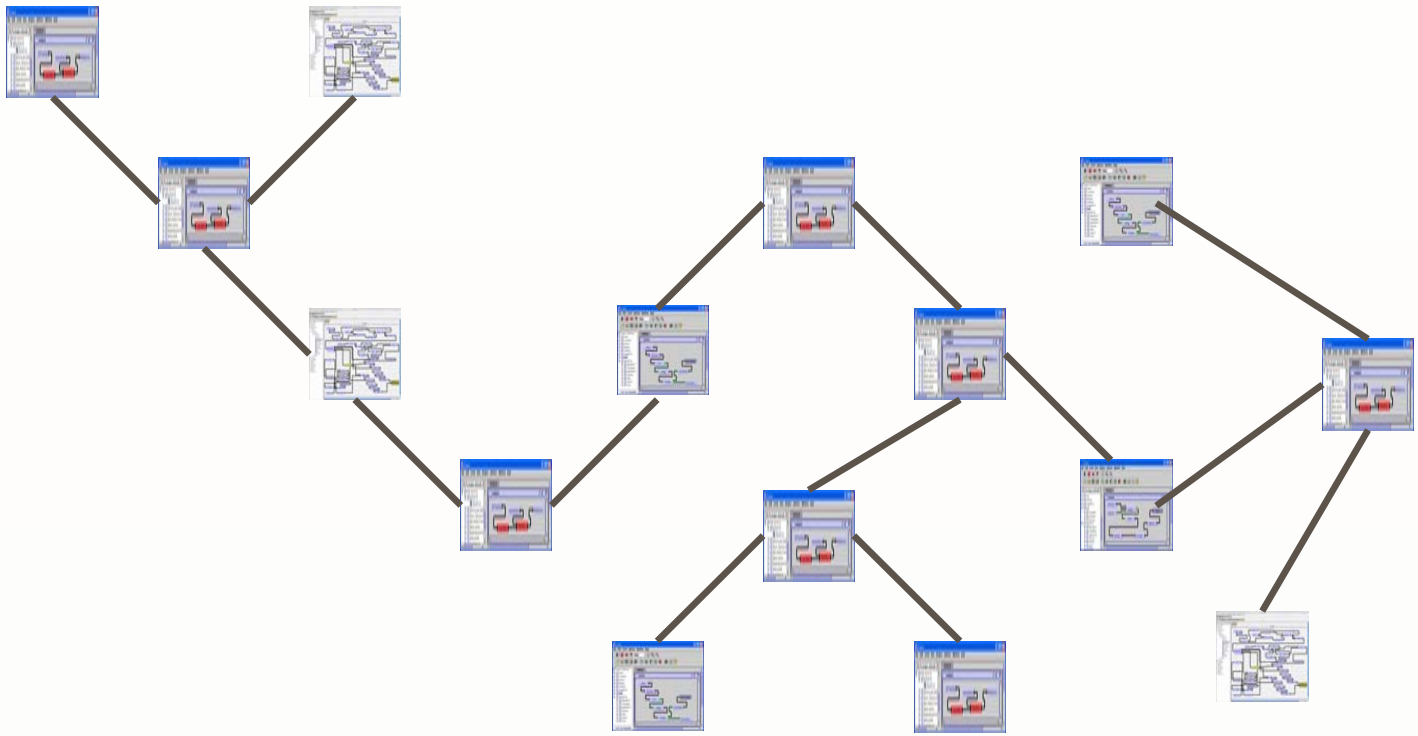
Programming Workflow with Triana Services



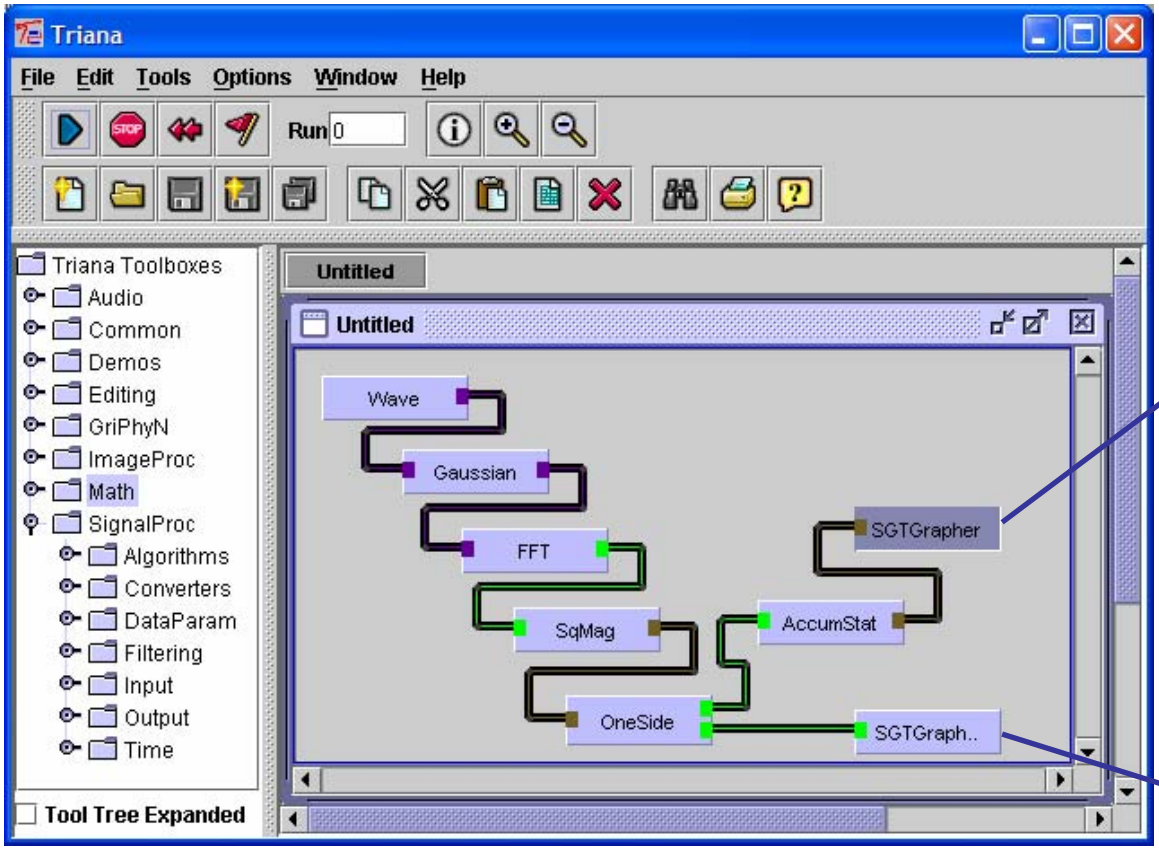
Matthew Shields,

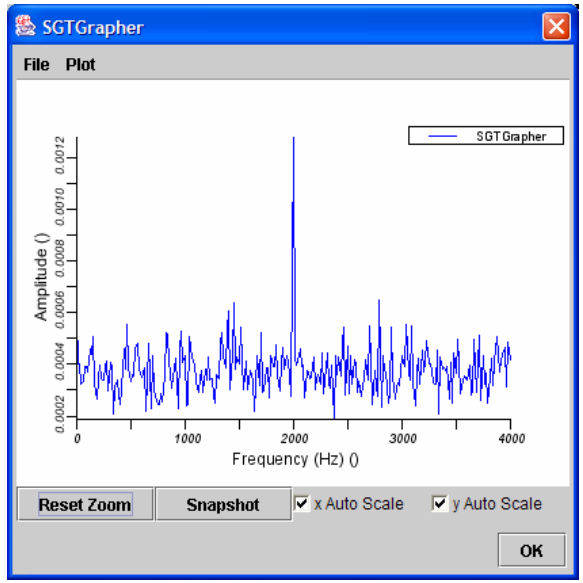
SC4DEVO Workshop, 12-15 July 2004

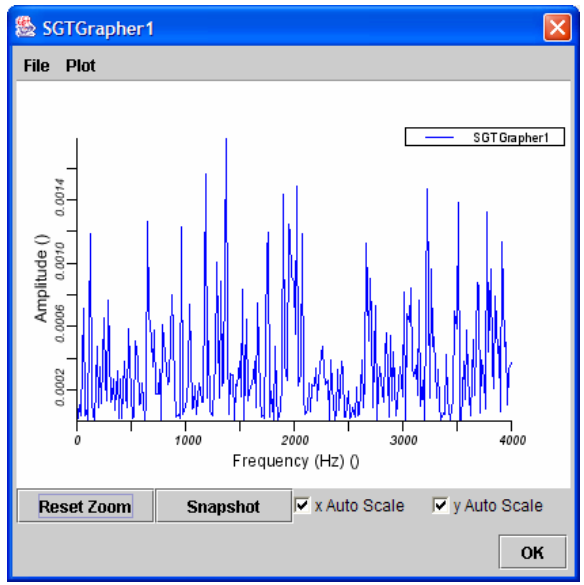
What is Triana?

