ESPRITSYSTEMS

Esprit Client/Server Technologie

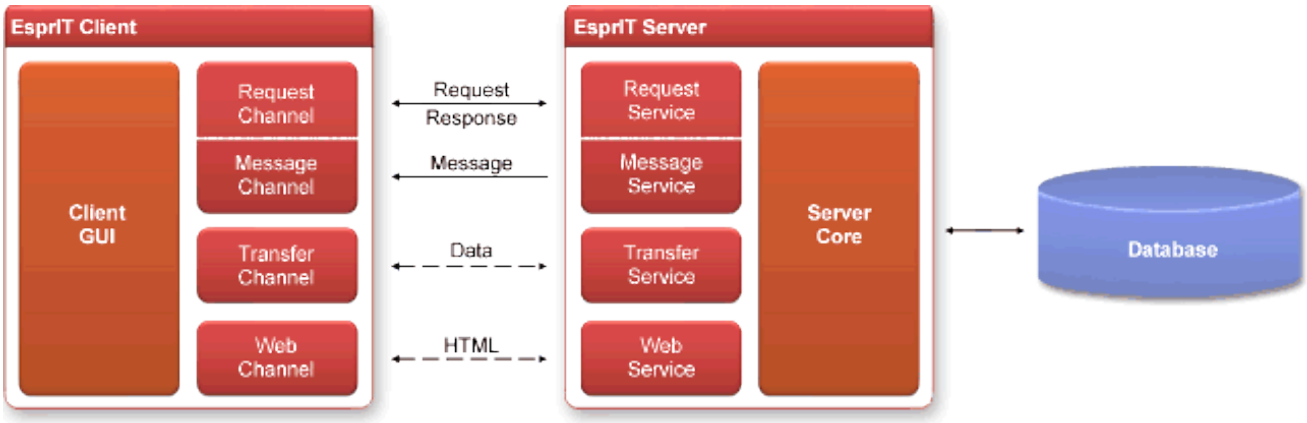
Um eine Software *revolutionär* zu nennen, muss sie schon so etwas wie ein *Alleinstellungsmerkmal* besitzen. Wir glauben, dass die ***Esprit-****Technologie* genau dieses Kriterium erfüllt, denn im Gegensatz zu den bisher üblichen *Request/Response* orientierten Systemen ist ein *Esprit*-Server in der Lage, seinen Clients jederzeit asynchrone Nachrichten zu schicken. Dadurch eröffnen sich eine Vielzahl von neuen Möglichkeiten und Anwendungsfeldern.

Ein *Esprit*-Server benachrichtigt beispielsweise seine Clients, wenn ein Datensatz in der Datenbank geändert wurde. Sämtliche Clients erhalten so – ohne eigenes Zutun – die neue Information und sind daher stets auf dem aktuellsten Stand. *Esprit*-Applikationen besitzen dadurch eine ganz besondere Dynamik und etwaige Inkonsistenzen werden von vorne herein vermieden.

Mit der *Esprit-Technologie* wurde das Client/Server-Computing praktisch neu erfunden. Sie ermöglicht die Entwicklung von Client/Server Anwendungen auf eine ganz neue Art. Da sie auf modernster Java-8 Technologie basiert, ist sie zudem plattformunabhängig. *Esprit* setzt Maßstäbe im Bezug auf Flexibilität, Performance, Robustheit, Wartbarkeit, Einfachheit und Preis!

Neuartiges Client/Server Konzept

Das Konzept ist einfach - und darin liegt die Genialität: Den Kern der *Esprit-Technologie* bildet der *Esprit*-Server, ein innovatives Middleware-System, speziell zur Unterstützung von Rich-Clients. Im Gegensatz zu herkömmlichen Clients verbindet sich ein *Esprit*-Client über eine Doppelkanal-Verbindung mit dem Server.



- Ein *Request/Response* Kanal für synchrone Anfragen
- Ein *Instant-Message* Kanal zum Empfang asynchroner Server-Nachrichten

In einem *Request* kann ein Client dem Server auch mitteilen, dass er die *Response* als Nachricht erhalten möchte. Auf diese Weise sind sehr einfach auch asynchrone Requests möglich. Beide

Netzwerkkanäle werden gemeinsam als eine *Session* verwaltet. Da es sich um stehende Verbindungen handelt, ist die Performance außerordentlich gut. Zwei zusätzliche Netzwerk-Kanäle können bei Bedarf genutzt werden:

- Ein *Transfer* Kanal zur bidirektionalen Übertragung von Dateien
  z. B. für den Austausch von Massendaten
- Ein *Web* Kanal zum Abrufen von HTML Dokumenten,
  auch zur dynamischen Ausliefern der Client-Software über Webstart.

