Practical API-Development with Gemstone/S

Marten Feldtmann

dimap - das Institut für Markt- und Politikforschung, Hamburg + Bonn
infratest-dimap, Berlin
GESS Software mbH, Hamburg

ESUG 2018

About

- Marten Feldtmann
  - Smalltalker since VisualWorks/Atari-ST
- GESS Software, Hamburg
  - http://www.gessgroup.de/
  - **CATI Software**
- dimap, Hamburg and infratest dimap, Berlin
  - http://www.dimap.de, http://www.infratest-dimap.de/
  - Election Results Presentation for Television, HbbTV and Internet (ARD, SWR, WDR, NDR, RB, RBB, BR, ...)

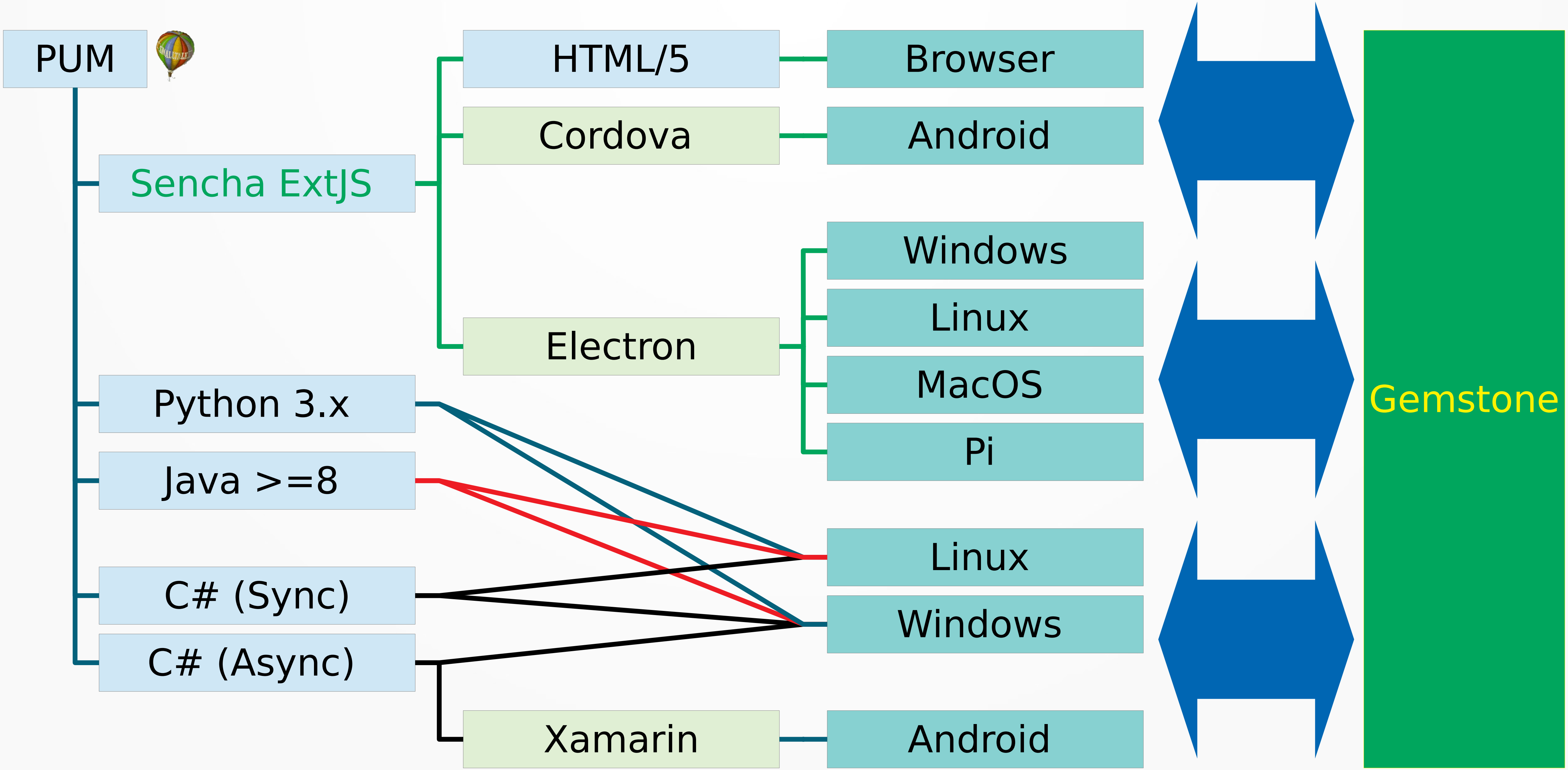
Project/Circumstances

- Left Smalltalk development around 2002
  - VisualWorks with Argos/Versant (OODBMS-Experience)
- General good experiences
- Out of Smalltalk business over night (Versant)