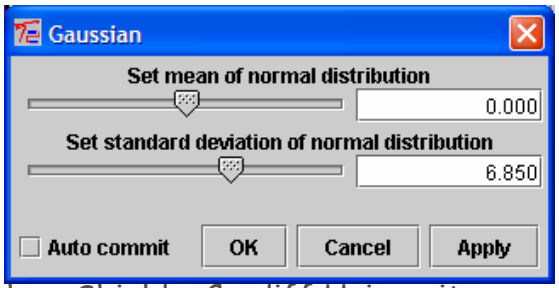
- Distributed Problem Solving Environment
  - Composing, Compiling and Running Applications
- Multiple problem domains
  - Signal Processing, Audio, Maths, Image Processing
- Intuitive to use
- Extensible
- Hide distributed computing details
- Middleware agnostic
  - P2PS, Web Services, Grid Computing

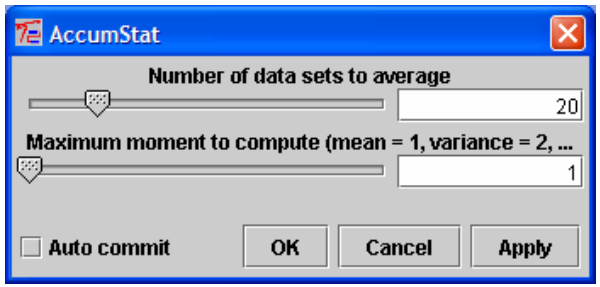
What is Triana?











Triana Workflow

- Triana is flow based
  - Data flow - data arriving at component triggers execution
  - Control flow - control commands trigger execution
- Decentralised execution
  - Data or Control messages sent along communication "pipes" from sender to receiver causes receiver to execute
  - Synchronous or Asynchronous messaging (Implementation dependant)
  - Multiple inputs can block or trigger immediately (Component designer defined)

Components and Definitions

- Component is unit of execution
- Components are defined in XML files:
  - Similar to WSDL
  - Naming information
  - Input and output ports
  - Parameter information
- Why Components?
  - To simplify the application design process and to speed up application development
- The component model provides an infrastructure for the interaction of components

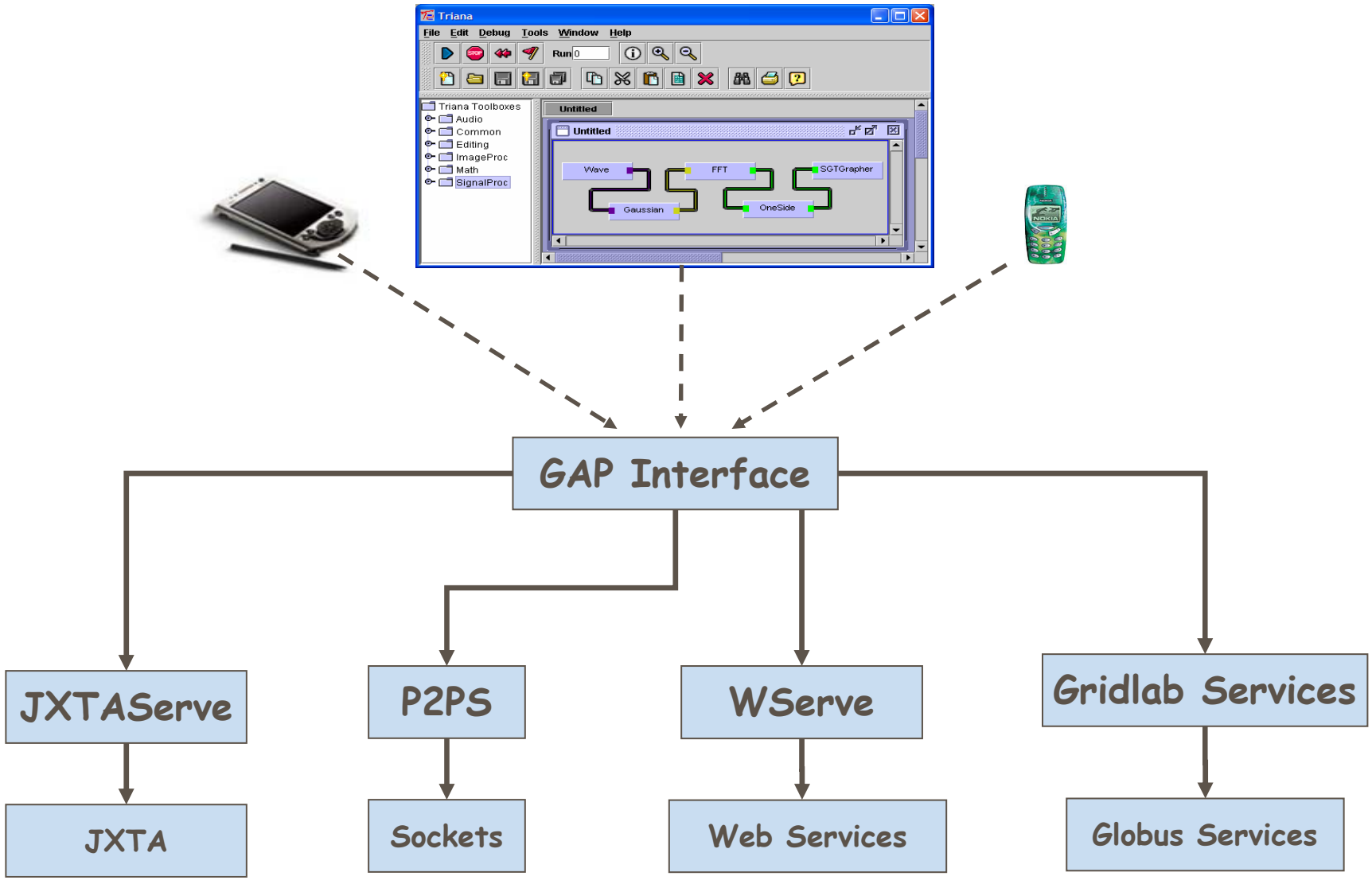
Taskgraph

- Internal object based workflow graph representation
  - Taskgraph - DCG
  - Tasks
  - Connections
- External XML representation
  - Simple XML syntax
  - List of participating task definitions
  - Parent/Child connection
  - Hierarchical (Compound components)
- Alternative Languages & Syntax
  - e.g. BPEL4WS
  - Available through pluggable readers & writers.