Esprit Netzwerk Schnittstelle EMI

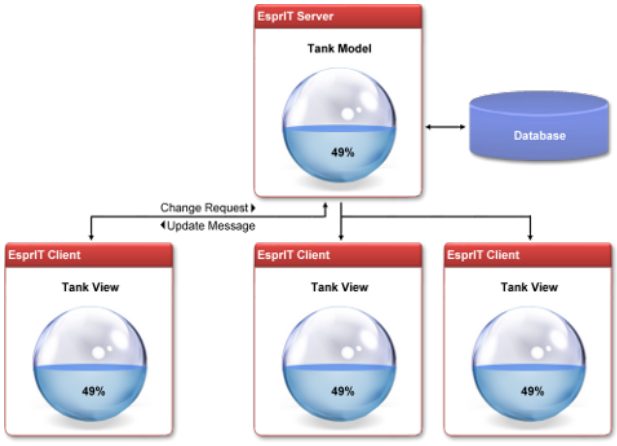
Die Kommunikation zwischen Client und Server geschieht über eine spezielle **EMI**-Schnittstelle (**E**sprit **M**ethod **I**nvocation). Diese funktioniert ähnlich, wie Java **RMI**, allerdings über eine stehende Verbindung. Das Besondere bei EMI ist, dass ein Client Requests an den Server absetzen kann, aber auch umgekehrt, der Server Requests an die Clients tätigen kann. Die Client→Server bzw. Server→Client Aufrufe sehen dabei wie ganz gewöhnliche Java Methodenaufrufe aus. Das Netzwerk ist für den Programmierer vollständig unsichtbar.

Eine weitere Besonderheit ist, dass Requests asynchron sein können. Dies bedeutet, dass der Client asynchrone Tasks auf dem Server startet und vom Server laufend über deren aktuellen Stand informiert wird. Ist ein solcher Task fertiggestellt, dann erhält der Client automatisch das Ergebnis zugeschickt und kann entsprechend reagieren.

Die Netzwerk-Funktionalität ist mit **EMI** extrem leicht erweiterbar. Dazu muss von einer Serverkomponente lediglich ein Interface implementiert werden. Die clientseitige Implementierung davon wird dynamisch - bei Bedarf - automatisch generiert.

Model-View-Controller (MVC) über Netz

Das *Model-View-Controller* Prinzip (MVC) wird in der GUI-Programmierung gerne verwendet, da es äußerst robust funktioniert, nur war es leider bisher auf lokale Anwendungen beschränkt. Mit dem integrierten *Instant-Message Service* des Esprit-Servers ist es nun möglich, dieses Prinzip auch über Netz zu realisieren. Die sich daraus ergebenden Möglichkeiten sind enorm.



Den Kern des Prinzips bildet ein serverseitiges Modell, das gewisse Eigenschaften speichert. Wenn ein Client (*Controller*) eine Eigenschaft per Request verändert, dann schickt der *Esprit*-Server eine entsprechende Event-Nachricht an *alle* Clients, die ihrerseits ihre lokale Sicht (*View*) aktualisieren.

Ein Client muss also keinen Aktualisierungs-Request durchführen, um up-to-date zu bleiben, vielmehr wird er über Änderungen vom Server dynamisch benachrichtigt. Auf diese Weise können Inkonsistenzen in der Client-Sicht gar nicht erst entstehen. Bisher übliche *Polling* Verfahren erübrigen sich, wodurch dem Server eine erhebliche Last erspart bleibt.

Remote Data Binding

Das sog. *Data-Binding* wird häufig in Benutzeroberflächen eingesetzt, wo es darum geht, dass eine GUI-Komponente stets den aktuellen Stand einer Variablen repräsentiert. Dies wird erreicht durch einen Event-Mechanismus, bei dem auf Änderungen „gehorcht“ und dann entsprechend regagiert wird.

Der Esprit-Server unterstützt sog. *Remote Properties*. Dabei ist jeweils eine **Server-Property** an eine korrespondierende **Client-Poperty** „gebunden“, d. h. die Server-Property hält die Client-Property per Event-Notifikation stets auf dem aktuellsten Stand. Die ClientProperty ihrerseits ist an eine GUI-Komponente „gebunden“, die ihren Zustand visualisiert.

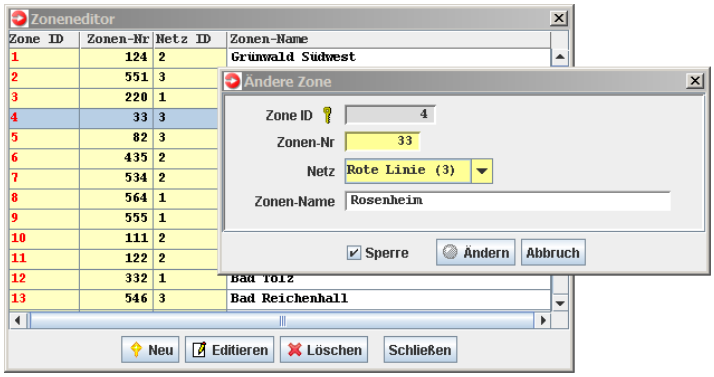
Damit ist die GUI-Komponente praktisch direkt an einen Wert, der auf dem Server verwaltet wird, gebunden (auf Wunsch auch bidirektional). Server-Properties können beliebige Werte verwalten, wie Objekte, Listen oder Maps. Mit *Remote Properties* ist es extrem einfach, zentrale serverseitige Daten allen Clients zur Ansicht und/oder zur Änderung bereitzustellen.

Down-Requests (Kommandos an Clients)

Die Esprit-Technik ermöglicht es, Anfragen nicht nur an einen Server, sondern auch an andere Clients zu schicken. Mit sog. *Down-Requests* kann ein Client auch per Kommando ferngesteuert werden, was eine Fülle von Möglichkeiten eröffnet. So kann der Server beispielsweise veranlassen, dass ein Client sich gerade fertig gewordene Ergebnisse automatisch abholt. Auch Rundfragen an alle Clients sind auf diese Weise leicht zu realisieren.