How to get to current state

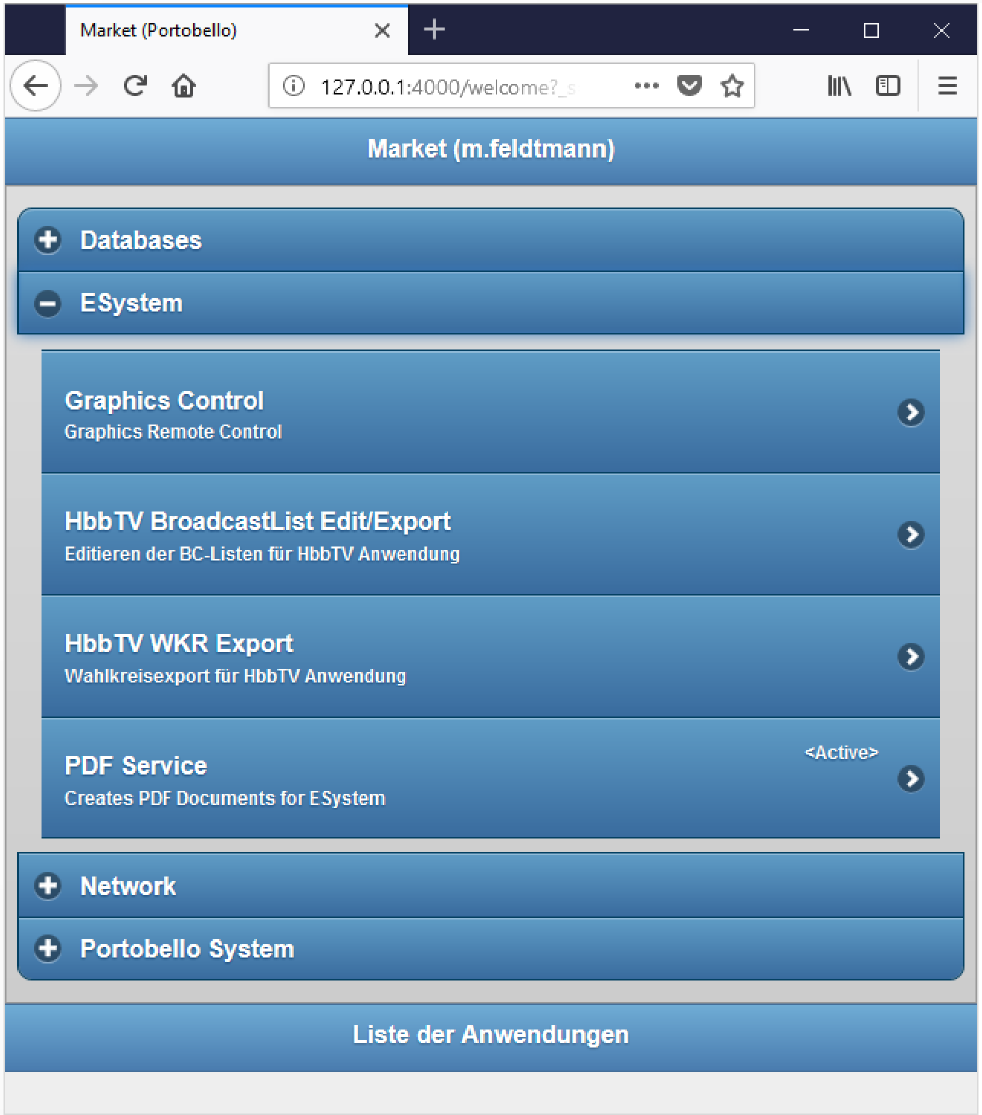
- Current State
- Evolution of a way to bring back Smalltalk
  - Early Attempts to bring back Smalltalk
  - Object-Modelling/DB-Layout/Code Generation
  - Programming Language Support
  - Runtime
  - Infrastructure, Scheduling
- No brilliant ideas – just work over years with an idea in mind.