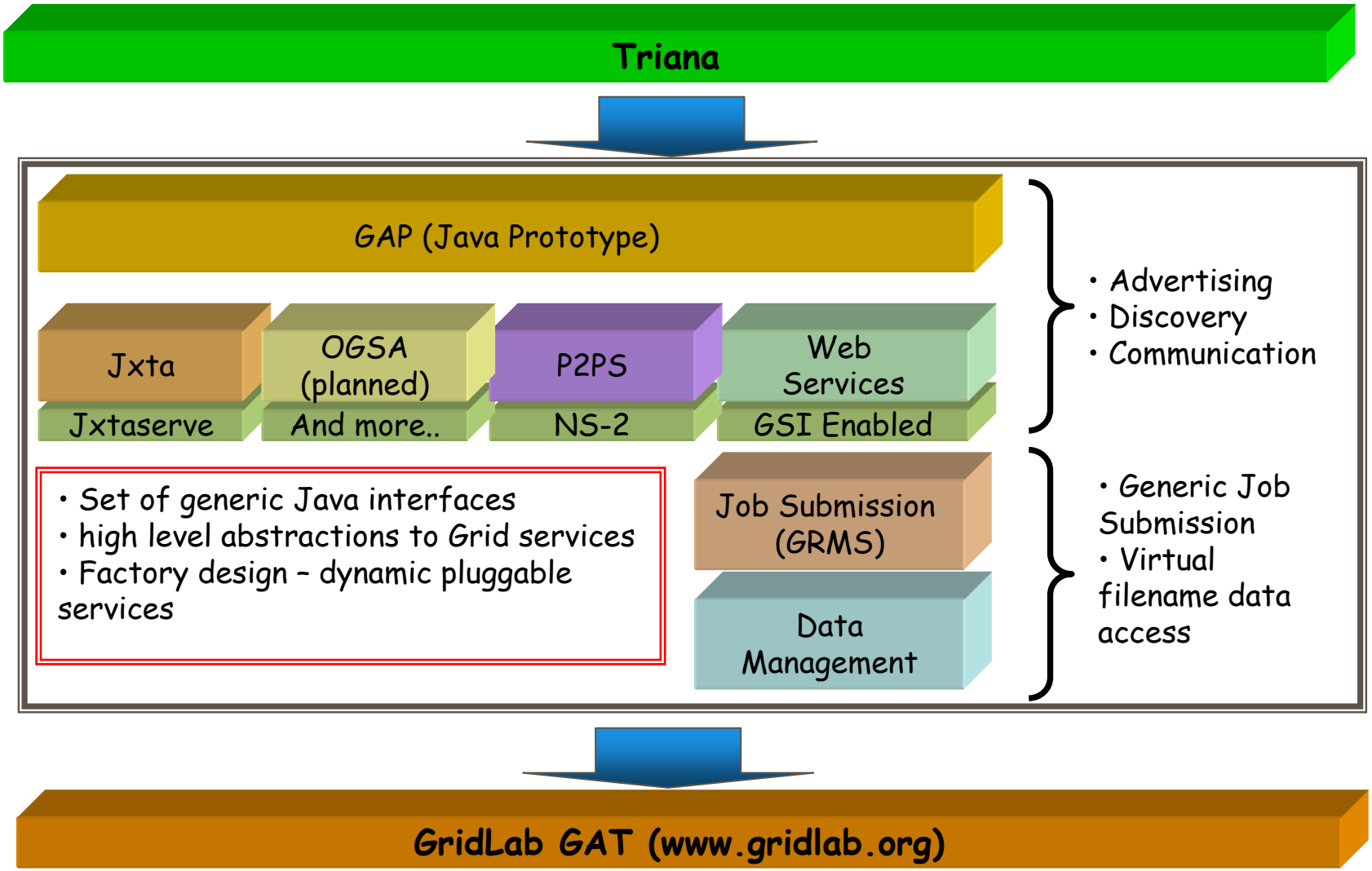
Workflow

- No explicit language support for control constructs
- Loops and execution branching handled by components
  - Loop component - controls loop over sub-workflow
  - Logical component - control workflow branching
- Unlike BPEL4WS or similar
- Flexibility of control - constraint based loops etc…
- Prevents workflow language feature creep

Current Triana Architecture



Java GAT Prototype



GAP Overview

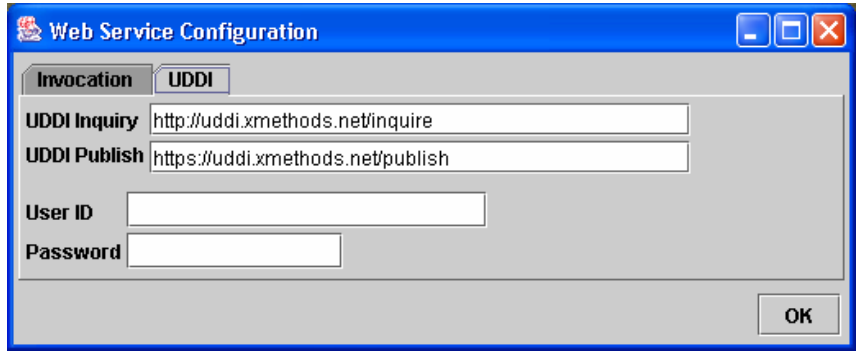
- Everything is a service!
  - Defined interfaces (WSDL)
  - Message based communication (SOAP)
- Java interface classes with concrete implementations that form the GAP bindings
- The core interface includes:
  - Service Creation and Discovery
  - Pipe Creation and Discovery
  - Message Communication
  - Information, Job Submission, Data Management

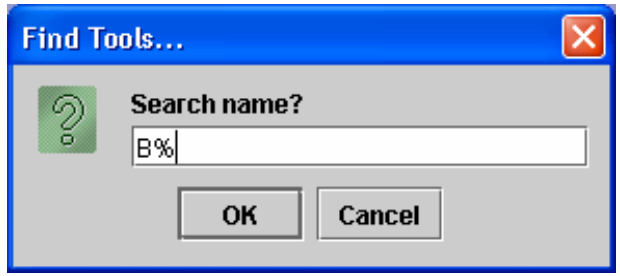
GridLab GAT & SAGA

- Grid Application Toolkit (GAT)
  - Written in C
  - API to shield application developers from implementation details
  - Adapters provide bindings to implementations
  - Triana & Catcus demonstration applications
- GAP is an adapter for the Java GAT (pending), providing:
  - Advertisement, Discovery, deployment and communication of services
  - GRMS job submission adapter
  - Data Management Services
- Simple API Grid Applications (SAGA)
  - GridLab input to this GGF RG

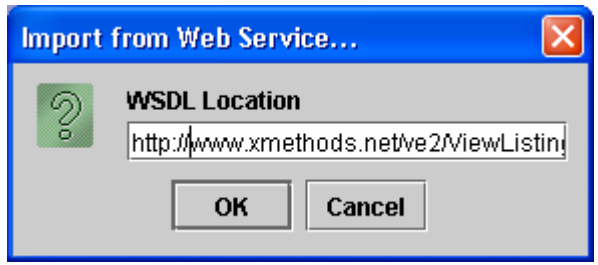
Web Service Discovery 1

- Triana allows users to query UDDI repositories





- Alternatively, users can import services directly from WSDL



Web Service Discovery 2

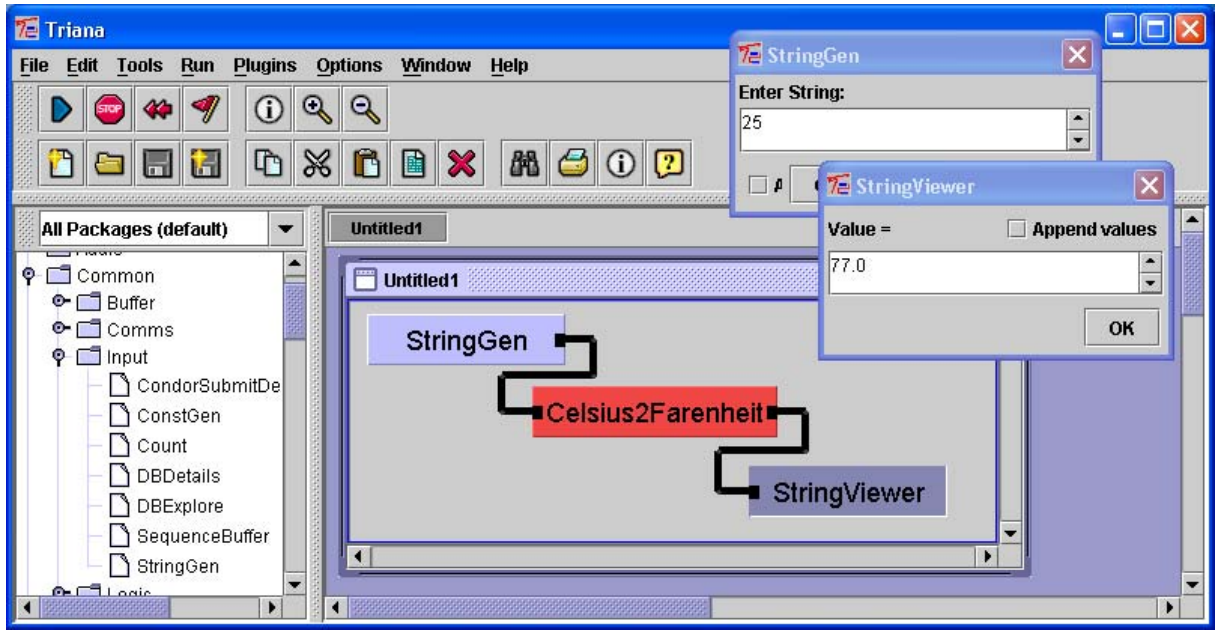
- Discovered/Imported Web Services are converted into Triana tools