Server Resource-Sperren

Will ein Client eine Server-Ressource (z. B: Datei auf Serverseite oder einen Datensatz) verändern, dann kann er dafür eine serverseitige Exklusiv-Sperre belegen. Dies ermöglicht ihm ein vor konkurrierenden Zugriffen geschütztes Editieren. Aus Sicherheitsgründen sind diese Sperren zeitlich begrenzt (Leasing). Deshalb werden sie vom Client periodisch nachgetriggert, damit sie nicht vorzeitig automatisch verfallen. Clients, die derartige Sperren zu lange halten, ermahnt der Server. Sollte der Client nach einem gewissen Timeout weiterhin nicht „loslassen“, wird die Sperre zwangsweise freigegeben.

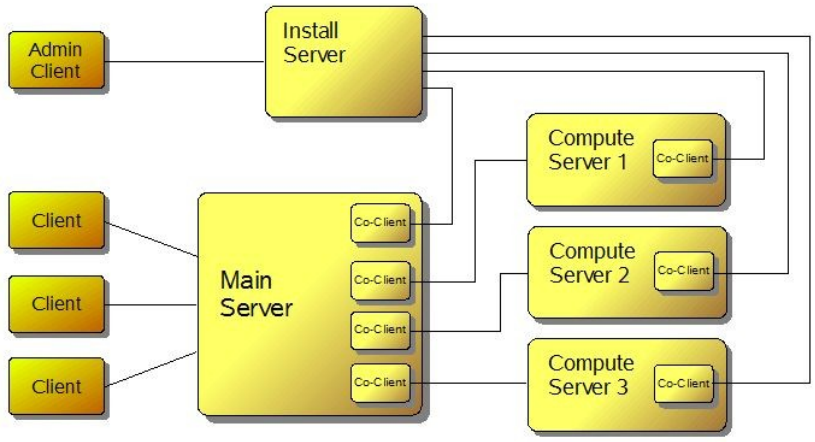


Dieses Bild zeigt ein automatisch generiertes Eingabeformular für einen Datensatz. Mit dem Sperr-Flag wird der Datensatz für andere Benutzer temporär zum Editieren gesperrt.

Server-Verbundnetze

Ein *Esprit*-Server kann selbst Client von einem oder mehreren anderen *Esprit*-Servern sein. So lassen sich viele Server in einem Netzverbund zusammenschalten. Requests können dann von Server zu Server "durchgereicht" werden, wobei sie auf jedem Server unterschiedlichen Code ausführen. Dabei sammeln sie Ergebnisse auf, bevor sie zum Client zurückgeschickt werden.

Die Server-zu-Server Verbindungen sind fail-safe: Fällt ein Rechner aus, wird dies von den anderen „gemerkt“. Die betroffenen „Server-Kollegen“ versuchen periodisch, die Verbindung wieder aufzubauen. Ein neu gestarteter Server „weiß“ per Konfiguration wie er sich wieder ins Netzwerk einklinken muss. Alle seine Inter-Serververbindungen sind dann sofort wieder online.



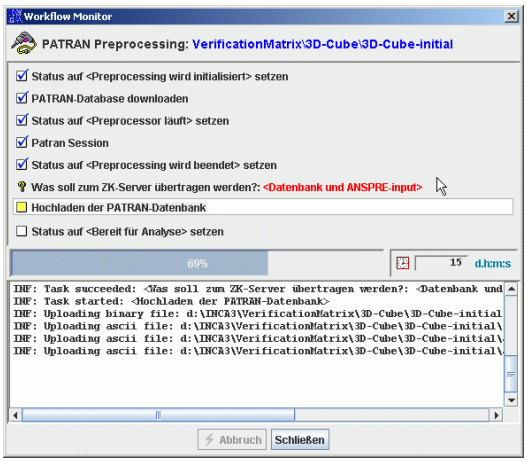
Dieses Bild zeigt eine reale Konfiguration mit untereinander vernetzten Servern. Alle Rechner werden automatisch vom Installationsserver softwaremäßig auf dem aktuellsten Stand gehalten.

Transaktionaler File-Transfer Service

Der *Esprit*-Server unterstützt den bidirektionalen Transfer von Massendaten (Dateien bzw. Datei-Sets) zwischen Client und Server sowie zwischen untereinander gekoppelten Servern. Die Übertragung erfolgt transaktional, d.h. nach dem Motto „alles oder nichts“. Sie wird über einen speziellen Transferkanal abgewickelt und stört deshalb nicht die normale Interaktion zwischen Client und Server. Wahlweise können die Daten synchron oder asynchron, auf Wunsch auch in komprimierter Form übertragen werden. Ein zusätzlicher *Stream-Download Service* erlaubt dem Client eine Serverdatei direkt aus einem Netzwerk-Stream zu lesen und zu verarbeiten, ohne sie auf den Client zu kopieren. Dies ist ggf. deutlich performanter als ein einfacher File-Transfer.

Client Workflows (mit Monitoring)

Eine der wesentlichen Stärken der *Esprit*-Technologie besteht darin, dass eine Serie von asynchronen Operationen gekoppelt ablaufen können. Solche Operationen (*Tasks*) können zu einem *Workflow* zusammenstellt werden, der dann die Ablaufsteuerung und Kontrolle übernimmt. Ein *Workflow-Monitor-Dialog* listet alle Tasks in einer Checkliste auf, visualisiert den Fortschritt mit einem Laufbalken und hakt die erledigten Tasks ab.