Summary - Platform Overview



First attempt

- VA Smalltalk (2012/3)
  - Seaside + JQueryMobile
    - headless task on each computer
    - **0MQ** networking client
      - service detection
    - installation practise
    - ICU Unicode experience
    - IC packaging
    - Modular system
    - CouchDB
    - PDF service



Graphics Control
Graphics Remote Control

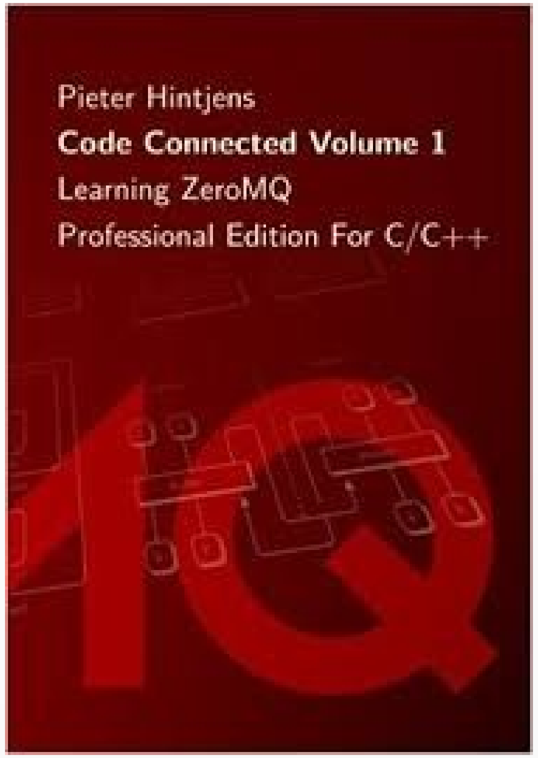
HbbTV BroadcastList Edit/Export
Editieren der BC-Listen für HbbTV Anwendung