  (service name = tool name)
  (input message parts = in nodes)
  (output message parts = out nodes)
  etc…

- Web Service tools are displayed in the user's Tool Tree (alongside local tools)

Connecting Workflows

- Web Service tools can be dropped onto the user's workspace and connected like local tools
- A workflow can contain both local and Web Service tools

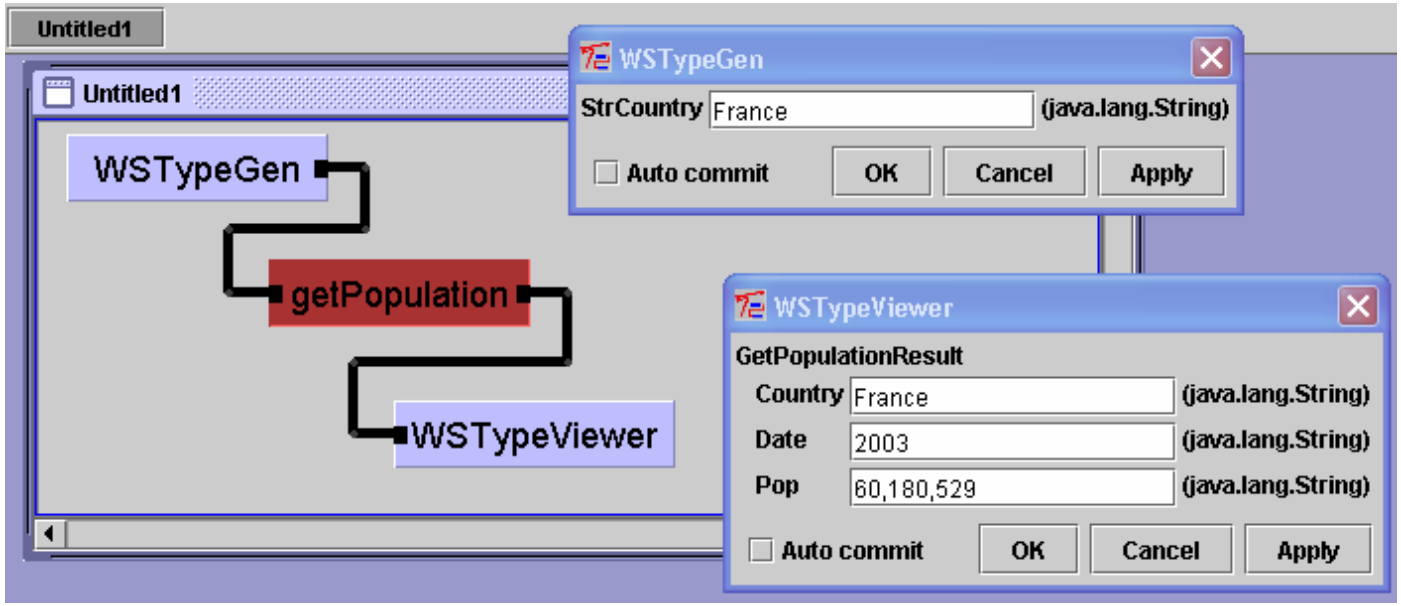


Invoking Web Services

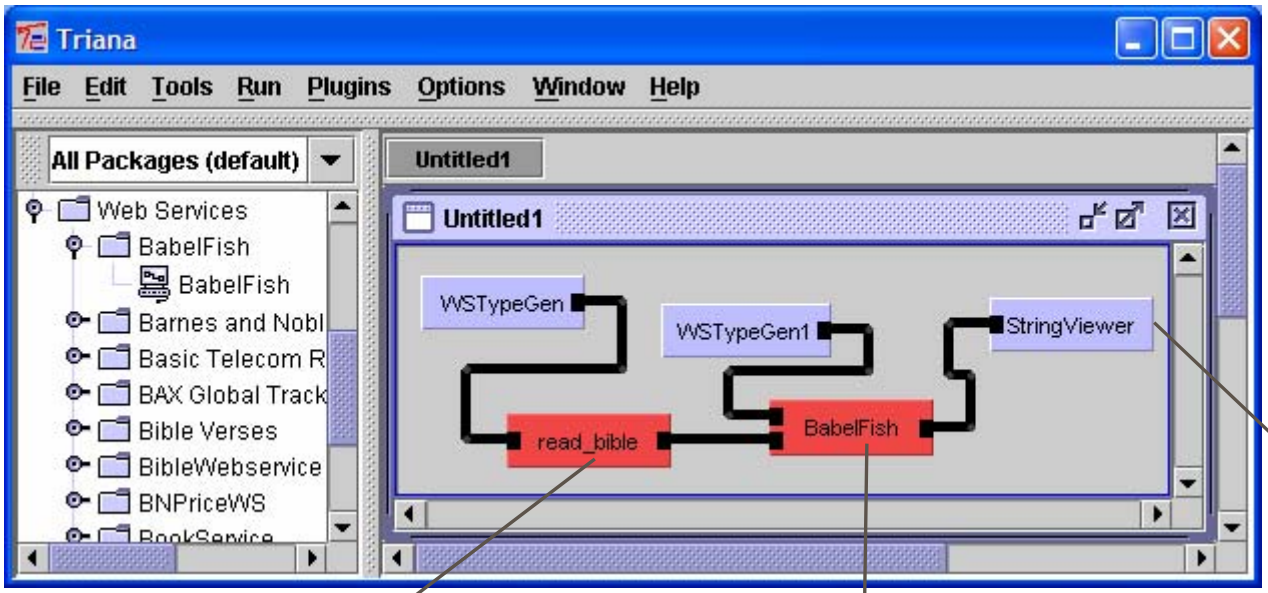
- Web Services are dynamically invoked using Apache AXIS (when input data is received)
- Three stages:
  - A static stub for the web service is generated using WSDL2Java
  - The stub is compiled using javac
  - The stub is dynamically loaded and invoked with the input data
- Generated/compiled stubs are cached
  - Saves regenerating/compiling stubs each invocation

Complex Data Types

- Users can build their own interface for creating/mediating between complex types
- Alternatively, Triana can dynamically generate an interface from the WSDL2Java generated bean class

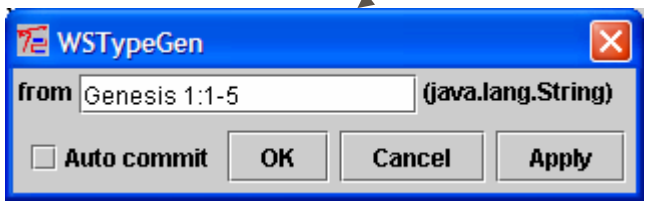


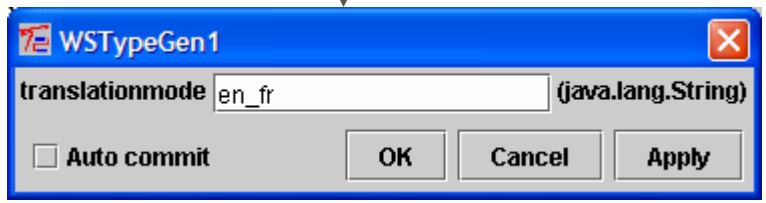
Converting the Bible into French

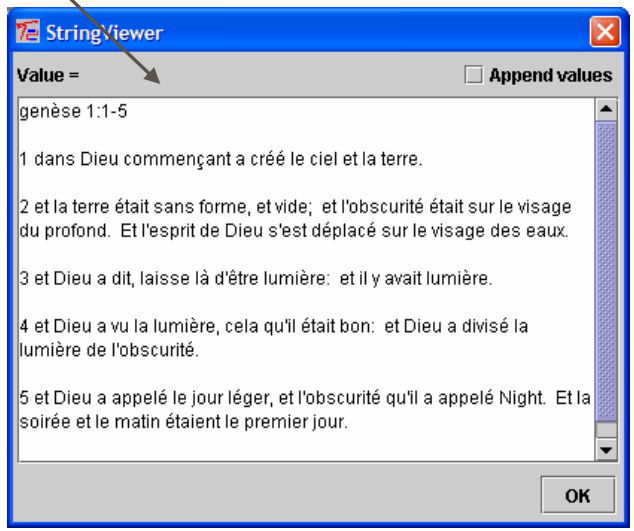


Simple example:

- read_bible - extracts verses from the bible
- BabelFish - translates between English and French (and other languages)
- Result = The Bible translated into French !



