENDBREITE WB
🟧 R Conta...	🟩 Ausgel...	525919 525919	57541 930483	579.0 6.0	1350.0 1290.0	4600.0 0.0	MANSFELDER 011	6.0 95	1290.0 85	0.0 0	ENDBREITE F-PRB
🟧 R Conta...	🟦 L Coil ...	525769 525769	57541 930574	584.0 6.0	1150.0 1040.0	4500.0 0.0	MANSFELDER 011	6.0 95	1040.0 85	0.0 0	ENDBREITE WB
▨ Unproc...	🟥 ? Invalid	555450 555450	57542	565.0 6.0	1370.0 1290.0	4800.0 0.0	ALUMECO 105	1.5 11	1250.0 69	2500.0 0	WB 340-380
▨ Unproc...	🟥 ? Invalid	569240 569240	57541	577.0 6.5	1600.0 1535.0	4600.0 0.0	MANSFELDER 011	6.5 95	1530.0 85	0.0 638	ENDBREITE WB
⬜ P In Pr...	🟦 K Arbei...	464629 464629	57542 568132	565.0 7.0	1370.0 1310.0	4800.0 0.0	THYSSEN KR 021	1.0 51	1250.0 69	0.0 0	WB 340-380
▨ Unproc...	🟥 ? Invalid	555450 555450	57542	565.0 6.0	1370.0 1290.0	4800.0 0.0	ALUMECO 105	1.5 11	1250.0 69	2500.0 0	WB 340-380
▨ Unproc...	🟥 ? Invalid	526499 526499	57541	579.0 6.0	1350.0 1240.0	4800.0 0.0	HYDRO DORM	1.95 01	1230.0 69	0.0 0	ENDBREITE WB
▨ Unproc...	🟥 ? Invalid	565930 565930	57541	579.0 6.0	1350.0 1290.0	4800.0 0.0	HUESECKEN 061	0.7 28	410.0 69	0.0 0	ROLLGANGKO WB 340-38
▨ Unproc...	🟥 ? Invalid	598160 598160	57541	579.0 6.0	1350.0 1290.0	4800.0 0.0	RAUTARUUKK 011	3.0 22	1250.0 69	2500.0 0	WB

Status 🟩

Kapazität 179

2006-10-04 17:40:00 (## von)
2006-10-04 18:00:00 (## bis)
Arbeiten im Emulsionsraum / Rollgangsbespruehung

Esprit Feature Overview

- True client-sessions, based on a double channel connection
- Use Esprit **SPI** (**S**erver **P**rogramming **I**nterface) for robust client/server communication .
- Support for synchronous and asynchronous requests
- Asynchronous messaging using the Esprit instant message service
- Model-View-Controller principle is fully implemented over net
- Support for local /remote tasks, controlled and monitored by workflow's
- High performance file persistency using NDF-format
- High performance database persistency using DBObjects
- Support for transactional file transfers and stream downloads
- Support for resource-locks for concurrency control
- Ability to build server networks using inter-server-connections
- Time controlled background jobs (cron like)
- High sophisticated management of users, groups and permissions
- Support for server-to-client down-requests. Server controls its clients
- Clients can directly communicate among each other
- Life monitoring of server-activity/statistics via terminal-connection
- Client may work in online- or offline-mode as needed
- Support for connecting to many databases of different vendors with direct data-exchange between them
- Multi-language support for client GUIs
- Automatic software installation/upgrade via Esprit-AppStore reduces maintenance to zero

Your Advantage

- For applications up to 1000 users Esprit may be an excellent alternative to a monstrous application server
- An Esprit-application may be a suitable rich-client alternative to a web application when complexity raises
- The Esprit client/server system is build on top of a software framework that contains decades of experience
- Esprit provides a sophisticated concept for controlling and monitoring remote processes
- Esprit serves as a central place for integrating all your tools
- Esprit is implemented in newest Java technology and runs on any platform
- Esprit applications have best possible performance because only delta-information is exchanged between client and server
- Esprit applications have excellent client/server load balance. Others typically have a high load on the server side
- Old programs can relatively easy be rewritten in Esprit-technology, gaining client-server ability as a spin-off

Advantages for developers

- You only need to develop, what you actually need - the network remains hidden to you
- You can develop client/server functionality of highest complexity and robustness
- The source code of your project remains clean, well readable, testable and maintainable
- Esprit software is extremely well tested, mature and robust. You won't loose much time with debugging
- Esprit software is easy to learn. No special know-how is required except standard Java
- Esprit applications are extremely compact and memory efficient – only a few megabytes on the whole.
- Having the source code available gives you safety and independence – and you can see how things are done
- You need not include any foreign software into your project. The Esprit-software has all you need

Conclusion

Esprit is a pure Java client/server framework for rich client support. The Esprit-server's unique instant-message service allows for fully realizing the model-view-controller principle over net. Therefore the server is capable to keep all its clients up to date with its data – avoiding inconsistencies by concept. Custom applications use this system mainly for observation such as car tracking, measurement or process-management.

The German government (BGR) uses Esprit-software in safety critical applications. Therefore it fulfills highest requirements in quality and robustness. It has grown up within more than 15 years, has achieved a high grade of maturity and is continuously being tested with high coverage.

For more information look at: www.esprit-systems.de

Freepik Design