HbbTV WKR Export
Wahlkreisexport für HbbTV Anwendung

PDF Service <Active>
Creates PDF Documents for ESystem

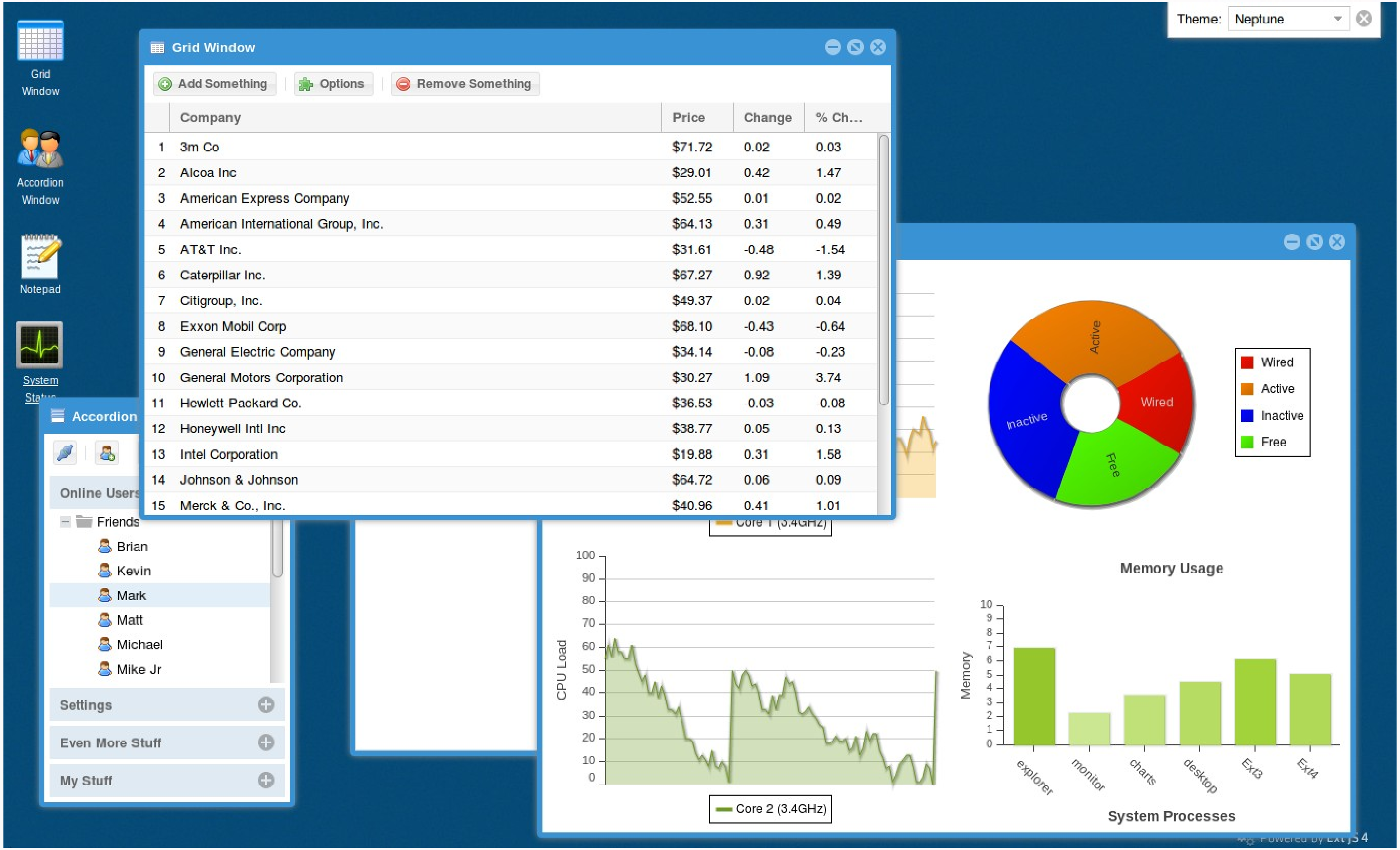
Tribute to Pieter Hintjes

- Peter Hintjens
- 1962 - 2016
- Network Computing using 0MQ
- Thank you !



Pieter Hintjens
Code Connected Volume 1
Learning ZeroMQ
Professional Edition For C/C++

Second attempt



	Company	Price	Change	% Ch...
1	3m Co	$71.72	0.02	0.03
2	Alcoa Inc	$29.01	0.42	1.47
3	American Express Company	$52.55	0.01	0.02
4	American International Group, Inc.	$64.13	0.31	0.49
5	AT&T Inc.	$31.61	-0.48	-1.54
6	Caterpillar Inc.	$67.27	0.92	1.39
7	Citigroup, Inc.	$49.37	0.02	0.04
8	Exxon Mobil Corp	$68.10	-0.43	-0.64
9	General Electric Company	$34.14	-0.08	-0.23
10	General Motors Corporation	$30.27	1.09	3.74
11	Hewlett-Packard Co.	$36.53	-0.03	-0.08
12	Honeywell Intl Inc	$38.77	0.05	0.13
13	Intel Corporation	$19.88	0.31	1.58
14	Johnson & Johnson	$64.72	0.06	0.09
15	Merck & Co., Inc.	$40.96	0.41	1.01

Demo
VA
+
ExtJS
:-)