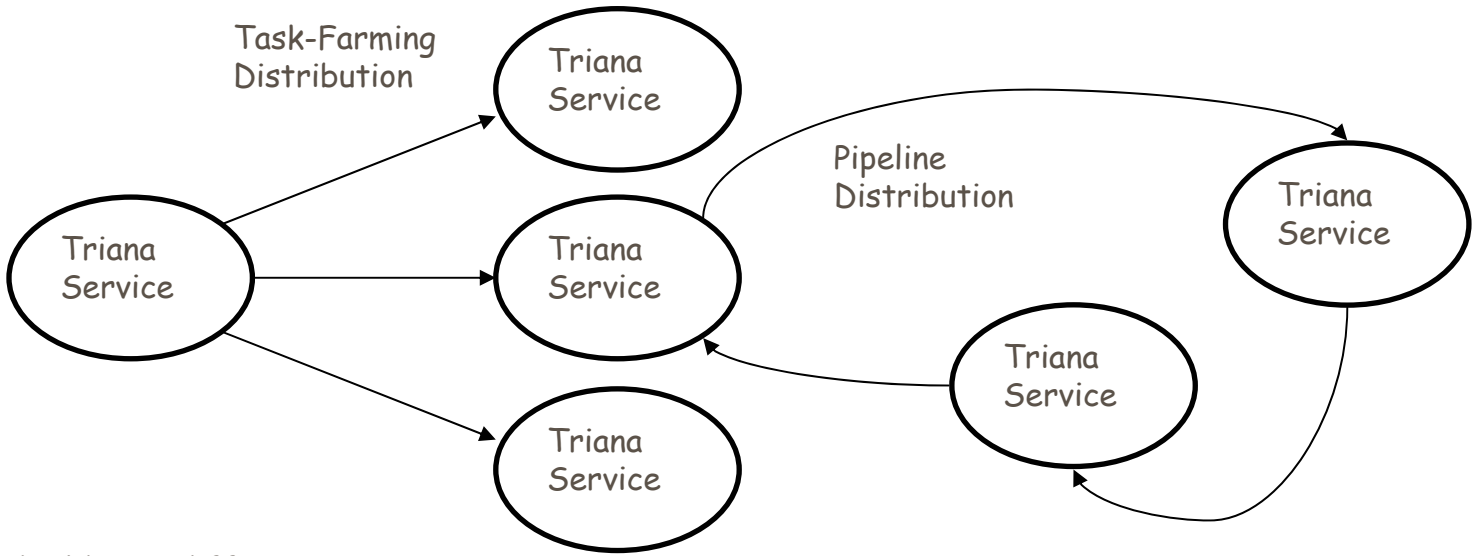




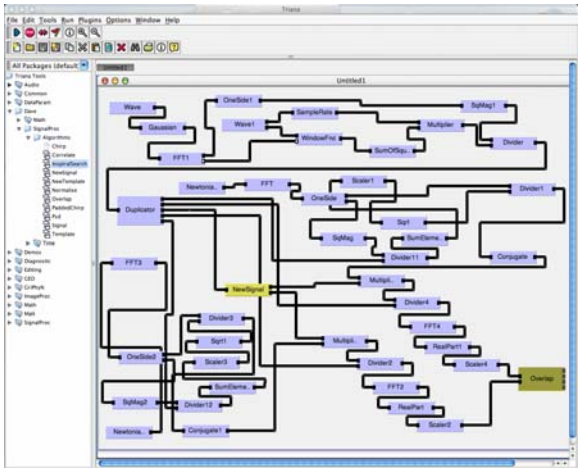
Red Boxes – Web Services

Distributed Workflow

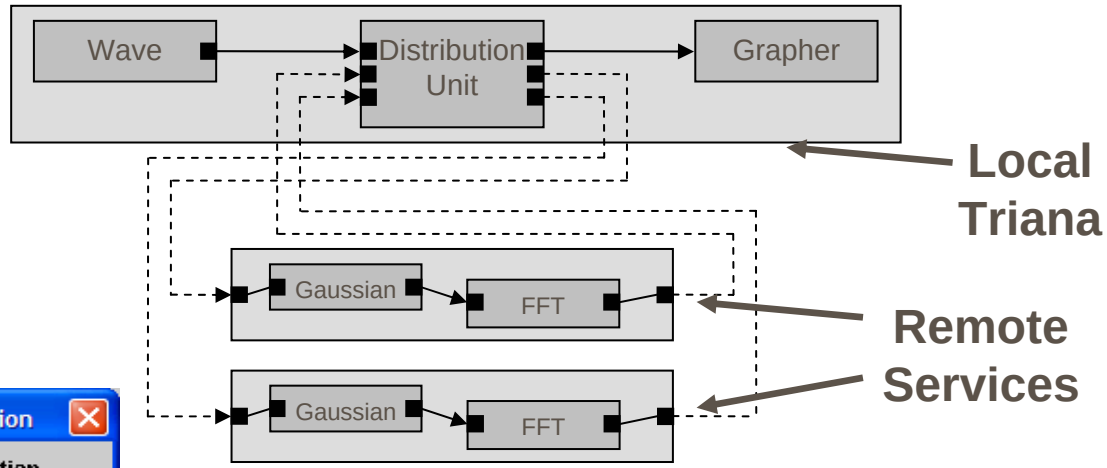
- Distributed Triana Workflows
  - Based around Triana **Groups** i.e. aggregate tools
  - Each group can be distributed
  - Distribution policies:
    - **HTC** - high throughput/task farming
    - **Pipeline** - allow node to node communication
  - Each service can be a gateway to finer granularities of distribution:



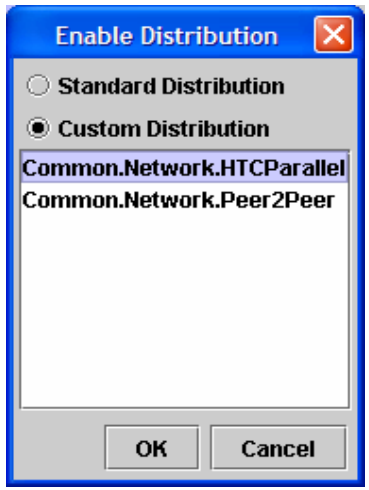
Dynamic Distributed Workflow



- The workflow is cloned/split/rewired to achieve the required distribution topology



Custom distribution scripts allow sub-workflows to be distributed in parallel or pipelined



- Distribution scripts are standard Triana workflows, enabling users to create their own custom distributions

Distributing Triana Workflow

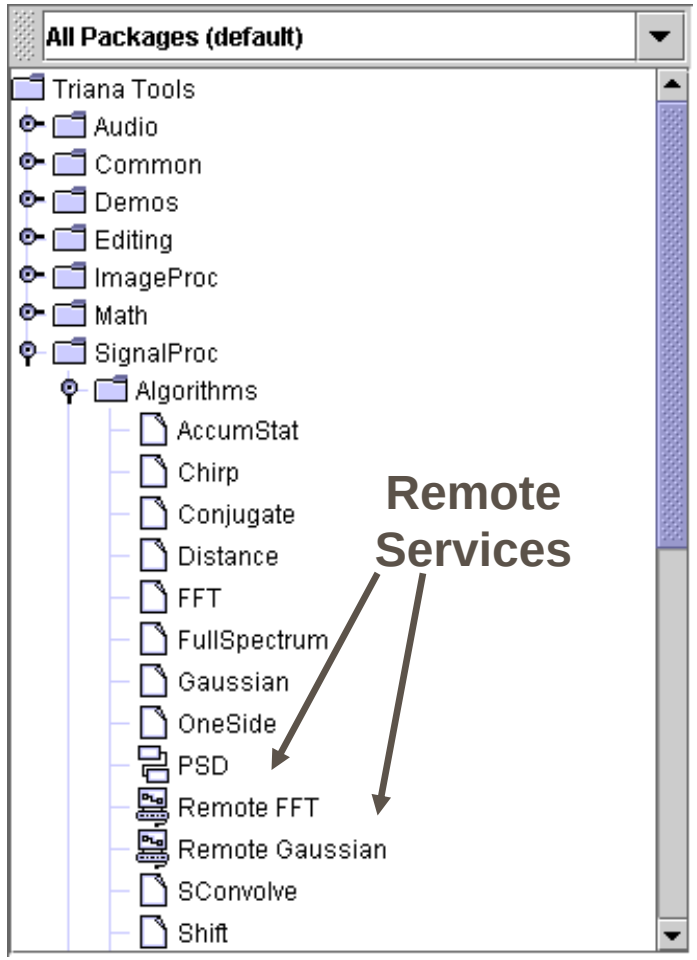
- Deploy Remote Triana Services on Resources
  - Service application installation
  - Service execution
  - Service discovery
- Mapping workflow to Triana Services
  - Workflow rewiring, XML definition for connections modified for remote location - sub-workflows duplicated
  - Data distribution, annotated sub-sections of taskgraph passed to resources