Ein Workflows kann auch per Dialog eine Benutzer-Eingabe anfordern, wenn z.B. entschieden werden muss, ob ein Task überschlagen, oder ein anderer eingefügt werden soll.

Ein einmal erstellter Workflow kann sowohl **clientseitig** als auch **serverseitig** ablaufen.

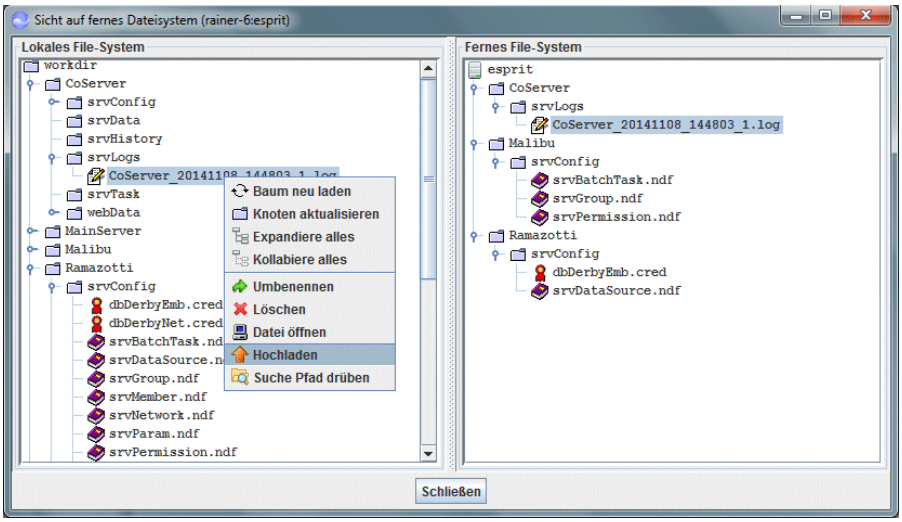
Die Taskliste dieses Workflows zeigt einen Verzweigungspunkt (Fragezeichen), wo der Benutzer zu einer Entscheidung aufgefordert wurde. Als Ergebnis seiner Entscheidung wurden dann die zwei nachfolgenden Tasks dynamisch eingefügt.

Ein Beispiel-Workflow könnte so aussehen:

- Starte CAD System zur Erstellung der Eingabe-Datei
- Übertrage die Eingabe-Datei auf den Zentral-Server
- Starte auf dem Compute-Server den Rechenlauf
- Übertrage die Ergebnisdateien auf den Client
- Starte CAD-System zur Anzeige der Ergebnisse

Man beachte, dass Workflows alles einschließen, was „laufen“ kann, z.B. auch lokale oder remote Betriebssystemprozesse (Rechenprogramme, CAD-Systeme, etc.), Datei-Übertragungen u.v.m.

Entfernte Filesysteme beobachten



Zu den besonders nützlichen Funktionalitäten des Esprit-Systems zählt der *Remote-FileTree*. Gemeint ist die Beobachtung von Filesystemen auf dem eigenen sowie auf anderen Rechnern im Netzwerk.

Das nebenstehende Bild zeigt links den lokalen, clientseitigen Dateibaum und rechts das Filesystem des Servers. Dateien können nun sehr leicht zwischen Client und Server ausgetauscht bzw. synchronisiert werden, mit der Besonderheit, dass jede Änderung im Server-Filesystem sofort auf allen anderen Clients sichtbar ist.

Die Zugriffe der Clients auf das ferne Server-Filesystem sind Rechte-gesteuert. Ein Client sieht nur die Teile des Baumes, auf denen er Leserecht hat. Er kann Dateien zum Server nur hochladen, wenn er das Änderungsrecht im Zielverzeichnis hat. Der Administrator kann Pfad-Rechte auf Benutzer- oder Gruppenebene zuweisen. Eine Applikation kann beliebige Pfad-Rechte frei erfinden, wie z.B. ein EXTRACT-Recht für Zip-Dateien u.v.m.. Client- und Server-Filesystem haben immer die gleiche Directory-Struktur, die auch durch Up- und Downloads nicht verletzt werden kann. Diese Funktionalität eignet sich z. B. hervorragend zur zentralen Verwaltung von Projekten.

Benutzer Kommunikation

Aufgrund des im *Esprit*-Server eingebauten *Instant-Message-Service* können Benutzer auch direkt untereinander kommunizieren, eine Möglichkeit, die z. B. von dem eingebauten *Chat-Tool* genutzt wird. Des weiteren kann der Administrator Info-Nachrichten an die Benutzer schicken. Ebenfalls enthalten ist ein Mail-Service, der Administratoren per Email über besondere Ereignisse benachrichtigt.

Server Hintergrundprozesse

Der Administrator kann zeitgesteuerte Hintergrundprozesse einrichten, jeweils mit der Angabe, wann und mit welcher Wiederholungsrate sie auszuführen sind. Ein spezieller Thread des Servers führt sie dann zum gegebenen Zeitpunkt aus (ähnlich Unix *crontab*). Ein laufender Hintergrundprozess sendet dafür registrierten Clients regelmäßig Fortschrittsmeldungen, so dass diese in der Lage sind, den Ablauf in einem Laufbalken zu verfolgen. Hintergrundprozesse können vom Administrator dynamisch umkonfiguriert oder auch manuell gestartet werden.