Result

JS
could
be
the
way

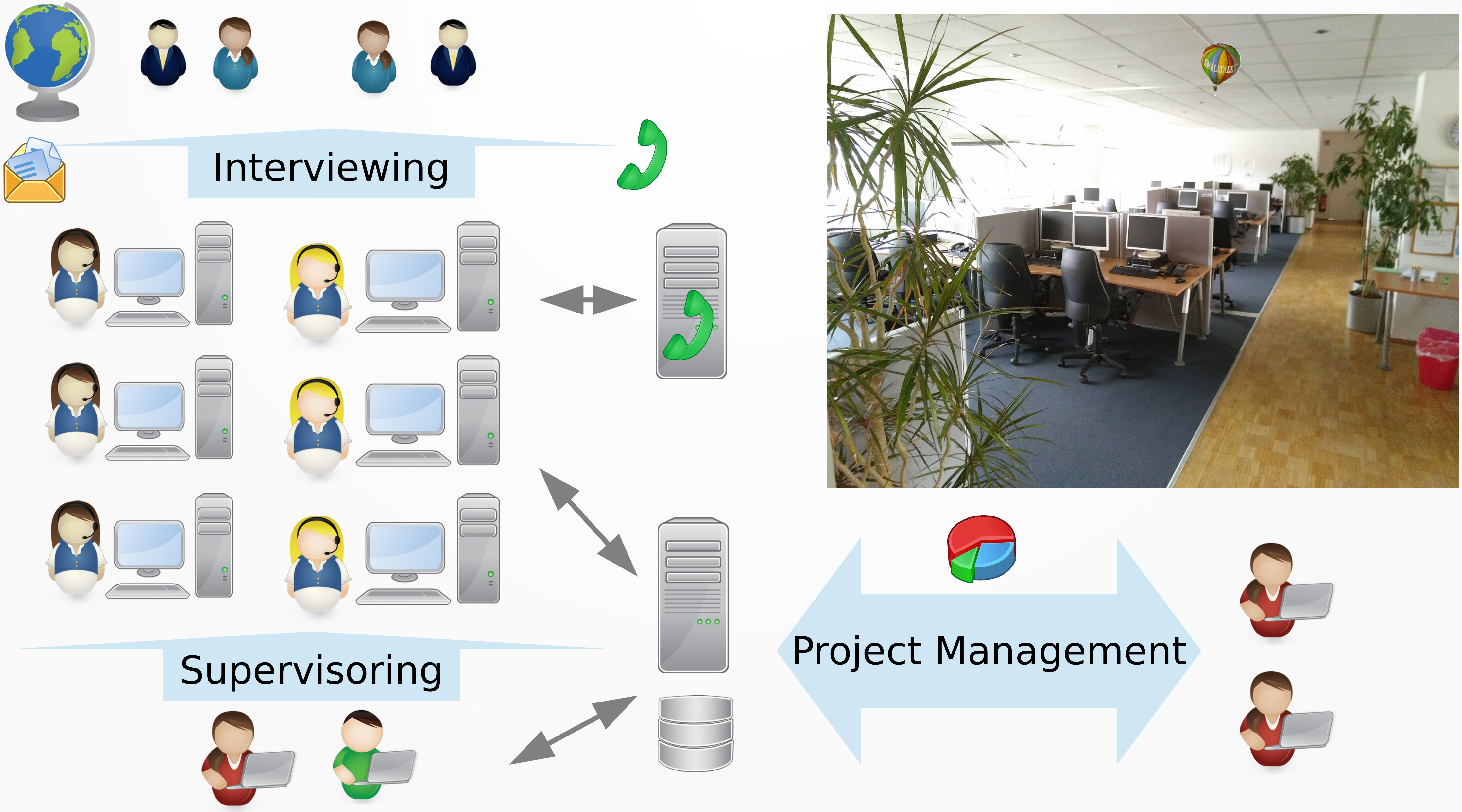
In the other company

- Other Company: *dimap*
  - Project for HbbTV result presentation for an election in Germany
    - Gemstone/S (3.1.0.x)
    - Sencha ExtJS (why not jQuery ?)
  - Manual Coding ALL stuf
    - HTTP-Request
    - Took long time, but result were ok

Product at the horizon

- CEO asked for help on new CATI System
  - First attempt failed completely
    - Lost on the RDBMS database side
    - No feeling for database
  - „Rewrite from scratch“
  - Look for suitable technology / system

CATI - System



Product Goals - CATI

- Initial target (reached)
  - 100 interviewers, 10 supervisors, 10 project managers
- Next target
  - 400 interviewers, 50 supervisors, 10 project managers
- Very different user profiles
  - Interviewer – calling people, execute the interview
  - Supervisor – managing interviewer, daily project work, statistics
  - Project Managers – overall project work, statistics

New Attempt

- Database ?
- Gemstone/S
  - This was not a decision based on facts
  - If you start such a project – have fun
- Reaction
  - Smalltalk ?
  - License Question ?
- Conditions
  - No pure Smalltalk application
  - Fallback considerations in case Gemstone/S is failing
  - Consider developers in company

New Attempt

- Overall Layout
  - Central Database is Gemstone/S (free)
  - System is running under Linux only
  - API-based (if a central component is failing)
  - UI is done in Javascript
    - Sencha ExtJS
  - UI talks with database via HTTP
  - Scripting language: Ruby (Hype !)

Initial Technical Goal

- Database Gemstone/S (starting with 3.1.0.5)
- **Two-Core** license for about $1500
- Code generation (as in our C# projects)
- Optimal support for UI (Sencha ExtJS Javascript)
  - Ajax-Calls (Single API Calls)
  - Store (CRUD – up to four API calls)
  - Class-based Javascript development
  - Paging
  - Sorting,
  - Filtering
  - Reminds me of the VASmalltalk Abt-Builder stuff, but at current state of the art UI
- => True seperated programming languages development

Modelling and DB-Layout

- Modelling / Generator Experiences
  - Own modeller („PUM“ – Poor Users Modeller)
    - Technical oriented modelling tool
  - Initially used for C#, started around 2003
  - Written in VASmalltalk
  - Very good experiences
  - Very special programming model for us and results were more than ok
  - Original C# model was „in-ram,distributed“ database oriented
  - Now the programming model has to be changed:
    - Need for a special Gemstone/S model

Programming Model – Decisions

- Defining Programming Model for Gemstone/S
  - Optimistic Locking, Revision Attribute if needed
  - Initially supporting Swagger (later OpenAPI) – but now obsolete
  - Initial REST oriented: PUT, DELETE, GET and POST
  - Changed to POST only
  - UTF8 only
  - JSON only - **one** JSON parameter allowed in each call
  - Two kind of API calls
    - Simple API-calls
    - CRUD structure support (4 API calls each CRUD)
  - Depends only on NeoJSON, Zinc* stuff and base Gemstone/S

Programming Model – Decisions

- Data Type Definition in JSON
  - Time (HH:MM:SS)
  - Date (YYYY-MM-DD)
  - DateTime (YYYY-MM-DDTHH:MM:SS.sss+xxxx)
  - Enumeration (as strings)
  - Flags (like Enums with „|“ as delimiter)