Deploying Services

- WSPeer - Axis based WS framework
  - Standards based - WSDL & SOAP
  - Automatically wrap Triana workflow as WS
  - Advertise & discovery using UDDI
  - Service communication with Axis
- P2PS - socket based Peer-2-Peer framework
  - Advertisement, discovery & communication in ad-hoc P2P networks
  - Advertise & discovery using subnet multicast & rendezvous peers
  - Service communication through socket based pipe

Deploying and Connecting To Remote Services

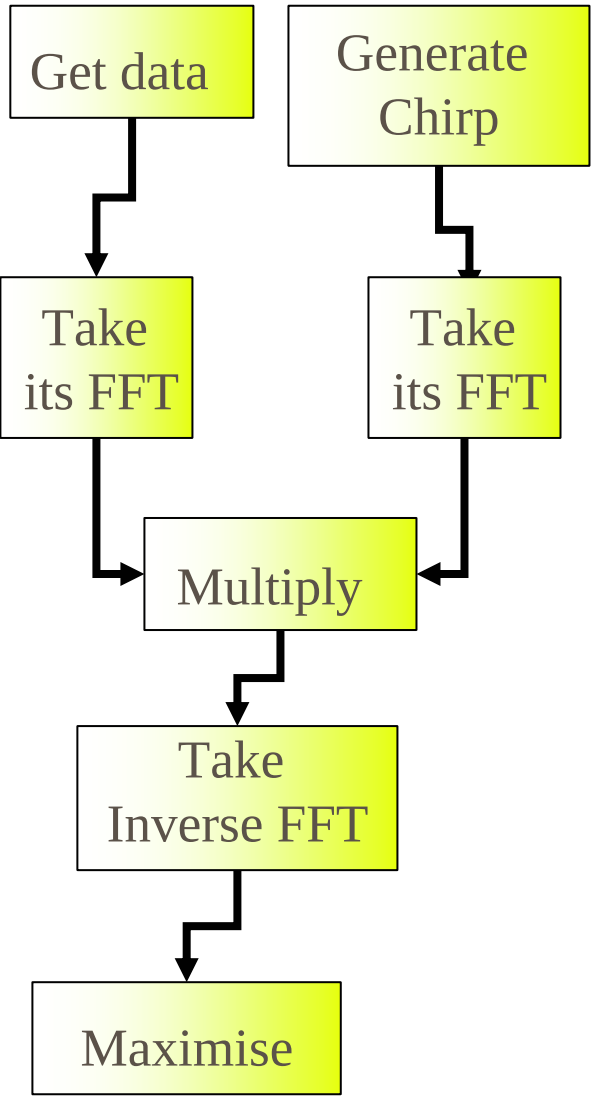
- Running services are automatically discovered via the GAP Interface, and appear in the tool tree
- User can drag remote services onto the workspace and connect cables to them like standard tools (except the cables represent actual JXTA/P2PS pipes)

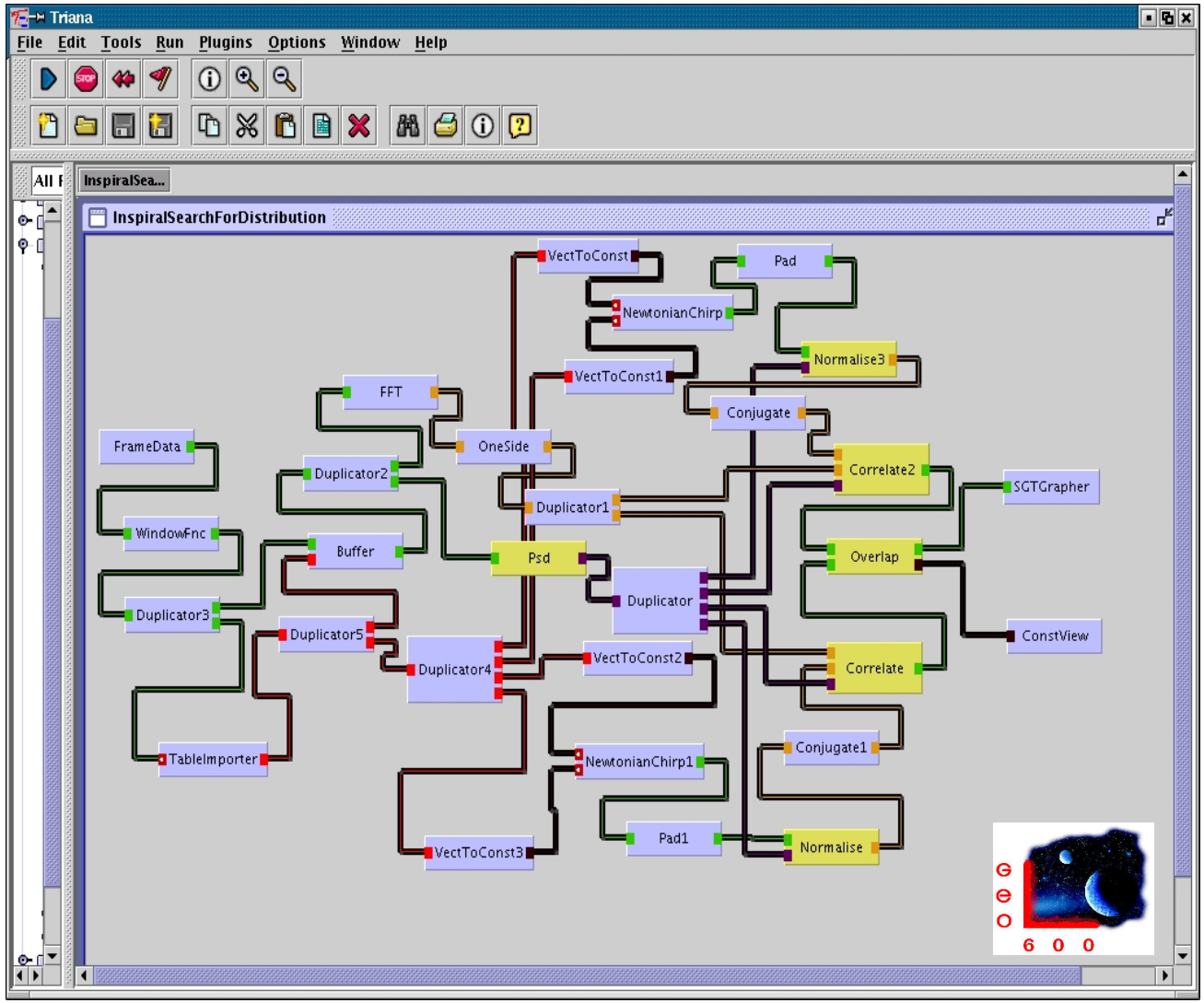


GEO 600 Matched Filtering

- Background
  - Simplified inspiralling binary search algorithm
  - Compact binary stars orbiting each other in a close orbit
  - As the orbital radius decreases a characteristic chirp waveform is produced - amplitude and frequency increase with time until eventually the two bodies merge together
- Computing
  - Need 10 Gigaflops to keep up with real time data (modest search..)
    - Data 8kHz in 24-bit resolution (stored in 3 bytes) -> Signal contained within 1 kHz = 2000 samples/second
    - divided into chunks of 15 minutes in duration (i.e. 900 seconds) = 8MB
- Algorithm
  - Data is transmitted to a node
  - Node initialises i.e. generates its templates (around 10000)
  - fast correlates its templates with data

