pvmanager status

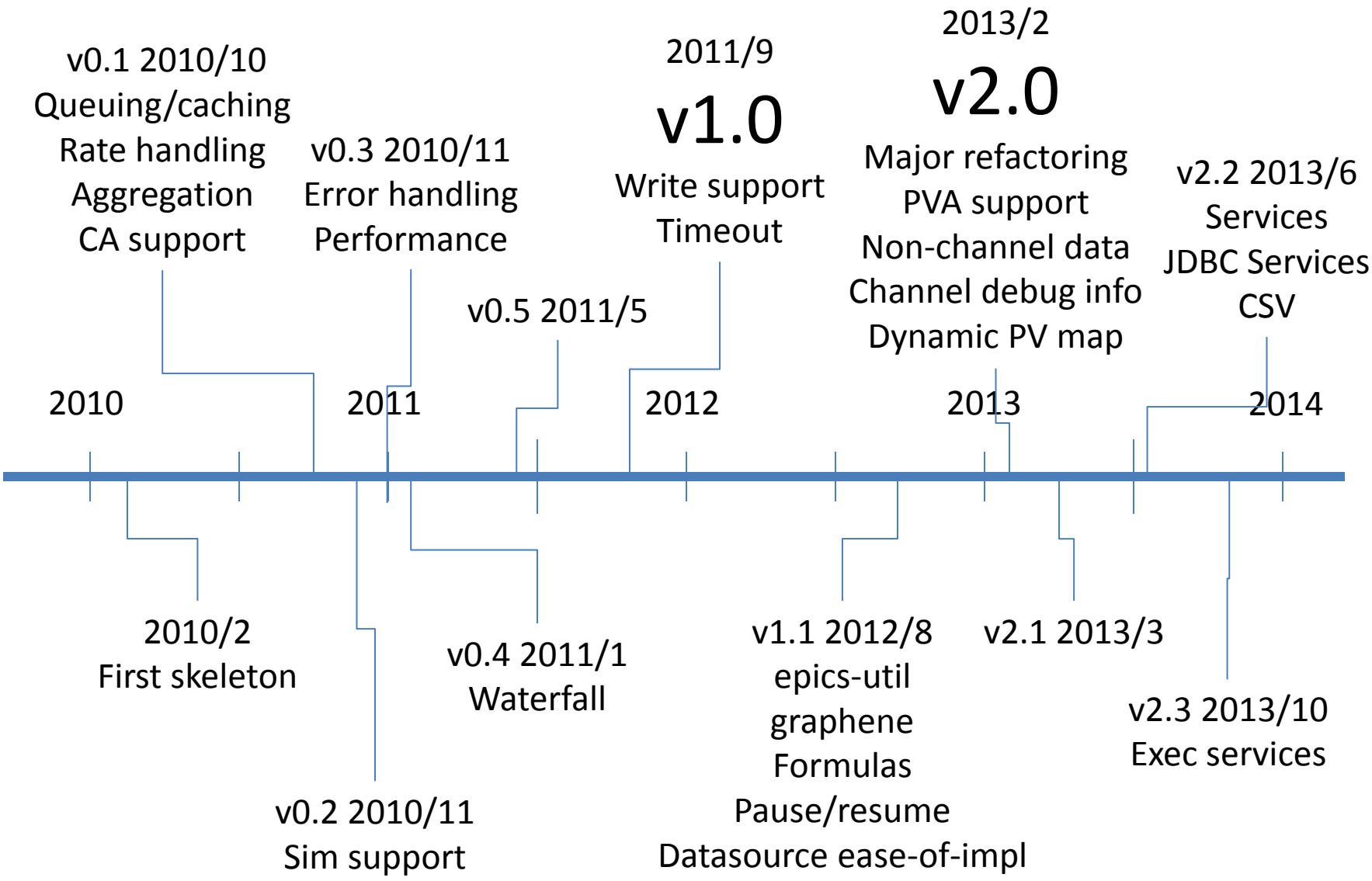
with tool strategy and architecture

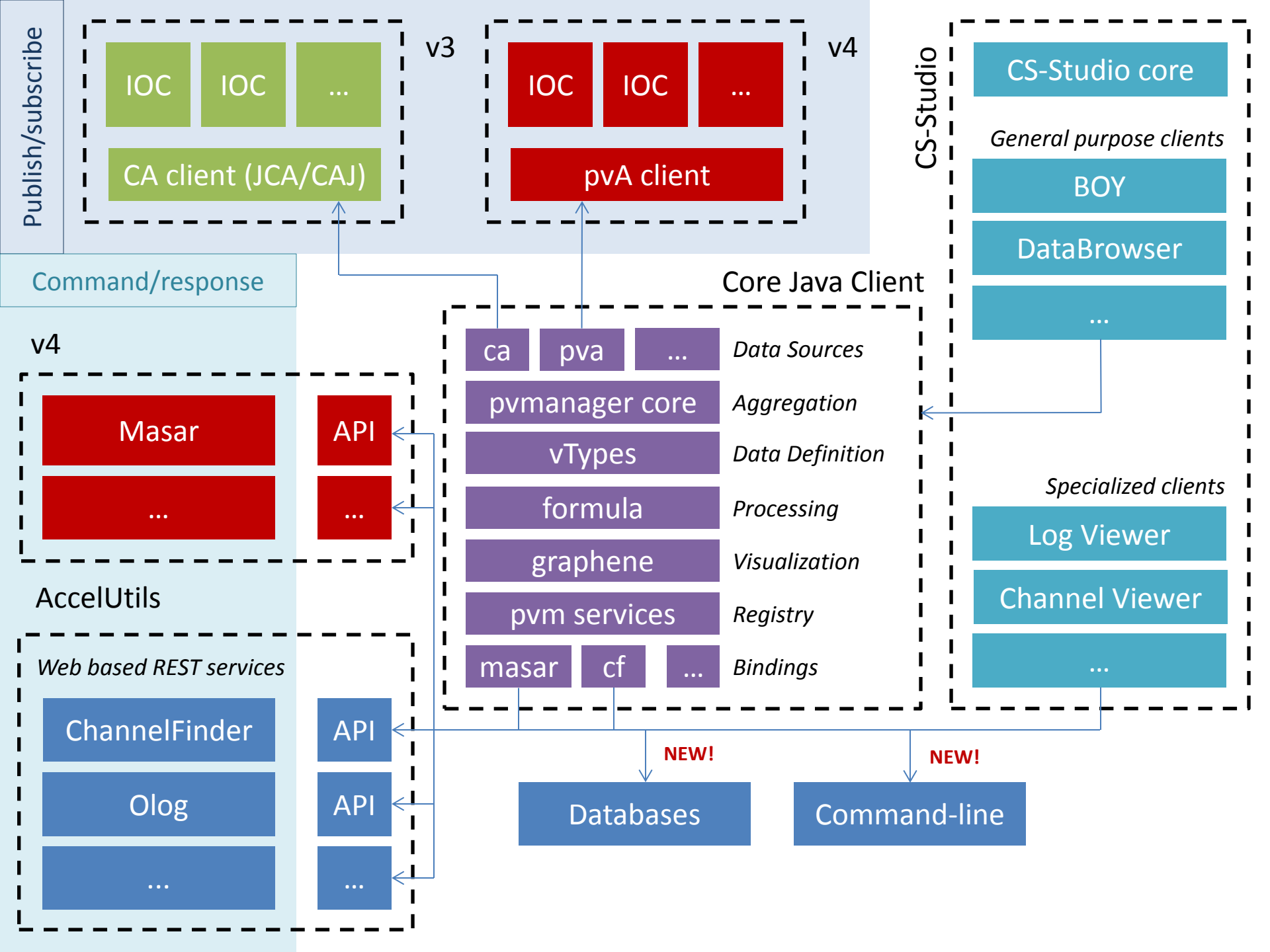
Gabriele Carcassi - BNL

Architecture objectives

- Architecture that can last for the next decade
  - A place for everything and everything in the right place
- Modularity
  - Allow small contributions
  - Lower the risk, compartmentalize failure
  - Cooperate on parts common to other labs without forcing the whole
- Allows use of coming technologies
  - Multi-core
  - High-resolution displays
  - Tablets
  - GPU

pvmanager timeline




Publish/subscribe
v3
IOC
IOC
...
CA client (JCA/CAJ)
v4
IOC
IOC
...
pvA client
CS-Studio
CS-Studio core
General purpose clients
BOY
DataBrowser
...
Command/response
Core Java Client
v4
ca
pva
...
Data Sources
pvmanager core
Aggregation
vTypes
Data Definition
formula
Processing
graphene
Visualization
pvm services
Registry
masar
cf
...
Bindings
Masar
...
API
...
Specialized clients
Log Viewer
Channel Viewer
...
AccelUtils
Web based REST services
ChannelFinder
Olog
...
API
API
...
NEW!
NEW!
Databases
Command-line

New in pvmanager ca datasource

- No resources to officially support JCA (CAJ only)
  - It kind of works, but no time to make sure it does
    - Couldn't even get it to work on Win64
  - JCA has 2 or 3 actual implementation inside. CAJ has one. All these implementations do not follow the same multi-threading semantic
    - You can't be sure on what thread you are notified and with what lock
  - It makes it impossible to have one implementation for everything that is both correct and efficient
  - Better to invest the time in just one version and fix all the outstanding issues there

Maintainer for JCA (JNI) and pvmanager integration

New in pvmanager ca datasource

- Improvements in CAJ (Murali + Matej + Gabriele)
  - More efficient handling of large arrays (not so new)
  - More uniform to C client library (not so new)
    - DBE_PROPERTY support
    - Variable sized array
    - MAX_ARRAY_BYTES
  - Fixing hard to reproduce bugs
    - Infinte loop in reconnect
    - Large number (750000) of channels connecting to the gateway

Murali and Matej are awesome!