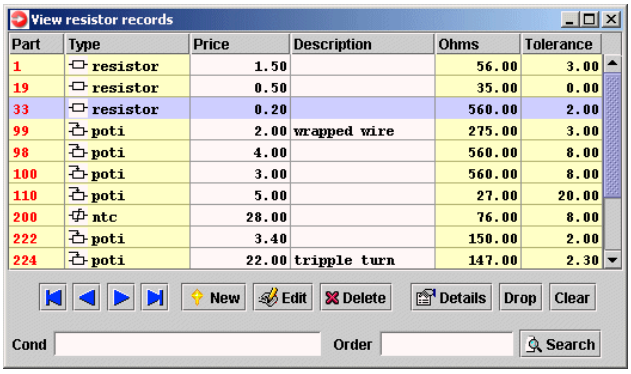
Persistenz mit dem *Neutral Data Format* (NDF)

Das *Neutral Data Format* (**NDF**) ist ein Datei-Datenformat zur Speicherung und Langzeitarchivierung von beliebigen Informationen. Anders als bei XML ist NDF ein *wirklich* leicht lesbares, sehr kompaktes Format mit minimalem Overhead an Meta-Information. Es unterstützt alle typischen Datenstrukturen der Informatik, wie Listen, Tabellen, Arrays und Objekte. Erzeugt wird es mit Hilfe des *NdfWriters*. Das Einlesen geschieht mit Hilfe des Event-gesteuerten *NdfParsers*. Writer und Parser sind extrem auf Performance getunt und damit auch tauglich für die Verarbeitung von Massendaten.

Persistenz mit *DBObjects*

Jeder Programmierer weiß, wie aufwendig es ist, die in einer relationalen Datenbank in atomisierter Form gespeicherten Daten in Objekte so abzubilden, wie man sie in der Programmierwelt benötigt. Das *Esprit*-Framework beinhaltet die *DBObject Suite,* eine besonders leistungsfähige Persistenzschicht, die Ihnen die Arbeit des *Object-Relational-Mappings* vollständig abnimmt.

Ein *DBObject* ist eine Java Klasse, die einen Datensatz modelliert. Es trägt die Information des Datensatzes und weiß selbst, wie dieser aus der Datenbank zu lesen und dort zu speichern ist.



Der Programmierer arbeitet mit *DBObjects* wie mit jedem anderen Java-Objekt und „merkt“ im Prinzip gar nicht, dass er mit der Datenbank kommuniziert. Dadurch ist er von SQL und anderen Datenbank-Spezifika entbunden.

Die *DBObject* Klassen werden mit Hilfe des *DBObject*-Compilers *direkt aus einer existierenden Datenbank* erzeugt - eine Sache von Sekunden. Sollten Tabellenstrukturen in der Datenbank verändert werden, genügt ein Rekompilieren. Aus solchen Datenbankänderungen resultierende Fehler im Anwendungsprogramm werden vom Java-Compiler erkannt und treten nicht etwa zur Laufzeit auf, wie es bei SQL-basierenden Programmen der Fall wäre. Die Möglichkeiten der *DBObjects* sind äußerst vielfältig. So lassen sie sich z. B. bausteinartig zu komplexen persistenten Objekten zusammenbauen.

Vielfache Datenbank Verbindungen

Ein *Esprit*-Server kann beliebig viele Datenbankverbindungen öffnen, sogar gleichzeitig zu Datenbanken von verschiedenen Herstellern. Mit Hilfe der *DBObject*-Persistenzschicht können Daten

hochperformant direkt zwischen den unterschiedlichsten Datenbanken übertragen werden – ggf. auch mit dazwischengeschalteter Logik.

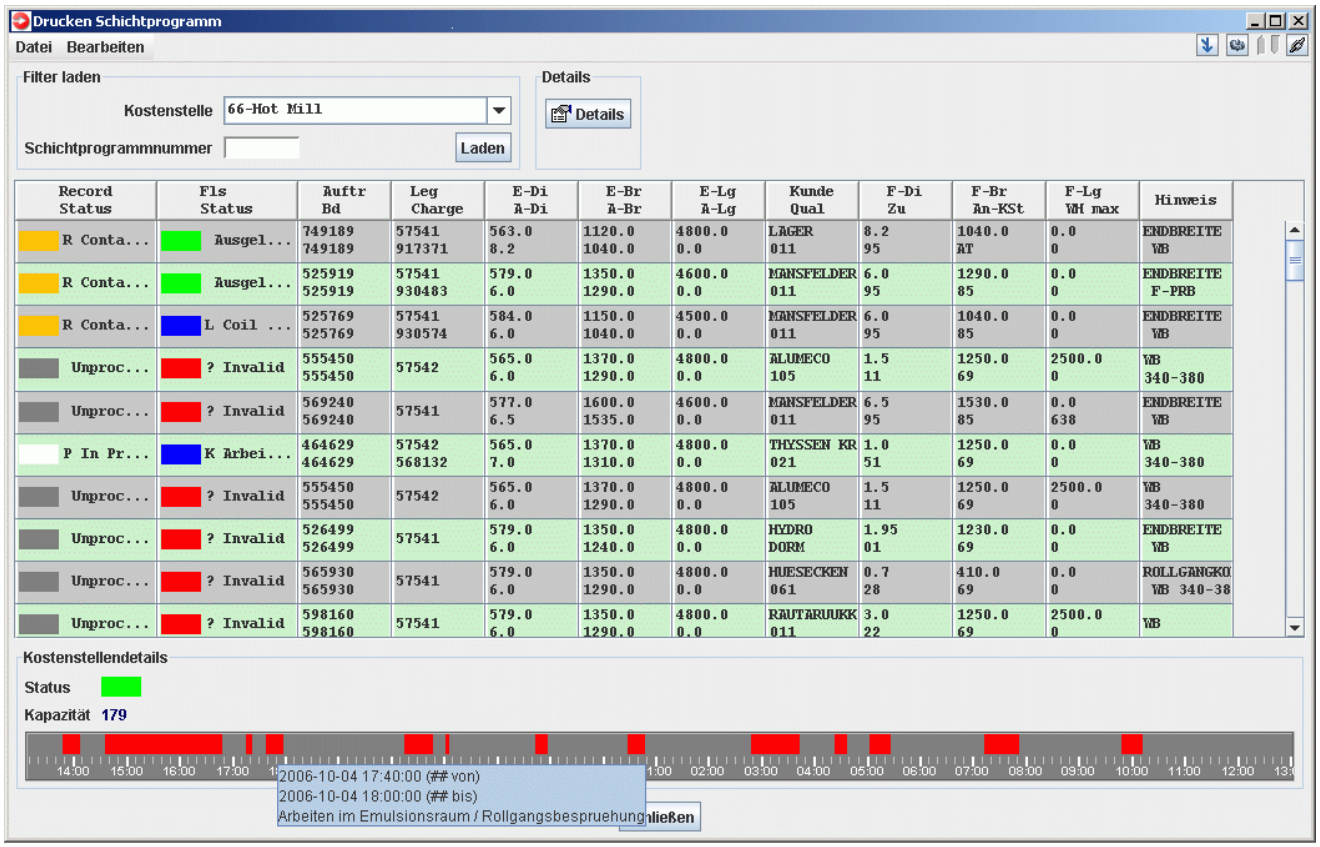
Automatische Client-Installation mit Esprit-AppStore