Fast Correlation in the Frequency Domain



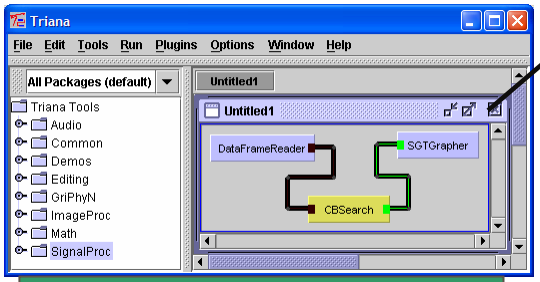


Matthew Shields, Cardiff University

Distributed Matched Filtering

Controller

Email, SMS notification



GAT (GRMS, Adaptive)

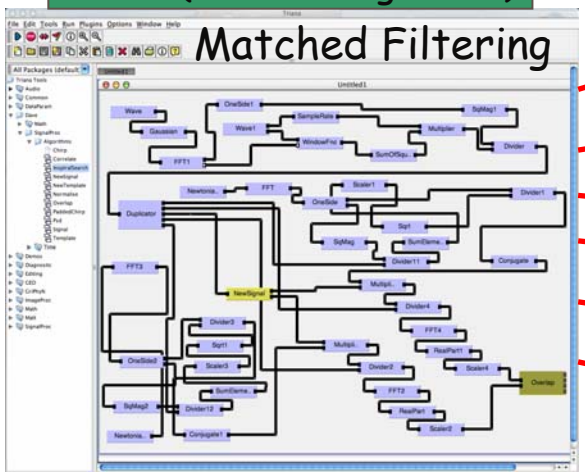
- Submit Job
- Optimised Mapping

Logical File Name

GW Data Distributed Storage

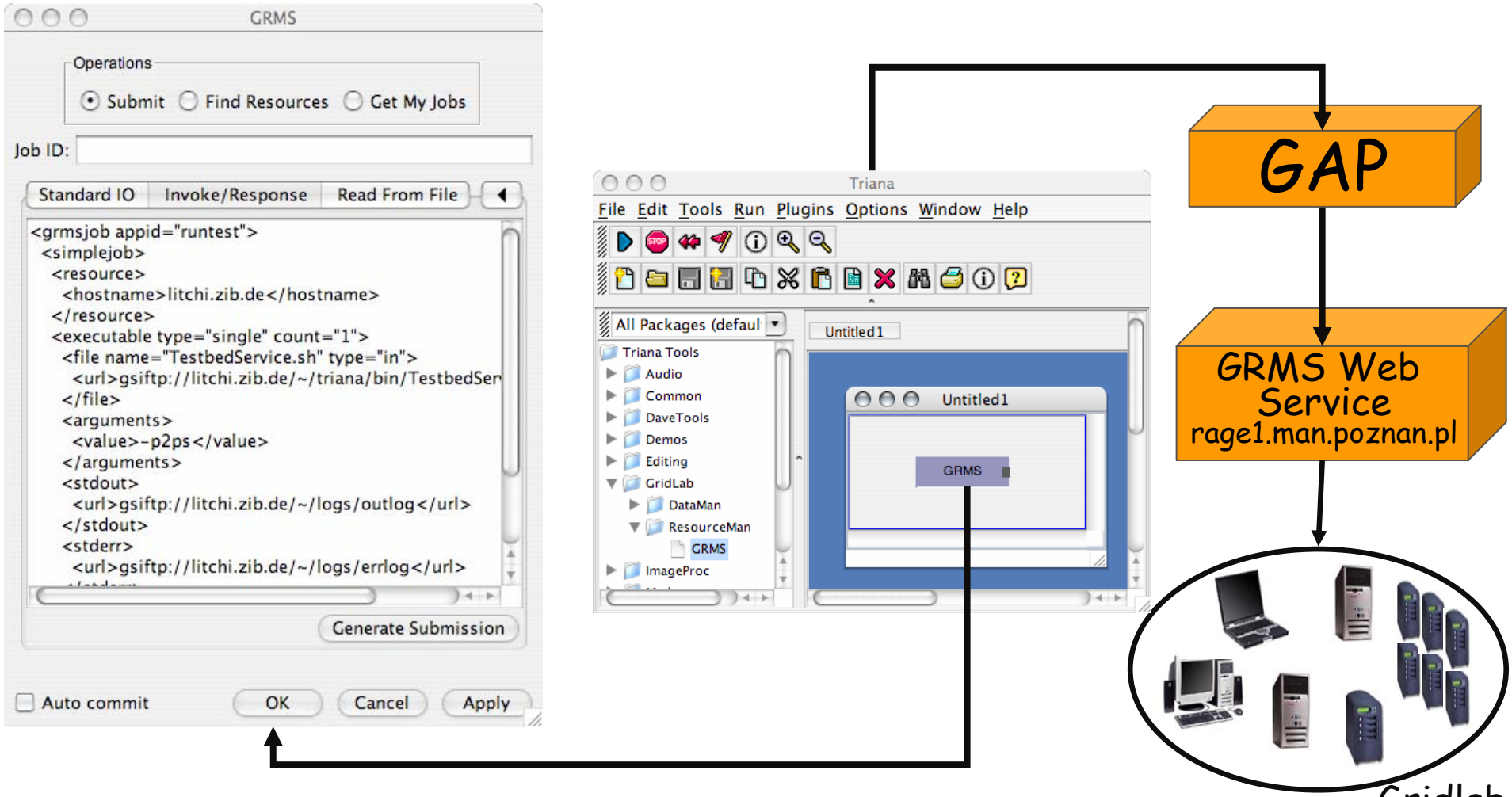
GW Data

GAT (Data Management)



Gridlab Test-bed

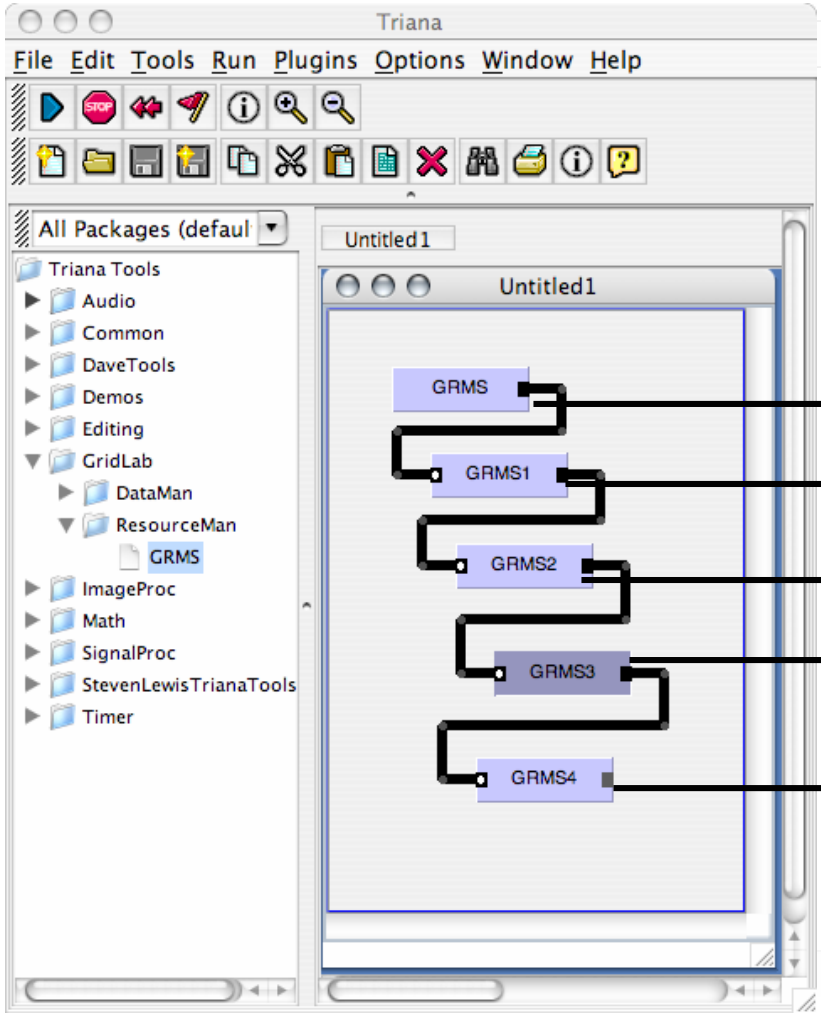
Triana Service Job Submission



Triana GRMS Component

- Front end to GridLab GRMS Web Service
  - Job Submission Service - interfaces with GRAM
- GAP Web Service binding + GSI Authentication
- Java CoG Kit
  - X509 Certificate handling
  - Axis authentication & communication
- GRMS executes applications on GridLab Testbed
  - Heterogeneous hardware platforms
  - Default software - Globus 2.4, GSISSH, cc, cvs, c++, F90, make, perl, mpicc