- Improvements in CA pvmanager datasource
  - Encapsulate “quirks” from CA (not so new)
    - Value only for RTYP
    - Long string support (reads VString from byte array)
    - Can choose between put and put-with-callback
    - Autodetects if var array is supported
  - Metadata monitor (not so new)
  - Various bug fixes and performance improvements from usage (not so new)
    - Lower connection priority for large arrays

New in pvmanager pva datasource

- Support for EPICS v4 pvAccess (in CS-Studio)
  - Publish/subscribe support (see later for services)
- Types supported:
  - Scalar (VNumber, VEnum and VString)
  - Array (VNumberArray and VStringArray)
  - Table
  - Matrices (VNumberArray 2D)
  - Images
- Work done Matej

New file datasource

- File datasource, allows to read files
  - file://path/to/file
  - Parses CSV tables
  - Handles images (png, bmp)
- It's mainly a stub
  - Add notification if file is update
  - Improve CSV parsing
  - Support for other files (matlab, excel, HDF5, …)

Looking for volunteer to expand it

Goals for pvmanager core

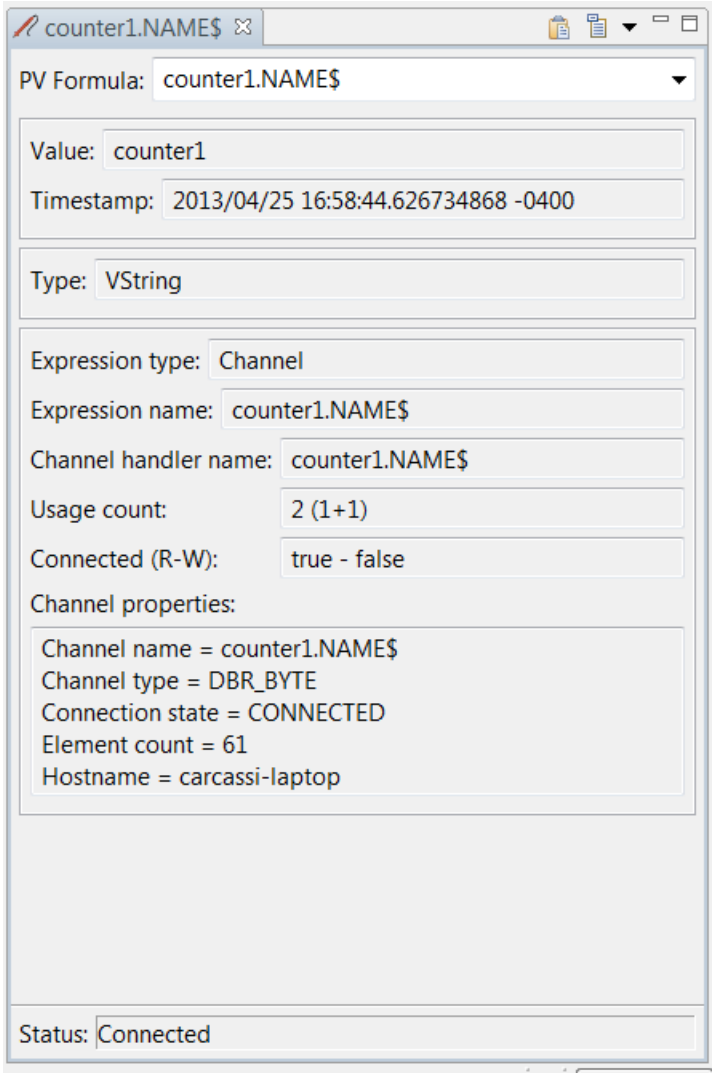
- Provide logic required by well-behaved application
  - Queuing or caching
  - Rate decoupling between subsystems (typically CA and UI, but also CA and anything else)
  - Rate limiting (notification rate capped)
  - Rate throttling (notification rate adapts, skips instead of building up lag)
  - Shared connections (similar requests on a single monitor)
  - Ability to offload work on a shared pool
- You still need to understand what all of this is
- You just don’t have to implement it

New in pvmanager core

- Becoming the main way to access data in CS-Studio
- Significant refactoring in 2.0 (02/2013)
  - Cleaner interface
  - Dynamic map
    - add/remove pvs, always get a single event with all values
  - Channels can report properties for extra debugging information

New in pvmanager core

- Probe can display the extra properties
- Note the long string support:
  - The value is vString
  - The channel type is DBR_TYPE



epics-util

- Placeholder for utility classes that hopefully will appear in standard Java at some point
- org.epics.util.array – numeric collection
  - Read only view, lazy computation/functional programming, single binding for all numeric types instead of 6 (double, float, long, int, short, byte), circular buffers, sorted views
- org.epics.util.time – time definitions and formats
  - Timestamp, TimeDuration, … while we wait for Java 8

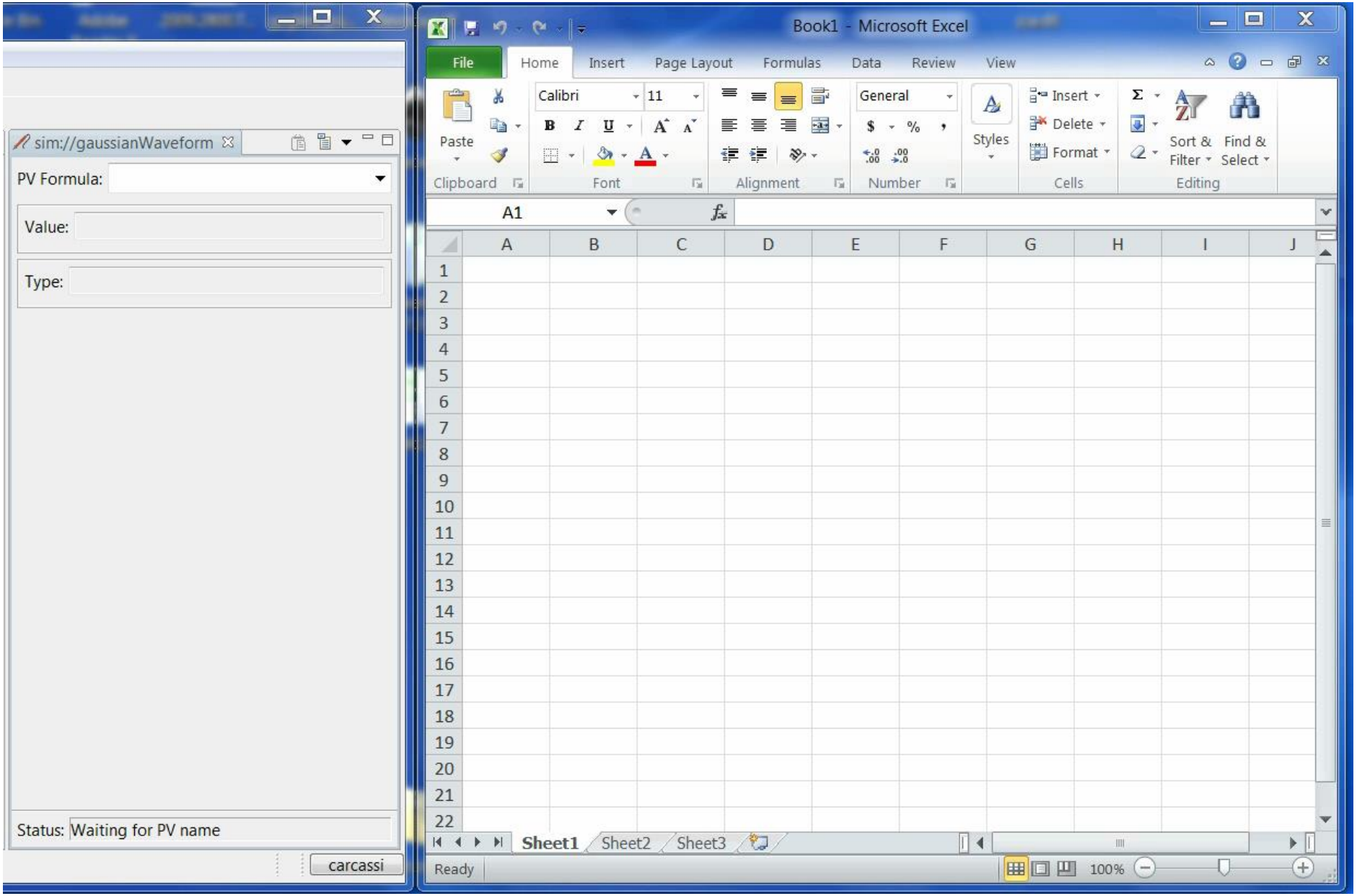
EPICS macro substitution,
unsigned primitive math,
common configuration mechanism