Für die Installation der Client Software – früher eines der größten Probleme – entsteht heute keinerlei Aufwand mehr, da sie automatisch über Java-Webstart durchgeführt wird. Das Esprit-System erweitert *Java-Webstart* um einen sog. *Esprit-AppStore*, mit dessen Hilfe beliebige Software über Netz installiert werden kann (auch native Software!). Dies gilt sogar für die Server-Software selbst. Idealerweise deponiert man eine neue Software Version auf einem zentralen Webserver. Von dort wird sie jeweils automatisch aktualisiert, wenn ein Client oder ein Server neu gestartet wird.

Anwendungen

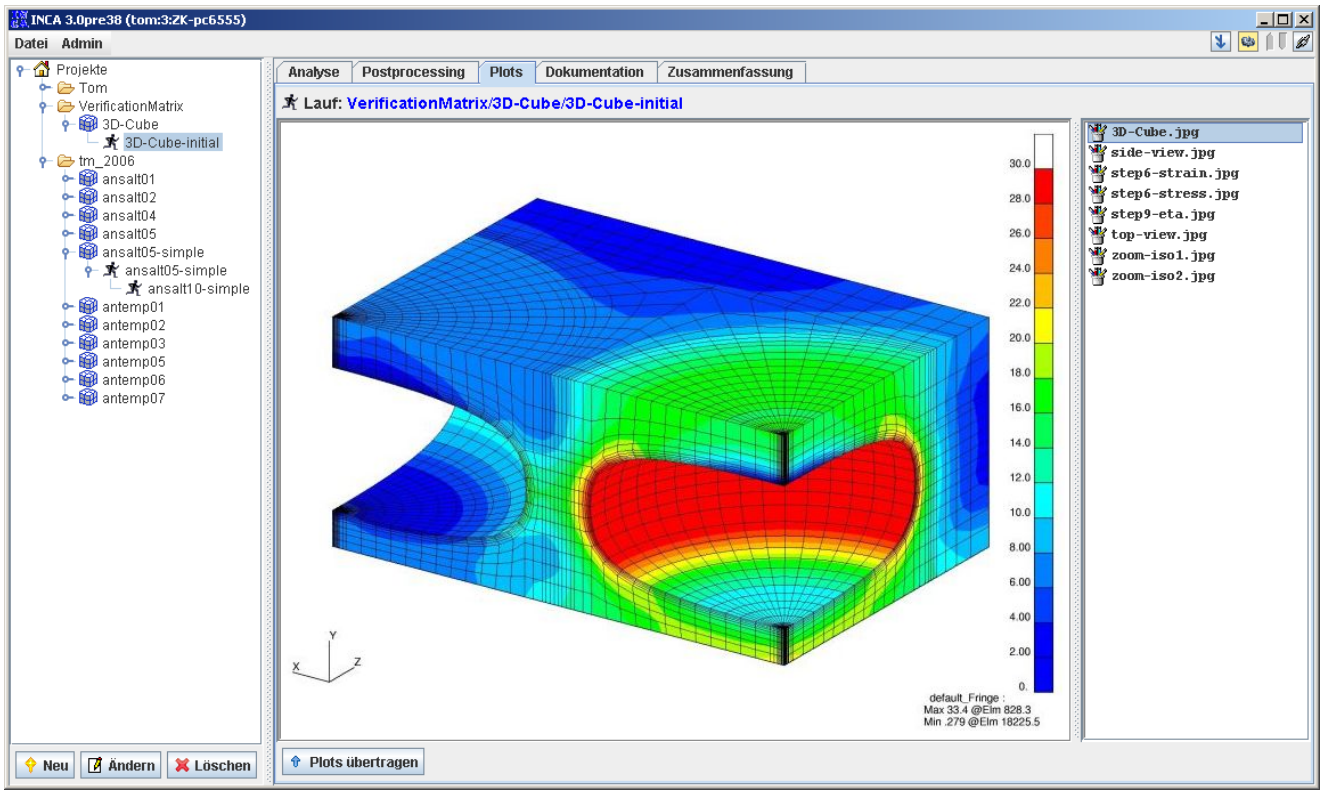
Hydro-AG

Die *Hydro Aluminiumwerke* (www.hydro.com) in Hamburg verwenden die *Esprit-Technologie* für das online Monitoring der Fertigungsabläufe in ihrem Aluminium-Walzwerk. Jede Tabellenzeile repräsentiert einen Fertigungsschritt, der als *Alive Business Objekt (ABO)* modelliert ist und daher automatisch stets aktuelle Werte anzeigt. Ebenfalls ein *ABO* ist die „wandernde“ Zeitachse im unteren Teil, die die aktuellen Zustände (z.B. Stillstandszeiten) der Fertigungsmaschinen anzeigt.



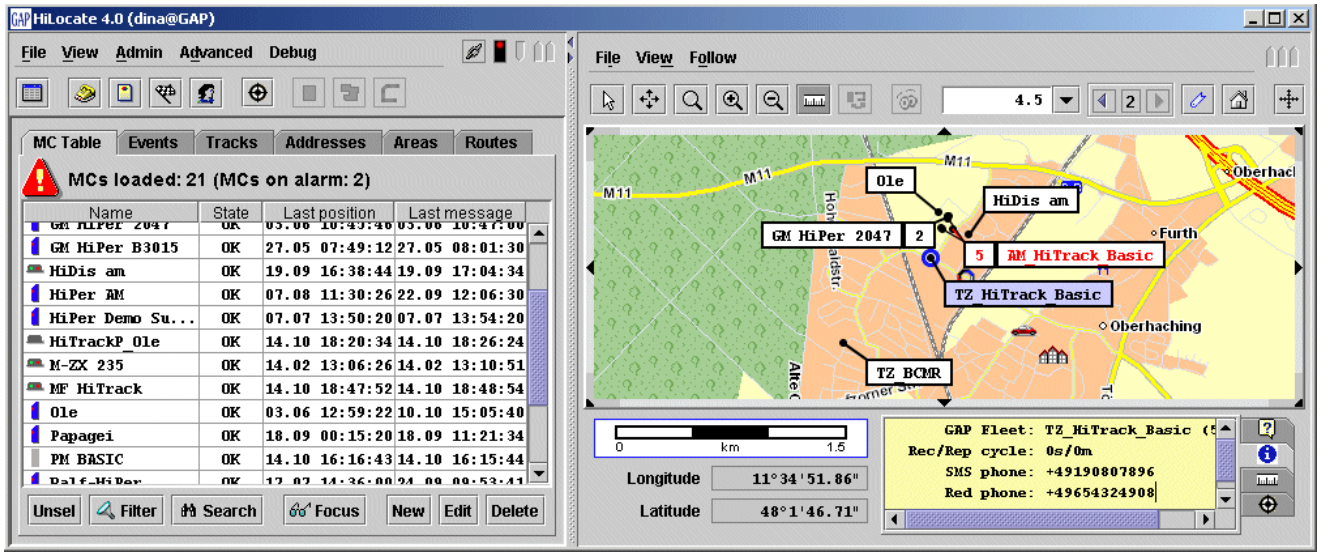
BGR (Bundesanstalt für Geowissenschaften und Rohstoffe)