Service Composition Workflow



GAP (using GRMS service)

- Multiple GRMS Components
  - Install Applications (ftp, tar, ant)
  - Start installed Triana Services

Applications/Collaborators

- GEO++ (GEO 600)
  - GW detector characterization
  - Veto studies
- GEO++ Monitors process the raw data for glitches, coherences, narrowband line sources, fluctuations in power in several frequency bands and record the results in appropriate database tables.
- Developed in C++ - fast, stable, handle large amounts of data.
- Triana imports proxies for GEO++ monitors
  - Workflow of proxies executed by component
- Triana data mining units access database
- Visualisation units for results

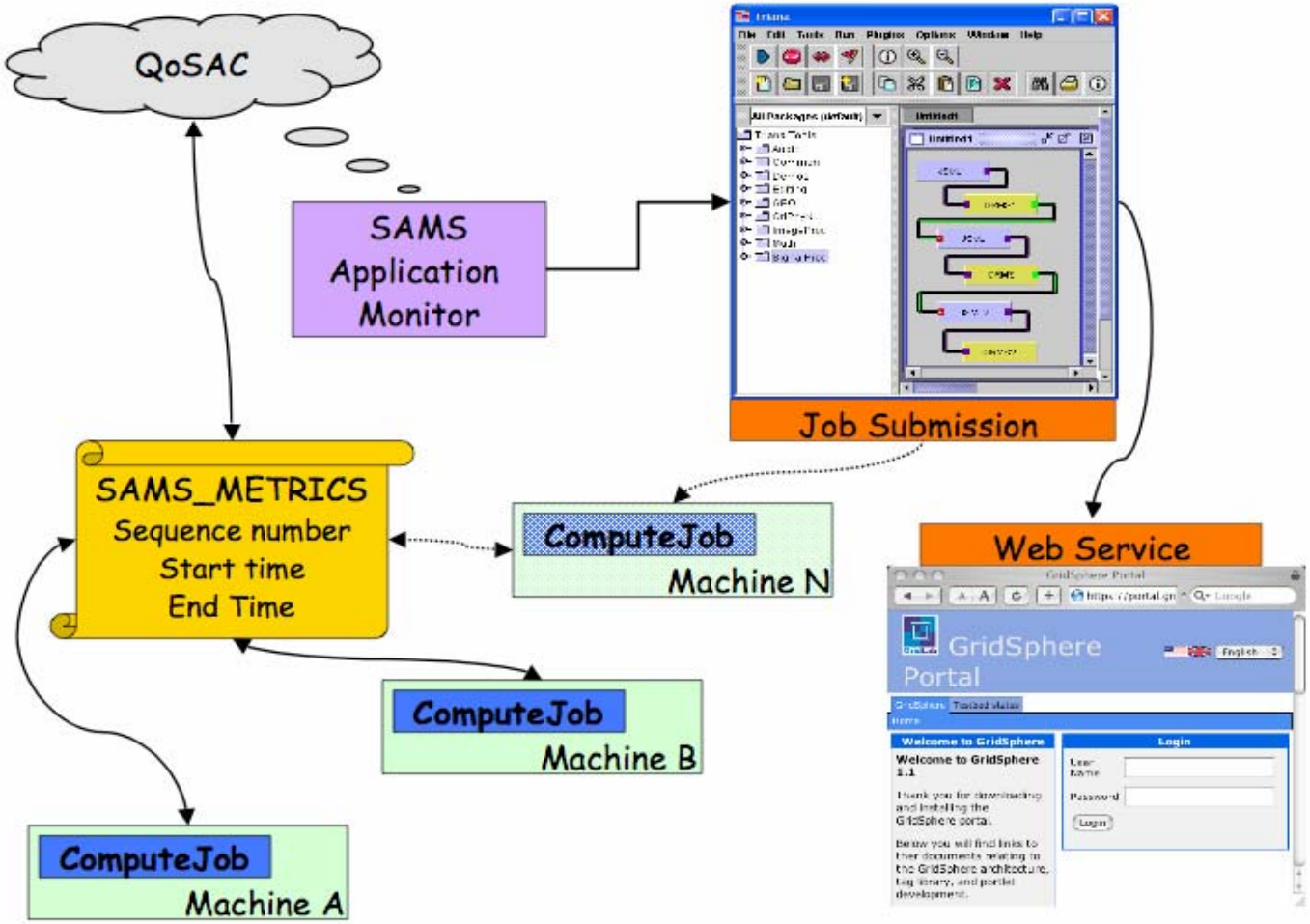
Applications/Collaborators

- EDG/EGEE:
  - GENIUS Integration - Triana running within GENIUS Portal (VNC Applet)
  - Workflow authoring - import job definition (JDL)/export Condor DAG
- GriPhyN/Chimera
  - Workflow authoring - import VDL, export DAX
- DIPSO
  - Multi-variate problems in Engineering
  - Choreographing web services
- GEMSS: (FP5 project)
  - Medical simulation
  - Application workflow, Choreographing web services

Future Work

- Provenance - electronic lab book
  - Part implemented - reproduce results
  - Store workflow, data objects, transitions
- Component & service checking
  - Versioning - is this the same version I used last time?
  - Verifying - who provided this?
  - Validation - does this do what I think it should?
- WSPeer - hosting WS in P2P environment
  - WS-RF implementation in P2PS
  - UDDI discovery replaced by P2P discovery

Simple Application Monitoring System



Simple Application Monitoring System

- SAMS implemented as Triana workflow
  - Each running service returns applicaton metric via GridLab Monitoring Service
  - QoS adaptive component retrieves metric and makes decisions about application
  - Job submission component start new service or releases existing service

Conclusion

- Distinct workflow types
  - Serial scientific workflow representing the algorithm
  - Job submission workflow to submit grid jobs that deploy multiple Triana Services on remote resources
  - Monitoring workflow examine & modify executing application
- GAP API
  - Web Service binding + GSI - Grid Job Submission
  - P2PS binding - service discovery + service communication
- Combined to perform parallel scientific computation

Thanks!

- The Astronomers: Prof. B Sathyaprakash, David Churches, Roger Philp and Craig Robinson

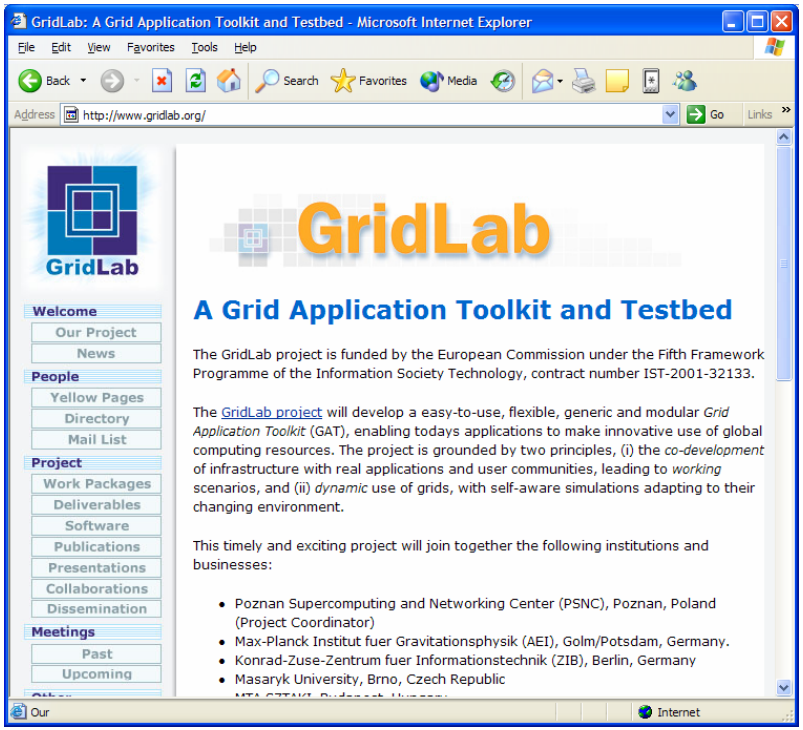
- The Triana team: Ian Wang, Andrew Harrison, Omer Rana, Diem Lam and Shalil Majithia

- All the partners in the GridLab project

Links

Information & Software

http://www.trianacode.org/



http://www.gridlab.org/