New in vTypes

- Control system data interfaces for Java client
- Taken out of pvmanager so they can be used for services
- VNumber and VNumberArray superclass
- Value to text formatting
- More consistent toString implementation
- VTable columns can contain timestamps
- First steps for exporting to CSV

Improve import/export (CSV, matlab, HDF5), common value formatting

EXPORT VTYPE TO EXCEL


sim://gaussianWaveform
PV Formula:
Value:
Type:
Status: Waiting for PV name
carcassi
Book1 - Microsoft Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Paste
Clipboard
Calibri
11
Font
Alignment
General
Number
Styles
Insert
Delete
Format
Cells
Sort & Filter
Find & Select
Editing
A1
Sheet1
Sheet2
Sheet3
Ready
100%

Goals for pvmanager formulas

- Current problems:
  - New type (i.e. table) or new service (i.e. channelfinder) means having to write new widgets and applications
    - Difficult to extend the tools
  - Each widget and application has to implement behavior that may be common, for example:
    - Display the alarm/time instead of the value
    - Display the value of the enum instead of the label
    - Display the FFT instead of the waveform
    - Display value in different unit
    - Options will be slightly different, more work and code to maintain
- This functionality should neither be in the datasources nor the widget
  - Need a better place!

pvmanager formulas

- Allow to specify formulas everywhere you had pvs
- Currently implemented

  ‘mypv’ pvs in single quotes

  “Some text” text literals in double quotes

  3.14 numeric literals with standard C/C++/Java notation

  ‘pv1’ – ‘pv2’ standard math operators

  log(‘mypv’) functions, pluggable in CSS
  java.lang.Math methods already implemented

  arrayOf(‘pv1’, ‘pv2’, ‘pv3’) coverts scalars into arrays

  columnOf(‘tablepv’, “column1”) column from table as an array

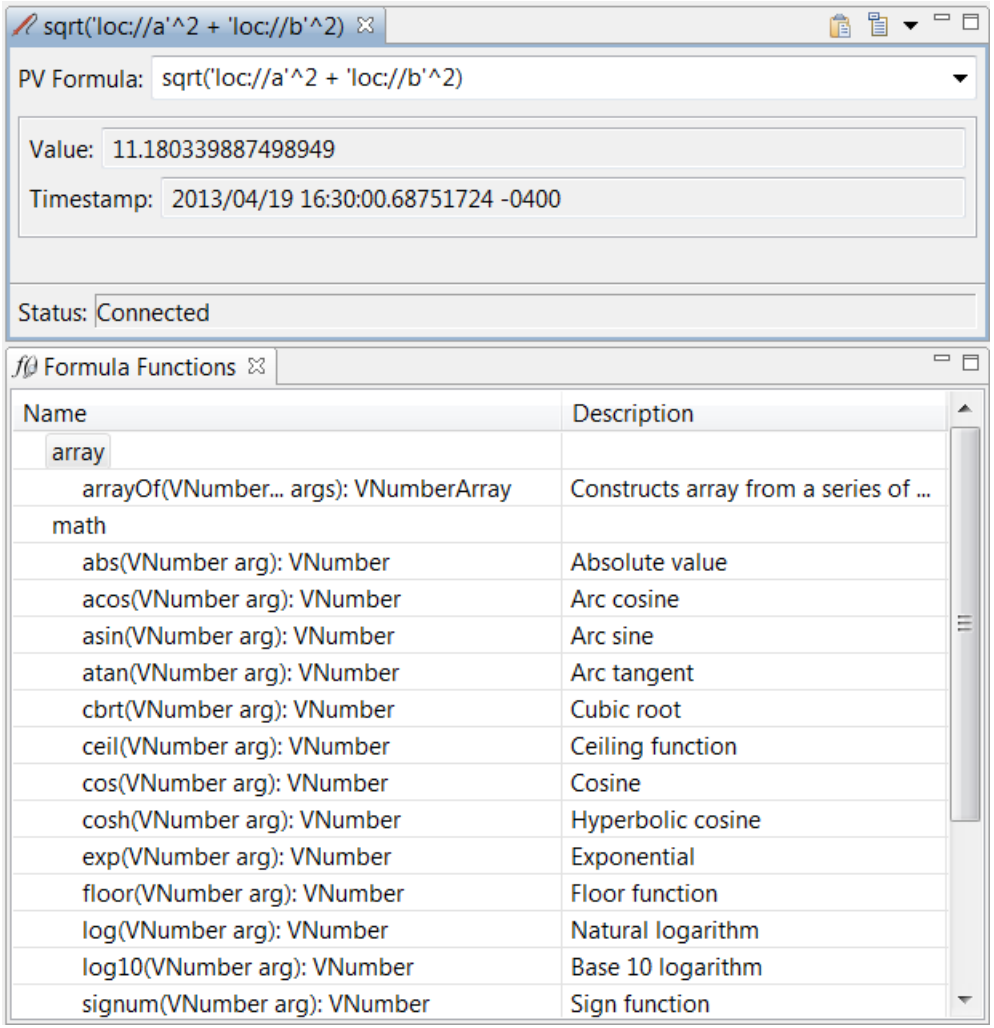
Unit conversion, function fit, linear algebra, full join

Functions

- Numeric operators
  - + - * / % ^ **
  - > < >= <= == !=
  - ?:
  - || &&
  - | &
- Math functions
  - abs
  - acos
  - asin
  - atan
  - cbrt
  - ceil
  - cos
  - cosh
  - exp
  - floor
  - integrate
  - log
  - log10
  - signum
  - sin
  - sinh
  - sqrt
  - tan
  - tanh
  - toRadians
  - toDegrees
- Array functions
  - + - / *
  - arrayOf
  - elementAt
  - rescale
  - subArray
- Table functions
  - columnOf
  - join
  - tableOf
- String function
  - concat
  - toString
- Pointer-like function
  - pv
  - pvs
- Other functions
  - highestSeverity

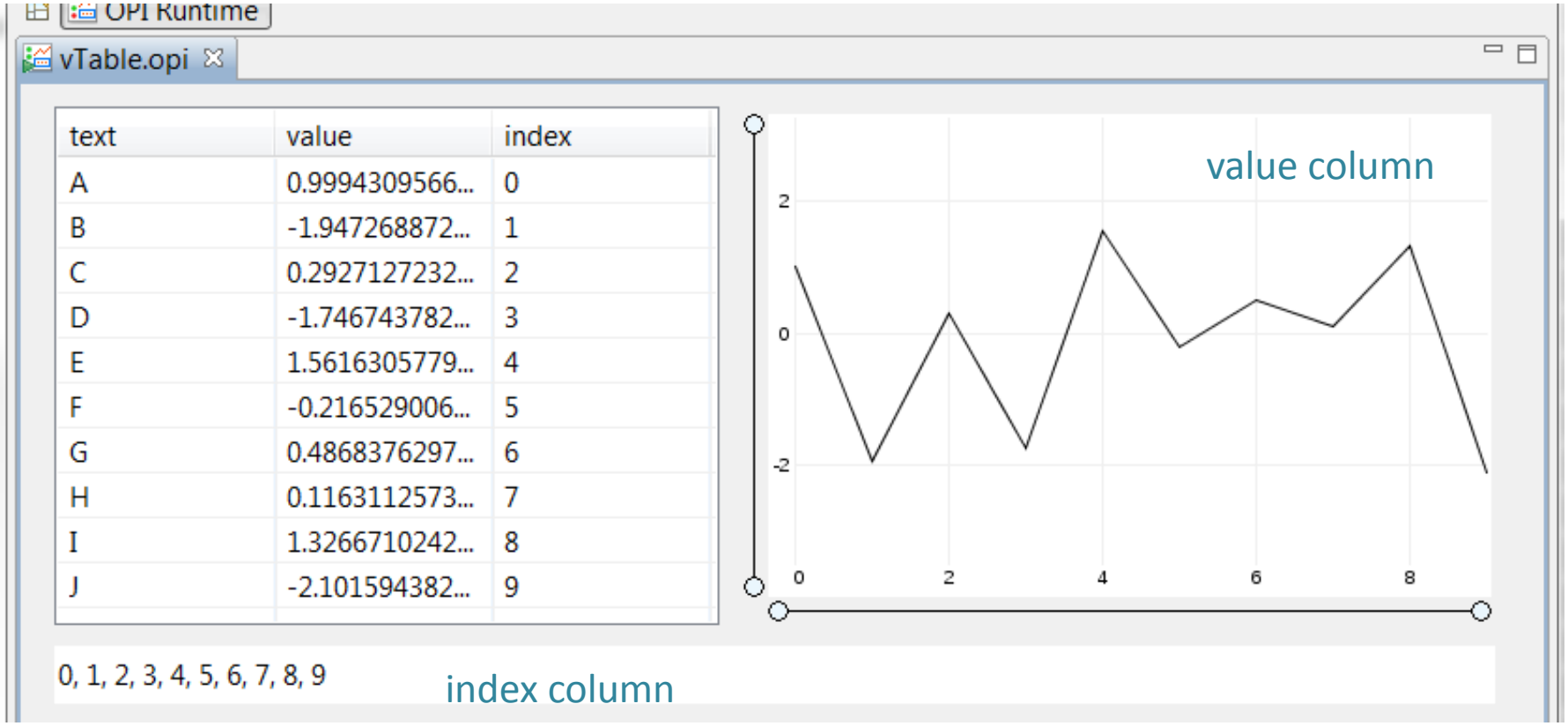
pvmanager formulas

- Formula in Probe
- Function viewer lists all the contributions and serves as documentation