Die *Bundesanstalt für Geowissenschaften und Rohstoffe* (*BGR*, www.bgr.bund.de) in Hannover benutzt die *Esprit-Technologie* zur Durchführung/Verwaltung von komplexen Finite-Elemente-Berechnungen in geologisch-wissenschaftlichen Anwendungen (Bergwerke, Erdverschiebungen, Erdspannungen, thermische Verläufe etc.). Mit Hilfe von *Workflows werden verteilte Rechen-vorgänge auf verschiedenen Servern gesteuert und kontrolliert - mit Event-Benachrichtigung über mehrere gekoppelte Esprit-Server hinweg.* Darüber hinaus werden große Datenmengen (Berechnungseingaben und -Ergebnisse) über transaktionale File-Transfers zwischen verschiedenen Rechnern transferiert (über den Transfer-Port des *Esprit*-Servers).



ISA-Telematics

Die Telematik-Applikation *HiLocate* (www.isatelematics.de) erlaubt die online Verfolgung von Fahrzeugen auf einer elektronischen Landkarte. Die Fahrzeuge senden periodisch ihre aktuelle GPS-Position an einen zentralen Server, der seinerseits die Clients benachrichtigt. Diese Applikation nutzt das durch die *Esprit*-Technologie ermöglichte Parallelprozessing auf Client und Server: Während der Server den aktuellen Kartenausschnitt generiert, berechnet der Client die Platzierung der Markierungsfähnchen zur Anzeige der Fahrzeugpositionen.