Modelling – Decisions

- Two Class Hierarchies (in DB)
  - Domain oriented: persistent classes
  - API oriented: transient classes
  - Domain class may have several „corresponding“ api classes
  - API class may have one „corresponding“ domain class
- Developer‘s world
  - Gemstone/S - uses both hierarchies
  - Client-developer – sees API classes only

Modelling – Attributes

- Base Types (int, string, ..)
- Array of base types (not association)
- „RcCounter“ support
- Calculated attributes
  - Either Smalltalk block defined in Tool
  - Calling to-be-implemented method

Modelling – Features

- „Access-Path“
  - Defines attributes based on access paths from a domain object to an attribute of another object
  - Very often used
    - Gemstone/S - „official“ way to access a far away attribute.
    - Enriches API classes to return more data than the base domain objet
    - Circumvents API classes limitations
      - API classes knows only their shadowing class attributes

Modelling – Associations

- Implemented as
  - Bag, Dictionary, Array, Queue, Set, Sorted, Indexed
- Additional hints
  - Conflict, Identity, Large, Unique
- Resulting classes
  - Array, IdentitSet, Bag, StringKeyDictionary, RcQueue, SortedCollection ….
  - Index support

Modelling – Associations

	Dictionary	Indexed	Ordered	Queue	Set	Sorted
conflict	Dictionary StringKeyDict. IntegerKeyDct.	Set	Array	RcQueue	Set	Sorted-Collection
identity	Dictionary StringKeyDict. IntegerKeyDct.	IdentitySet	Array	RcQueue	IdentitySet	Sorted-Collection
large	Dictionary StringKeyDict. IntegerKeyDct.	Set	Array	RcQueue	Set	Sorted-Collection RcIdentitySet IdentitySet *Bag
unique	Dictionary StringKeyDict. IntegerKeyDct.	Set	Array	RcQueue	Set	Sorted-Collection
none	Dictionary StringKeyDict. IntegerKeyDct.	Set	Array	RcQueue	Set	Sorted Collection

Gemstone – Query Facilities

- Regardless of database type – queries are a MUST
  - Gemstone/S possibilities (Hmm...)
    - Query Language
    - Filtering
    - Sorting
    - Multi-Index Searching
    - Offset, limit query etc ...
    - SELECT * FROM TABLE WHERE x=? offset x limit 25
  - Speed, Speed (e.g. Buffered-Tables in UI)
    - Presorted Structures ?

Gemstone/S – PUM Runtime

- Support
  - Optimistic Locking
  - Request Retries (up to 6 times)
    - Reduced errors at the client
    - Increasing concurrency
    - Increasing database load
  - JSON serialization of parameter (API Classes)
  - Application user session handling
  - Logging, Benchmarking
  - Support for 0MQ-Events on successful/failure transactions

PUM - Languages

- Creates topaz-script (now: 12MB) with all source code
  - classes, association handling, migration support
  - Index support
  - Setup persistent domain tree, API calls
  - Object initialization support
  - Copying API Classes ↔ DomainClasses
  - Placeholder for to-be-implemented methods

PUM – Languages - Javascript

- Creates Javascript files for ExtJS
  - Single Files for classes and stores
  - Merging process of newly created files with extensions in older files

PUM – Languages - Python

- Initial scripting language candidate: Ruby
- Change to: Python
  - Create Language Runtime based on the initial Swagger adaption of Swagger-Runtime
  - Simulators (Raspi Rack), service-scripts, background-tasks, API examples – all is written in Python
  - PUM creates ALL code in ONE large file. Easy to copy to a project.
  - Very good choice

PUM – Languages – Java, C#

- General
  - Either language is used in company, must connect to product or to place a solution within company
- Java
  - PUM Runtime written from scratch
    - External JSON library (Google)
  - Delivered as jar files to API-developers
- C# (Xamarin and/or .Net/Mono (Unity3D))
  - PUM Runtime written from scratch
  - External JSON library (Newtonsoft)
  - Two generators (synchronous, asynchronous model)
  - Delivered as Solution or dll to API-developers
  - Base of documentation generation (MS Sandcastle + Print & Manual)

PUM – Examples

C#:

```
var anApiClient = new ApiClient( jsonHandler, "http://compi.mydomain.de");

var aCATIAPISessionCreationInfo = new CATIAPISessionCreationInfo();
aCATIAPISessionCreationInfo.login = "root";
aCATIAPISessionCreationInfo.password = "rootpassword";
aCATIAPISessionCreationInfo.customerName = "gess";

var aCATIAPISessionResultList = anApiClient.CreateSession(aCATIAPISessionCreationInfo);
```