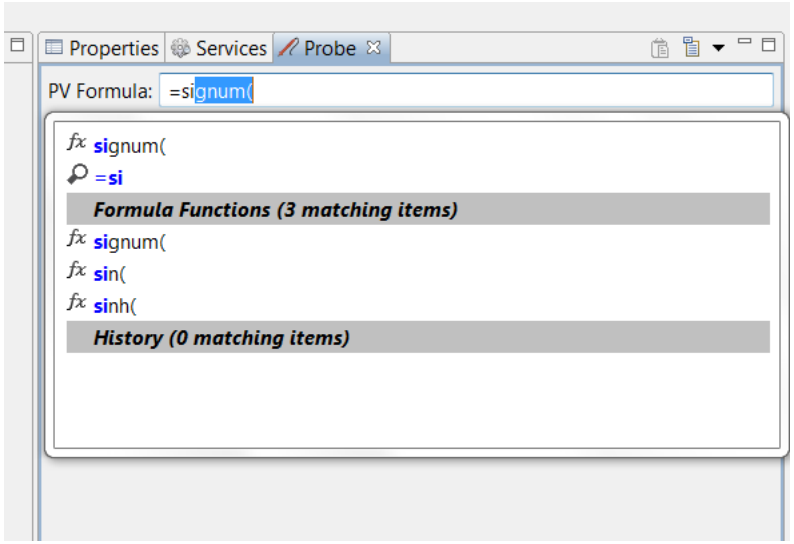
pvmanager formulas

- columnOf('sim://table', "value")



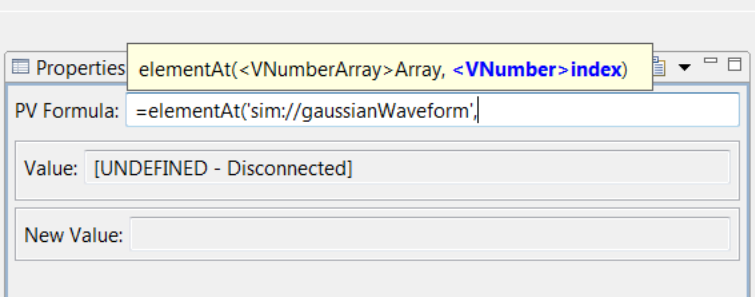
pvmanager formulas

- Can register functions through extension point
- We expect to:
  - Significantly increase the functionality
  - Reduce the complexity
    - Fewer widgets
    - Fewer parameters per widget
  - Increase consistency within CS-Studio
- If formulas become important for the whole of CS-Studio, makes it possible to have:
  - Formula editors
  - Syntax coloring
  - Autocomplete (Already being contributed by ITER!)



Formula auto-complete

Formula auto-documentation



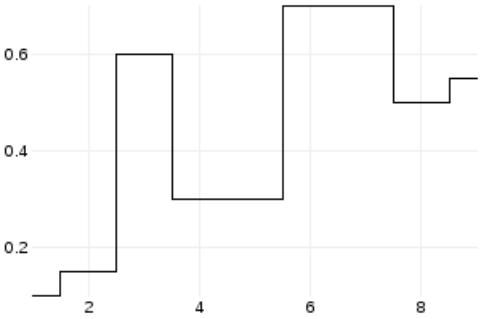
Goals for graphene

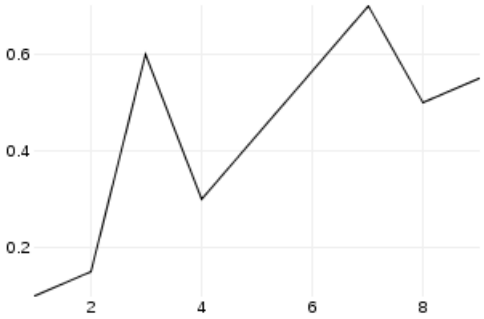
- One year ago we started working on a graph package with the following goals:
  - Interfaces for datasets
  - Rendering without UI
  - Changes are thread-safe and atomic
  - Publishable quality
  - Performance suitable for “large data” in real time

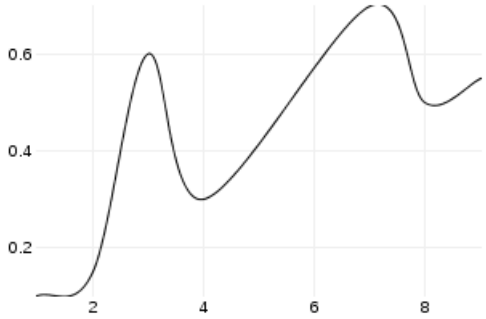
New in graphene

- Current status
  - Further ahead but not as far as I'd like
  - Four graphs are in decent shape:
    - Scatter graph
    - Line graph
    - Area graph
    - Bubble graph
  - Common patterns are starting to drive the architecture
    - Common elements to create the axis
    - Common elements to handle the dynamic range

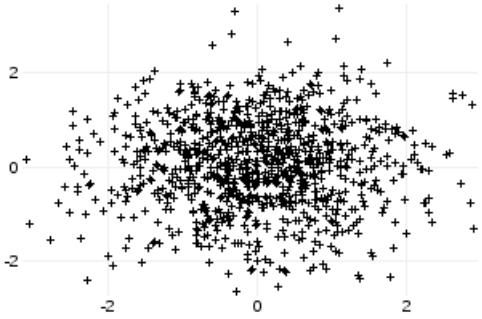
Line graph

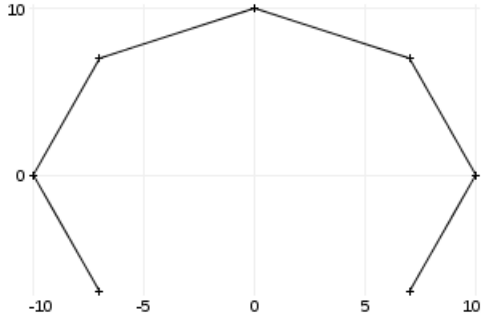




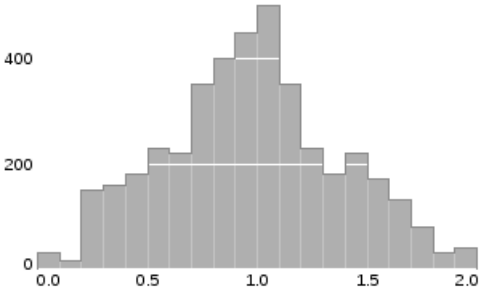


Scatter graph

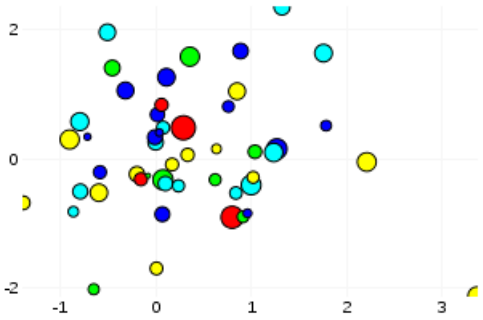




Area graph (histogram)



Bubble graph



Graphene in CS-Studio

- Line graph and scatter graph available as:
  - SWT widgets
  - BOY widgets
  - Independent views
- Table is king
  - Re-implemented to work off of tables
  - Does not work for histogram and intensity graphs, though
    - Have already ideas on how to extend VNumberArray with range information

CS-Studio

File Edit Search CSS Window Help

100%

OPI Runtime

scattergraph-axis.opi

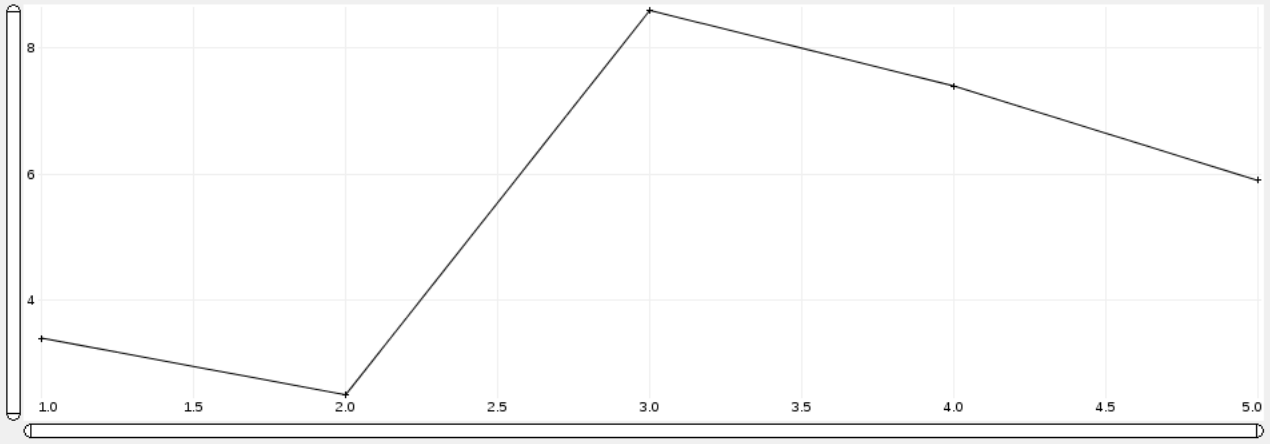
Scatter Graph - Dynamic axis

Example 1: scatter graph where the axis are user selected.