Merkmale auf einen Blick

Esprit Core Features

- Echte Client-Sessions basierend auf einer stehenden Doppelkanal-Verbindung
- Unterstützung für synchrone und asynchrone Requests
- Asynchrones Messaging per Instant Message Service
- EMI (**E**sprit **M**ethod **I**nvocation) als Netzwerk-Programmierschnittstelle
- Model-View-Controller Prinzip über Netz vollständig realisiert
- Remote Properties garantieren den Clients stets gleiche Sicht der Welt
- Unterstützung für lokale und remote Workflows
  (Prozess-Steuerung und Monitoring über mehrere Rechner hinweg)
- Hochperformante Datei-Persistenz im NDF-Format
- Hochperformante Datenbank-Persistenz mit DBObjects
- Support für transaktionale, bidirektionale File-Transfers sowie Stream-Downloads
- Unterstützung für Server Ressource-Sperren (ermöglicht geschütztes Editieren)
- Viele Server lassen sich zu einem Verbundnetz zusammenschalten
- Zeitgesteuerte Server-Hintergrundprozesse (crontab ähnlich)
- Eingebaute Benutzer- und Gruppenverwaltung, wahlweise in Datenbank oder Dateien
- Clients sind vom Server steuerbar (per Nachricht vom Typ Down-Request)
- Clients können untereinander kommunizieren (Chat Tool inklusive)
- Online Monitoring der Serveraktivität sowie Auswertung der Serverstatistik
- Umfangreiche Möglichkeiten des Server-Loggings (auch Client-spezifisches Logging)
- Online und Offline Arbeitsmodi (bei Offline keine Asynchronität und keine Event-Benachrichtigung)
- Gleichzeitige Verbindung zu Datenbanken mehrerer Hersteller und Möglichkeit der direkten Datenübertragung
- Sehr komfortable Unterstützung für Mehrsprachigkeit
- Esprit-AppStore Applikationen sind Webstart-fähig. Damit entfällt jeglicher Installationsaufwand

Ihre besonderen Vorteile

- Für Systeme mit einigen 100 bis 1000 Benutzern stellt das *Esprit*-System eine preiswerte Alternative zu hochkomplexen und teuren Applikations-Servern dar
- *Esprit*-Applikationen sind häufig auch die bessere Alternative zu Web-Applikationen
- Sie bauen auf ein leistungsfähiges Client/Server Entwicklungsframework, in dem Jahrzehnte an Erfahrung stecken und deren Qualität durch umfangreiche automatische Tests gewährleistet ist.

- *Esprit* bietet Ihnen ein Konzept, remote Prozesse zu steuern und zu kontrollieren
- *Esprit* bietet Ihnen ein Konzept zur Einbindung und Vereinheitlichung aller Ihrer Tools
- *Esprit* ist pure Java (stets die jeweils aktuellste Version) und daher auf jeder Plattform lauffähig
- *Esprit*-Applikationen sind hochperformant, da typischerweise nur Delta-Informationen über Netz ausgetauscht werden
- *Esprit*-Applikationen haben eine wesentlich ausgewogenere Client/Server Lastverteilung als herkömmliche Systeme, die typischerweise sehr serverbelastend sind
- Altanwendungen lassen sich relativ leicht in moderne *Esprit*-Anwendungen umschreiben. Dabei gewinnen Sie die Client/Server Fähigkeit gleich mit

Vorteile für Entwickler

- Sie entwickeln nur noch das, was Sie **wirklich** brauchen, das Netzwerk bleibt für Sie „unsichtbar“
- Sie können Client/Server Systeme mit höchster Komplexität entwickeln, die trotzdem äußerst zuverlässig funktionieren
- Der Quellcode Ihres Projekts bleibt sehr kompakt, übersichtlich, gut test- und wartbar
- *Esprit*-Applikationen sind äußerst robust. Sie verlieren kaum Zeit mit Fehlern und Debugging
- Die *Esprit*-Technologie ist leicht erlernbar. Neben Standard-Java ist kein spezielles Know-How erforderlich
- *Esprit*-Applikationen sind äußerst schlank und Memory-effizient
- Der Besitz der Quellcodes gibt Ihnen Sicherheit und Unabhängigkeit. Und sie sehen, wie’s funktioniert
- Für Ihre Entwicklungsumgebung werden keine zusätzlichen Fremdmodule benötigt

Esprit-Systems Know-How zu Ihrem Nutzen

Durch unsere langjährige Entwicklungstätigkeit in vielen Projekten besitzen wir ein umfangreiches Know-How im Bereich Java und Client/Server Computing, das wir Ihnen gerne zur Verfügung stellen. Wir analysieren Ihre DV-Infrastruktur sowie Ihre Datenflüsse und implementieren auf der Grundlage der *Esprit*-Technologie die für Sie beste Client/Server Lösung. *Esprit-Systems* hat es sich zum Ziel gesetzt, mit Ihnen gemeinsam Esprit-Anwendungen zu realisieren und diese Technologie als Standard für Rich-Clients zu etablieren.

Für den Fall, dass Sie eigene Entwicklungsressourcen zur Verfügung haben, hat sich bisher folgendes Konzept am besten bewährt: Sie erhalten von uns die Quellcodes unseres Entwicklungsframeworks und entwickeln auf dieser Basis Ihre Lösung selbst – natürlich mit unserer intensiven Unterstützung. So ist für Sie größtmögliche Transparenz und Unabhängigkeit gewährleistet. Wir schulen Ihre Entwicklungsmannschaft in professioneller Java-Programmierung und begleiten ihre Projekte mit Coachingmaßnahmen bis zum sicheren Erfolg.

Weitere Informationen finden Sie unter: www.esprit-systems.de