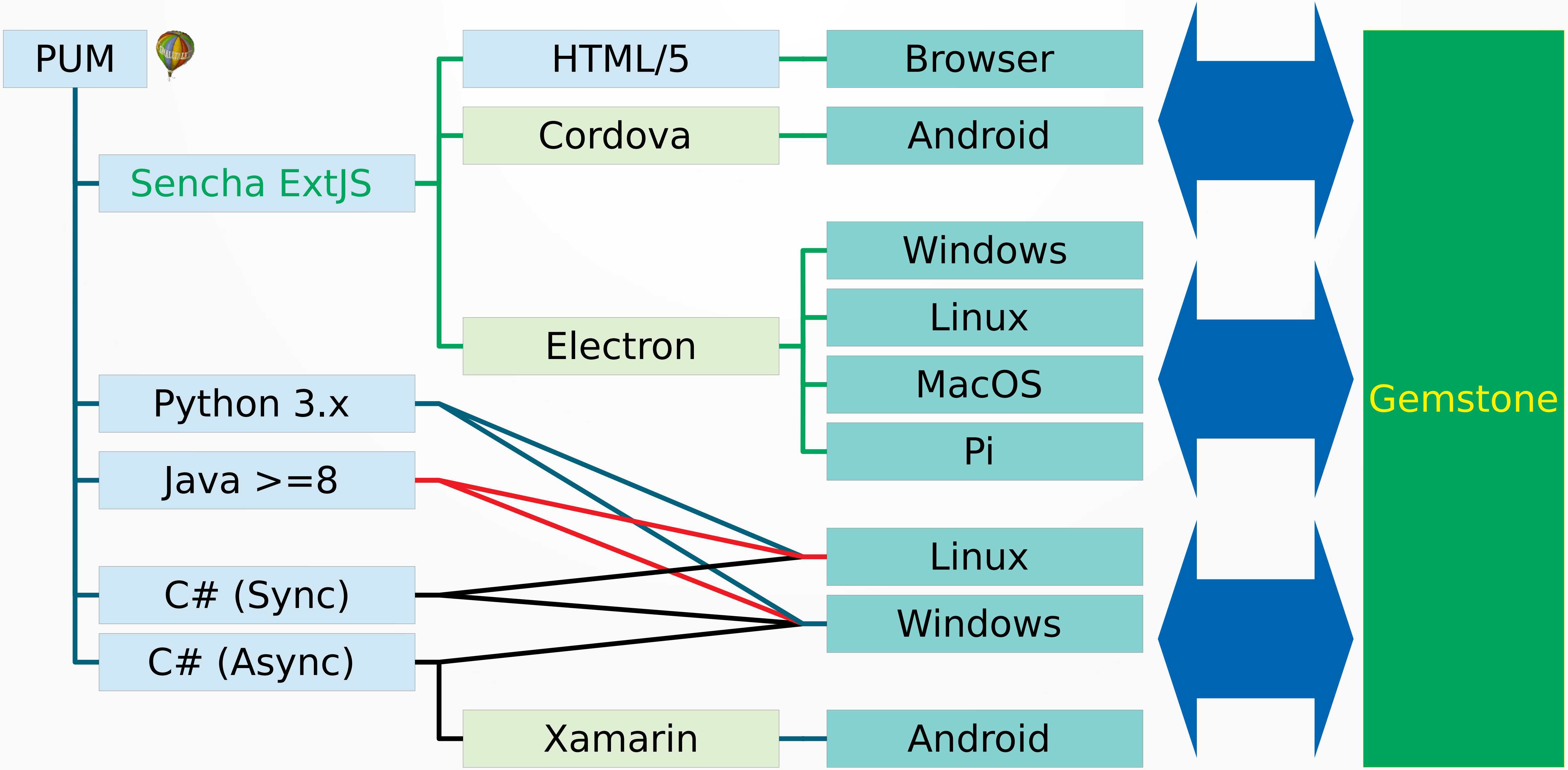
Python:

```
aClient = swagger.ApiClient('', "http://compi.mydomain.de")
api = webcatiapi.RESTApi(aClient)

aCATIAPISessionCreationInfo = webcatiapi.CATIAPISessionCreationInfo()

aCATIAPISessionCreationInfo.login = „root“
aCATIAPISessionCreationInfo.password = „“rootpassword“
aCATIAPISessionCreationInfo.customerName = „gess“

aCATIAPISessionResultList = api.CreateSession( aCATIAPISessionCreationInfo)
```