On the right we have a table constructed by using 3 different waveform. On the bottom we have a bubble graph constructed from it. You can see the uniform noise distribution on the Y axis and the gaussian peek scrolling right to left for the bubble size.

X axis: A

Y axis: B



A	B	C	D
1.0	3.4	100.0	3.1
2.0	2.5	400.0	-7.6
3.0	8.6	300.0	-2.2
4.0	7.4	800.0	0.5
5.0	5.9	300.0	4.1

carcassi

CS-Studio

File Edit Search CSS Window Help

100%

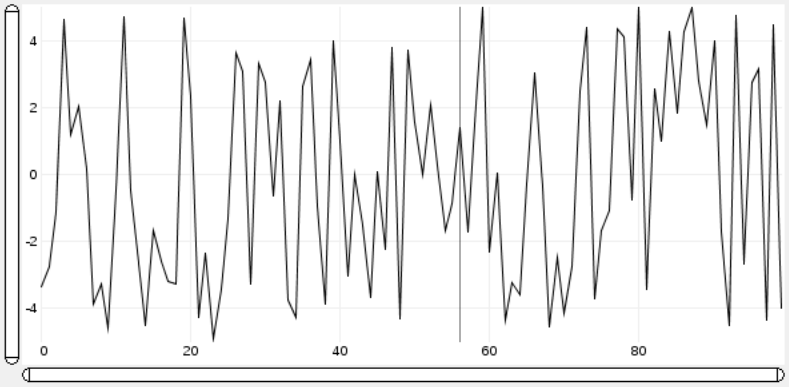
OPI Runtime

linegraph-selection.opi

Line Graph - Selection

Example 1: value selection from line graph based on a table.

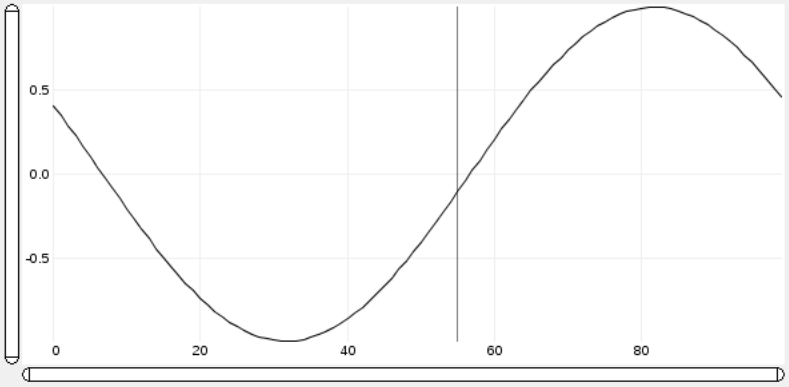
Moving the mouse over the graph will populate the table with the row corresponding to the value highlighted. Note that the whole row is selected, not just the data used for the graph.



Index	Value	Extra
56.0	1.4155543442...	15.600000000...

Example 2: value selection from line graph based on a waveform.

The selection works also with data coming out of a waveform. X and Y column are generated from the index and value of the waveform.



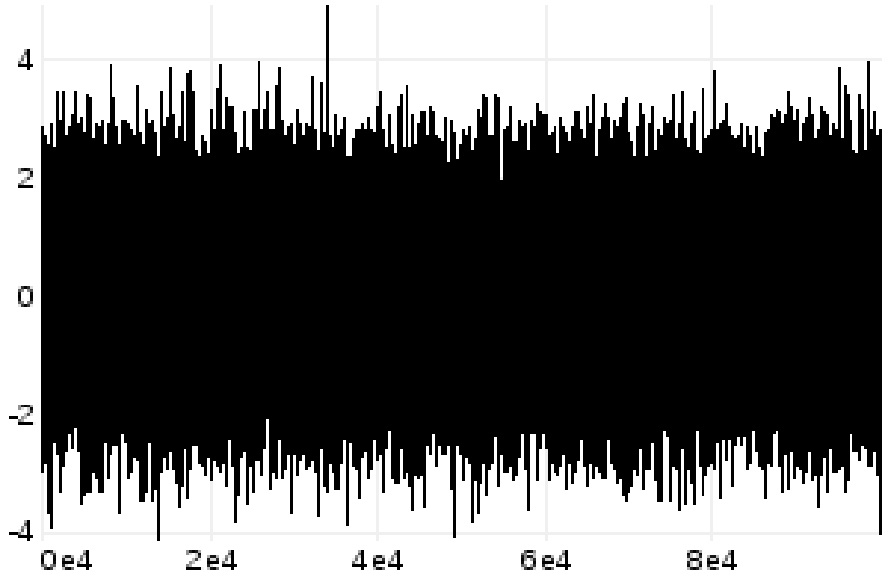
X	Y
55.0	-0.10435870986825124

carcassi

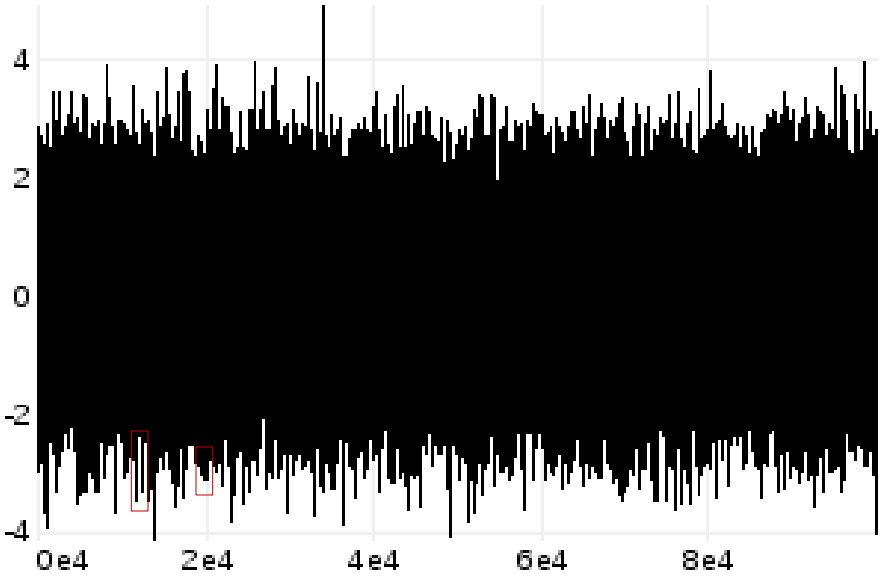
graphene performance

- LineGraph performance improved through data reduction
  - For each pixel draw 4 values: first, max, min, last
- Displaying 100,000 points, 300x200
  - How many differences can you find?

Without data reduction - 255.4 ms



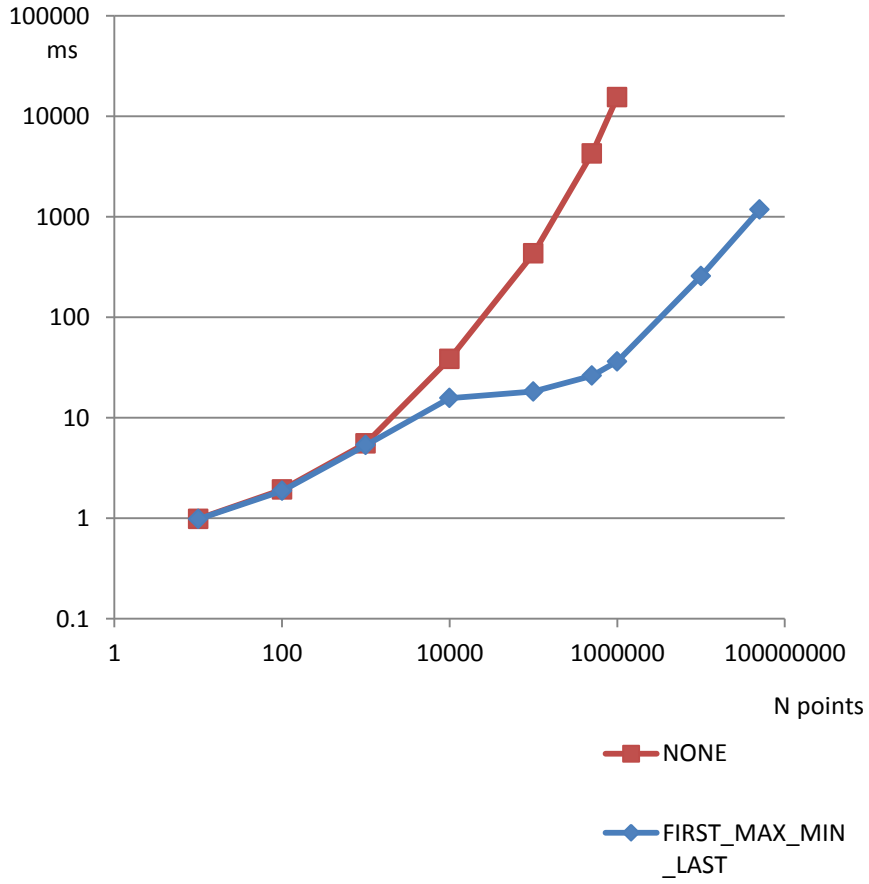
With data reduction - 6.077 ms



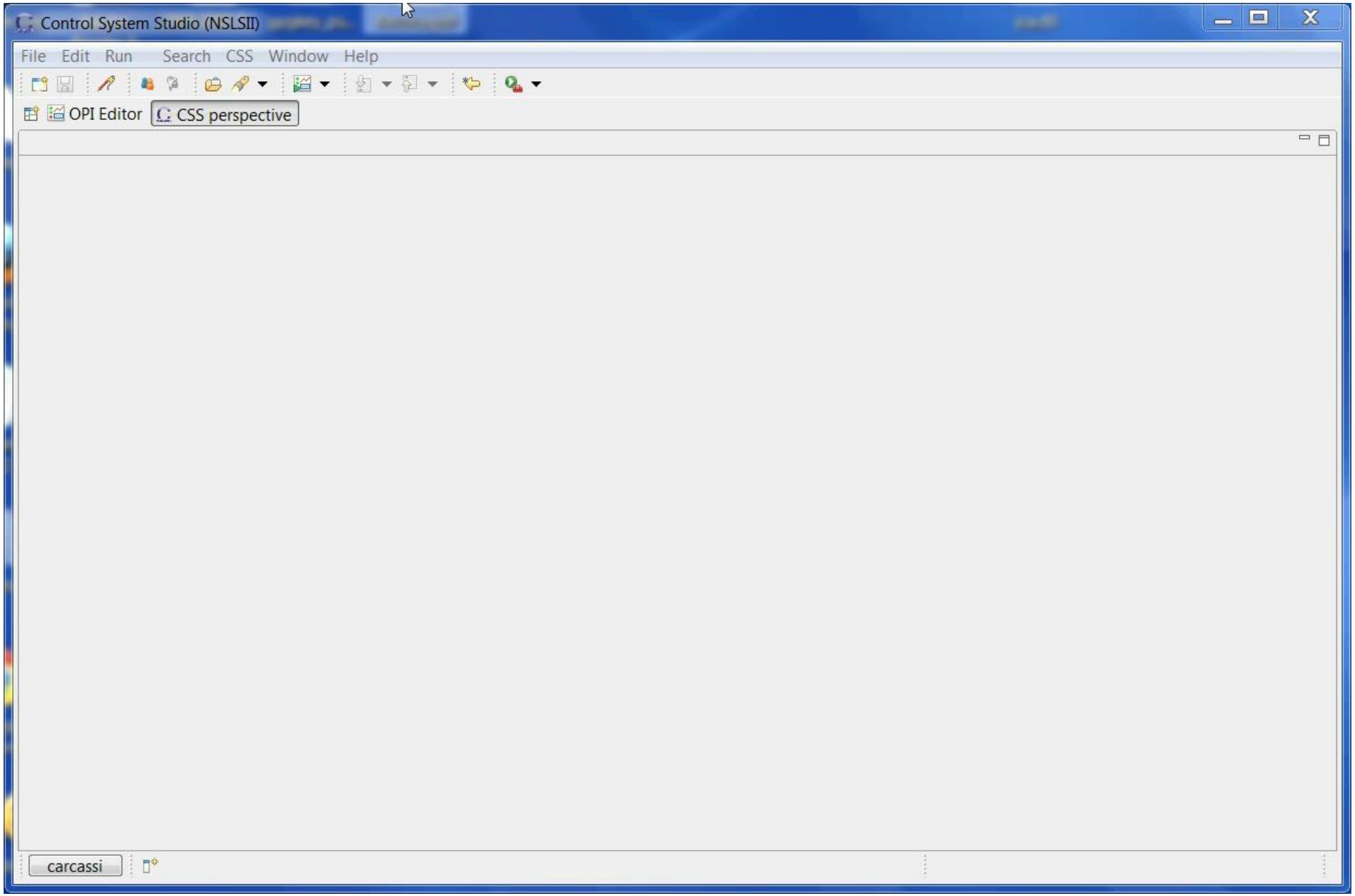
graphene

- Line graph, 600x400, JDK 7, i7 2.70GHz

N points	Time ms (no redux)	Time ms (redux)
10	0.986232349	0.980788919
100	1.924796332	1.872743481
1,000	5.553831552	5.345118646
10,000	38.57528235	15.73147583
100,000	431.8435542	18.28121567
500,000	4280.394044	26.20789778
1,000,000	15556.68846	36.36819781
10,000,000		257.451379
50,000,000		1178.120411



GRAPHENE LINE GRAPH DEMO


Control System Studio (NSLSII)
File Edit Run Search CSS Window Help
OPI Editor
CSS perspective
carcassi

Goals for pvmanager services

- Services cannot be integrated with standard pvmanager: *a service is not a pv*
  - Needs parameters
  - Needs someone to click send (services are not idempotent in general)
  - Each client needs its own return value
  - The result can be complex and may need several widgets to display
- In general, you'll need service specific APIs and applications, with specific workflow
  - There is no way around it

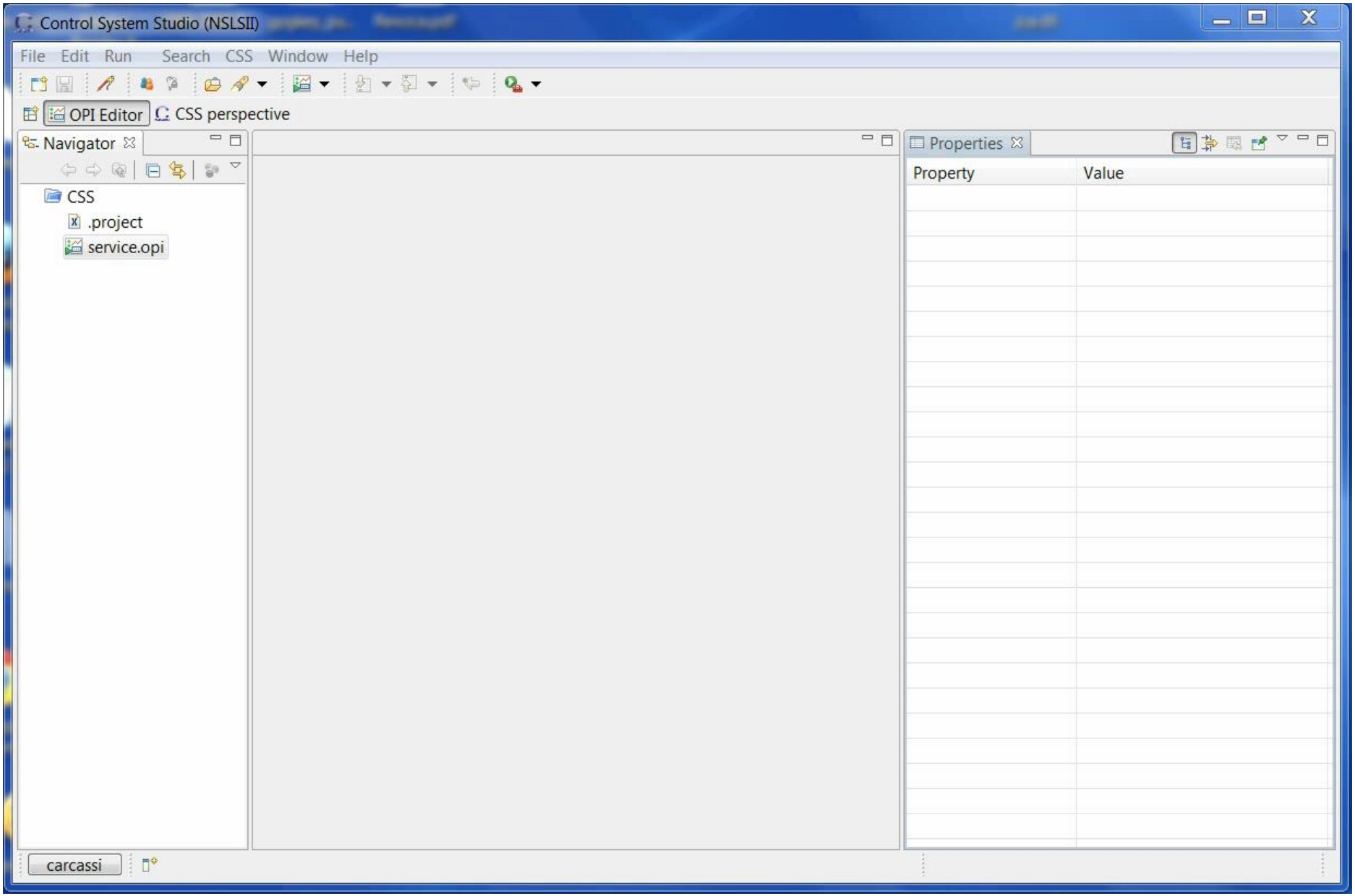
Goals for pvmanager services

- BUT! Some services in some cases
  - Have parameters that can be expressed by vTypes (e.g. VString and VNumbers)
  - Can return data that can be expressed in a vTable, vNumericArray, …
  - These can be digested by general purpose clients
- For example, channel finder
  - Will need a specific UI to manage tags and properties
  - Can export into a table the result of a query
- For the general purpose **only** we introduce pvmanager services

pvmanager services

```
public abstract class ServiceMethod {

    public String getName();
    public String getDescription();

    public Map<String, Class<?>> getArgumentTypes();
    public Map<String, String> getArgumentDescriptions();

    public final Map<String, Class<?>> getResultTypes();
    public Map<String, String> getResultDescriptions();

    public abstract void executeMethod(
       Map<String, Object> arguments,
       WriteFunction<Map<String, Object>> callback,
       WriteFunction<Exception> errorCallback);
}
```

PVMANAGER SERVICES DEMO


Control System Studio (NSLSII)
File Edit Run Search CSS Window Help
OPI Editor
CSS perspective
Navigator
CSS
.project
service.opi
Properties
Property
Value
carcassi

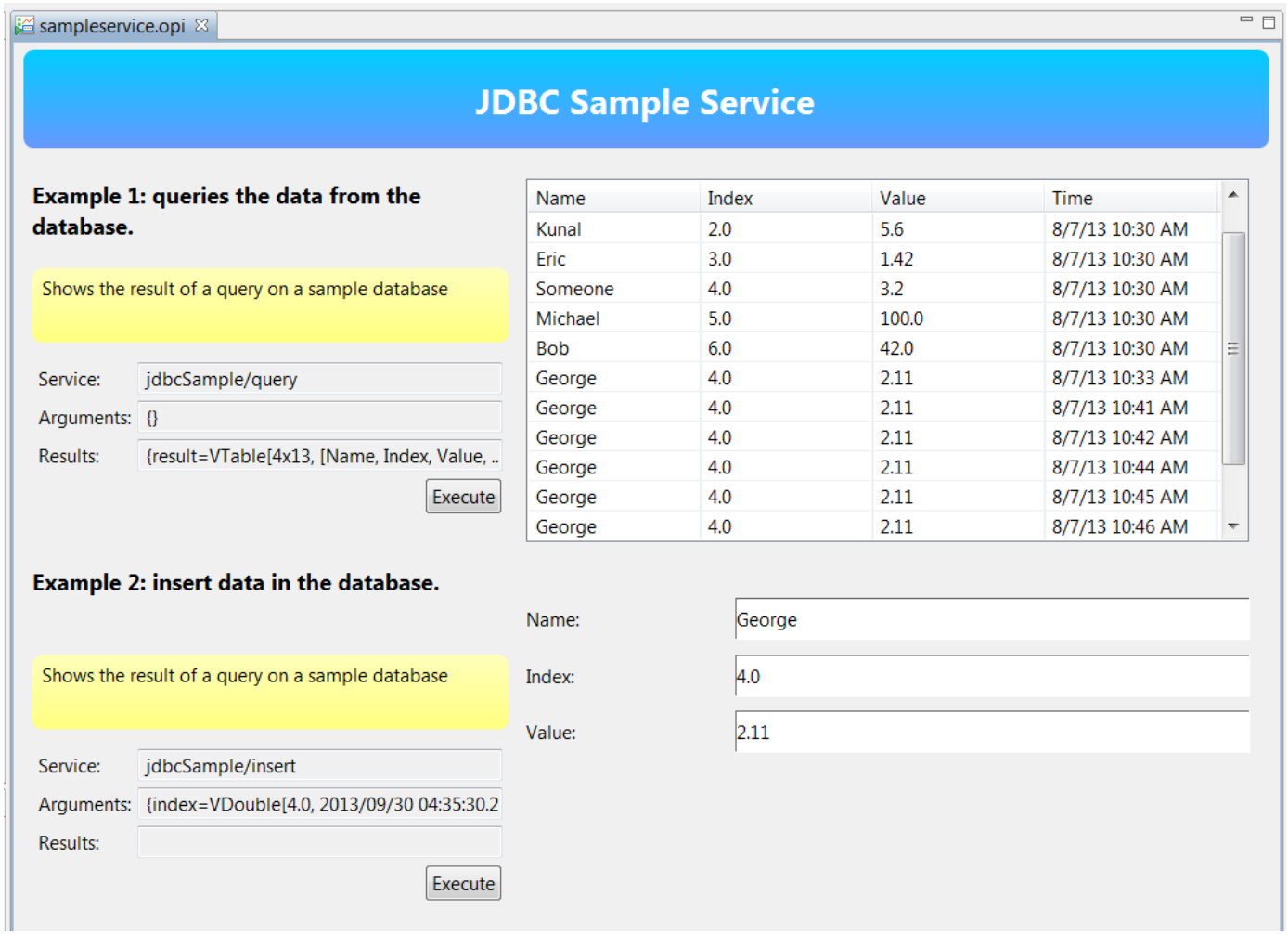
jdbc services

- Services that execute database queries
  - Xml files can be places in $cs-studio/configuration/services/jdbc
  - Each service is a database with a set of parameterized queries
  - Parameters can be VString/VNumber/…
  - Results are VTable

Maintainer to extend implementation

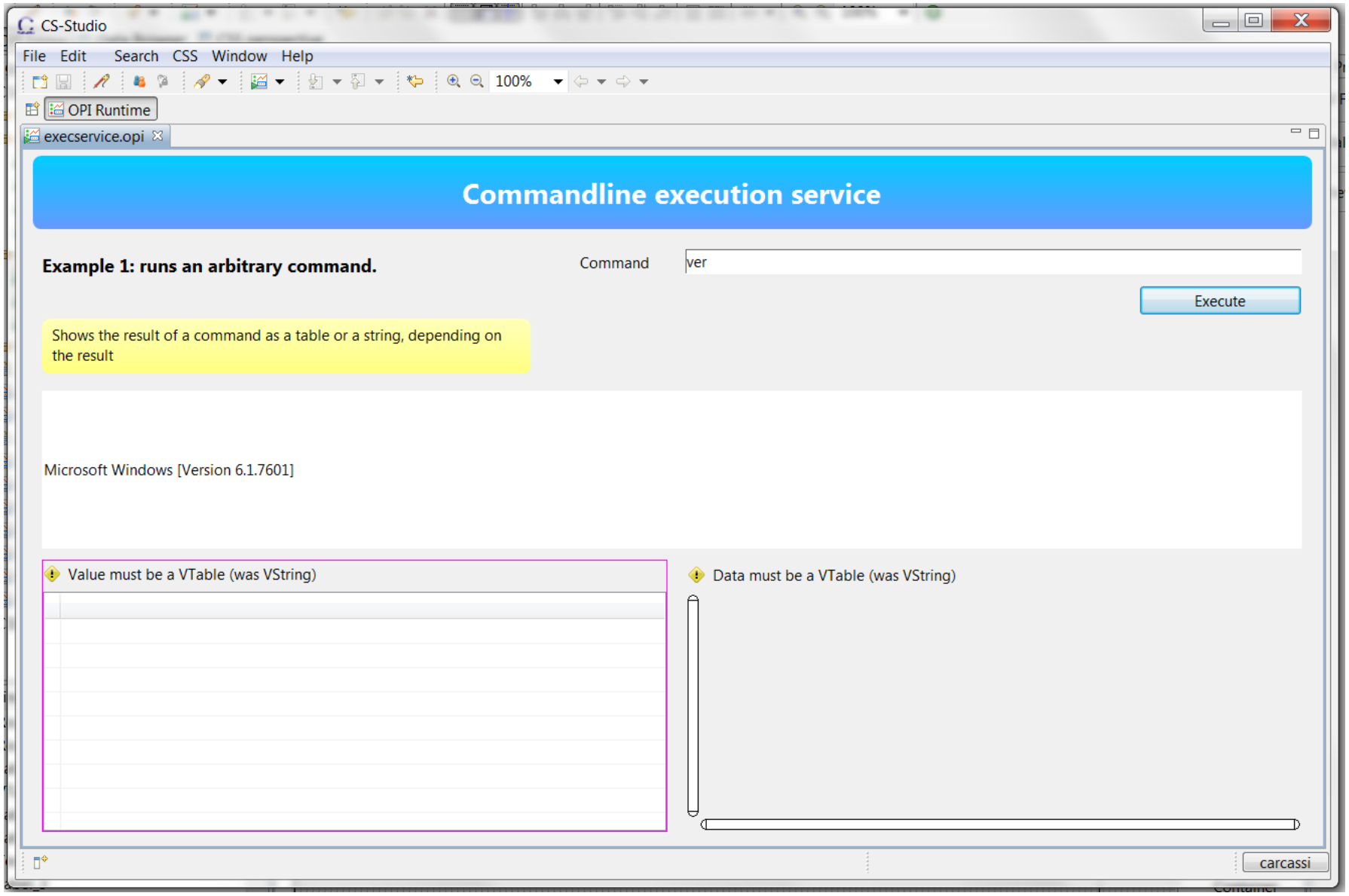
jdbc service example

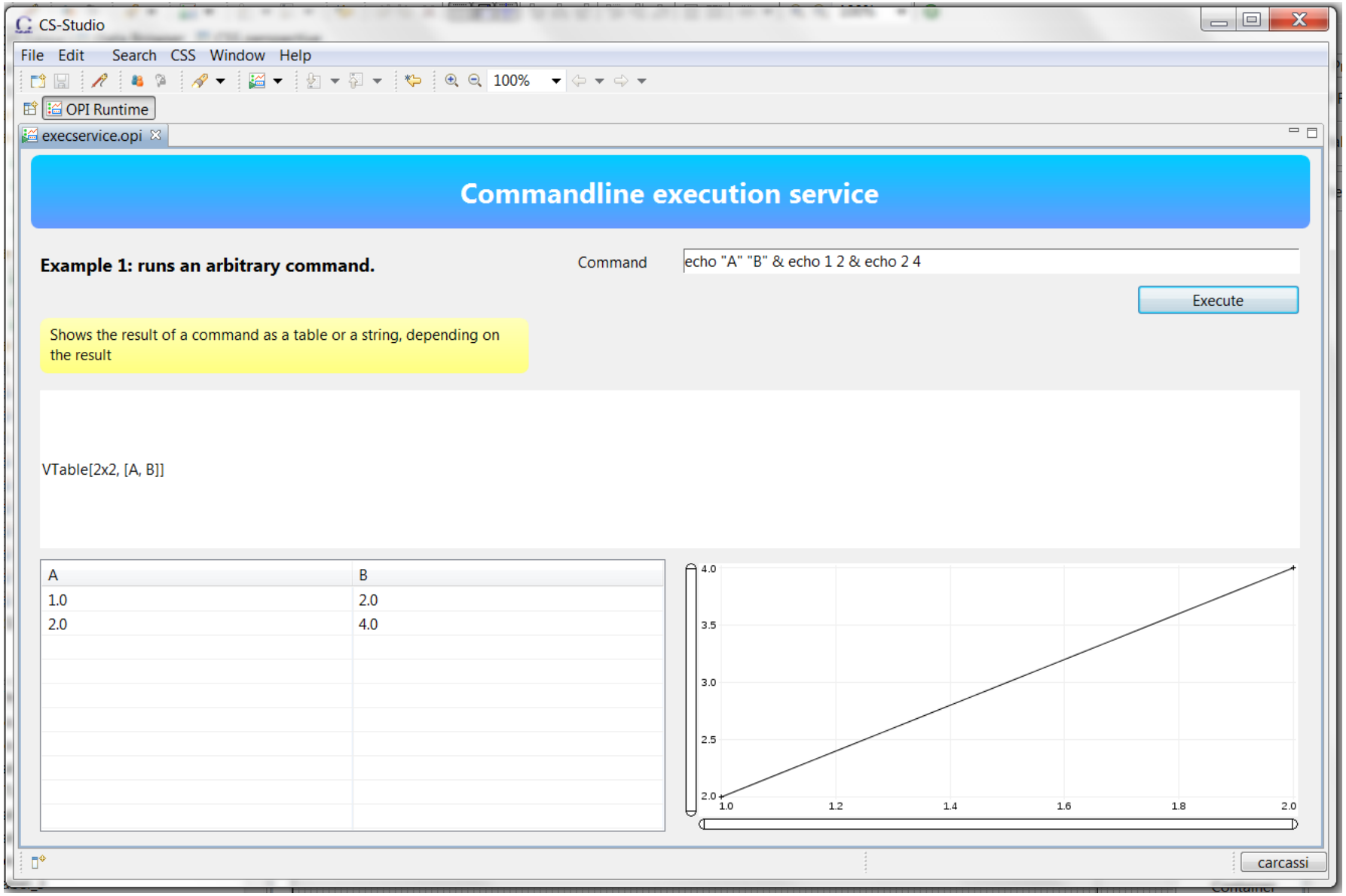
```
<?xml version="1.0" encoding="UTF-8"?>
<jdbcService ver="1" name="jdbcSample" description="A test service">
 <jdbcUrl>jdbc:mysql://localhost/test?user=root&amp;password=root</jdbcUrl>
 <methods>
  <method name="query" description="A test query">
   <query>SELECT * FROM Data</query>
   <result name="result" description="The query result"/>
  </method>
  <method name="insert" description="A test insert query">
   <query>INSERT INTO `test`.`Data` (`Name`, `Index`, `Value`) VALUES (?, ?, ?)</query>
   <argument name="name" description="The name" type="VString"/>
   <argument name="index" description="The index" type="VNumber"/>
   <argument name="value" description="The value" type="VNumber"/>
  </method>
 </methods>
</jdbcService>
```


sampleservice.opi
JDBC Sample Service
Example 1: queries the data from the database.
Shows the result of a query on a sample database
Service:
jdbcSample/query
Arguments:
{}
Results:
{result=VTable[4x13, [Name, Index, Value, ..
Execute
Name
Index
Value
Time
Kunal 2.0 5.6 8/7/13 10:30 AM
Eric 3.0 1.42 8/7/13 10:30 AM
Someone 4.0 3.2 8/7/13 10:30 AM
Michael 5.0 100.0 8/7/13 10:30 AM
Bob 6.0 42.0 8/7/13 10:30 AM
George 4.0 2.11 8/7/13 10:33 AM
George 4.0 2.11 8/7/13 10:41 AM
George 4.0 2.11 8/7/13 10:42 AM
George 4.0 2.11 8/7/13 10:44 AM
George 4.0 2.11 8/7/13 10:45 AM
George 4.0 2.11 8/7/13 10:46 AM
Example 2: insert data in the database.
Shows the result of a query on a sample database
Service:
jdbcSample/insert
Arguments:
{index=VDouble[4.0, 2013/09/30 04:35:30.2
Results:
Execute
Name:
George
Index:
4.0
Value:
2.11

exec service

- Exec services, services based on command-line execution
  - Xml files can be places in $cs-studio/configuration/services/exec
  - Each service is a set of parameterized commands
  - Parameters can be VString/VNumber
  - Result is a VString or VTable (if output is a CSV)
  - Generic exec/run service to execute an arbitrary command (a VString)


CS-Studio
File Edit Search CSS Window Help
100%
OPI Runtime
execservice.opi
Commandline execution service
Example 1: runs an arbitrary command.
Command
ver
Execute
Shows the result of a command as a table or a string, depending on the result
Microsoft Windows [Version 6.1.7601]
Value must be a VTable (was VString)
Data must be a VTable (was VString)
carcassi


CS-Studio
File Edit Search CSS Window Help
100%
OPI Runtime
execservice.opi
Commandline execution service
Example 1: runs an arbitrary command.
Command
echo "A" "B" & echo 1 2 & echo 2 4
Execute
Shows the result of a command as a table or a string, depending on the result
VTable[2x2, [A, B]]
A
B
1.0
2.0
2.0
4.0
4.0
3.5
3.0
2.5
2.0
1.0
1.2
1.4
1.6
1.8
2.0
carcassi

exec service example

```
<?xml version="1.0" encoding="UTF-8"?>
 <execService ver="1" name="execSample" description="A test service">
  <methods>
   <method name="echo" description="A test script">
   <command>echo You selected #string#</command>
   <argument name="string" description="The string" type="VString"/>
   <result name="result" description="The result"/>
  </method>
  <method name="script" description="My script">
   <command>myscript.py #value#"</command>
   <argument name="value" description="The value" type="VNumber"/>
   <result name="result" description="The script result"/>
  </method>
 </methods>
</execService>
```

Maintainer to extend implementation

pva service

- Working with Cosylab to create a pva to pvmanager service binding that looks similar
  - Write xml file
  - Put it in the configuration directory
  - CS-Studio is aware of the service

Framework for real time data manipulation

- pvmanager datasources: pluggable sources of live data
- pvmanager services: pluggable sources of static data
- pvmanager formula functions: pluggable ways to combine the data
- Graphene graphs: hopefully efficient way to display the data
- VType export: ways to export the data

Users can now quickly and safely combine these!

Infrastructure

- Projects are on SourceForge
  - http://epics-util.sourceforge.net/
  - http://pvmanager.sourceforge.net/
  - http://graphene.sourceforge.net/
- Built on cloudbees (Jenkins in the cloud)
  - https://openepics.ci.cloudbees.com/
- Libraries deployed in maven central
  - http://search.maven.org/#search|ga|1|g%3A%22org.epics%22

Still a lot more work to do …

… if you want to help make progress

Take something over

- Having an architecture does not mean that everything is done
  - Just means everything has a place
- Many things could be developed a lot better and farther
  - CSV import/export, exec services, jdbc services, file datasource, formula functions for physics, …
- Always looking for people to take over pieces

Gathering requirement

- Used to talk to multiple people before implementing anything. This has become impossible at this point.
- This is the current process for a new spec:
  - Whenever I need to make a substantial choice, I open a github issue with a proposal
  - Wait for feedback on that proposal
  - Follow up whenever I am implementing something
  - Close even if not enough review
- For changing spec:
  - Open issue on github
  - All sites need to sign off change if breaks compatibility