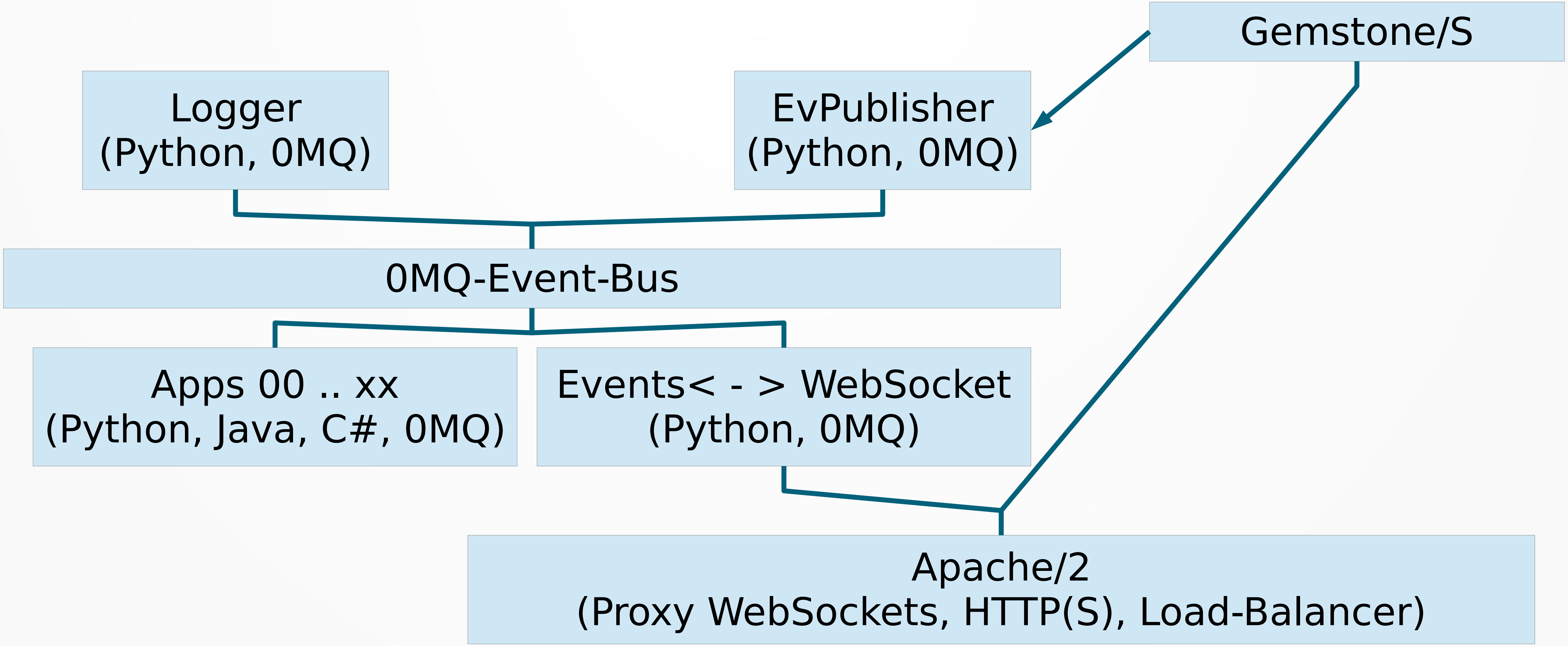
From passive to active

- Introduction of 0MQ intro product
  - Backend developer defines events for special domain situations in the model
  - The backend developer places events in case of commits and failures.
  - External application are listening to these events
    - No polling, language independent
  - Incredible experience and improvement of the system layout

Summary - Platform Overview



System Layout - Backend



Summary - Overall

- Summary
  - People get used to true multi-language development (MLD)
  - Long – not easiest, lots of wrong decisions - path of evolution
  - Great programming model on the backend due to Gemstone/S
  - Smalltalk used in company
  - Introduce Python as an accepted language (:-( )
  - Active database due to
    - 0MQ-based Event structure (MLD)
    - WebSockets (MLD)

Summary - License Costs/User/Year

- €1050 (Microsoft, Windows)
  - VisualStudio Pro. (C#)
- €650 (JetBrains, Windows + Linux + Mac)
  - WebStorm (Javascript), PyCharm (Python) and IntelliJ (Java)
- €1800 – Sencha ExtJS Framework (HTML)
- €500 – Sencha Test Framework
- €1500 – Gemstone/S, 2-core license (Linux)
- €500 - Help and Manual (Documentation)
- Total: ~€4500 + (€1500 - €6000) for database (web licenses)
- Too much ?

Thank you !

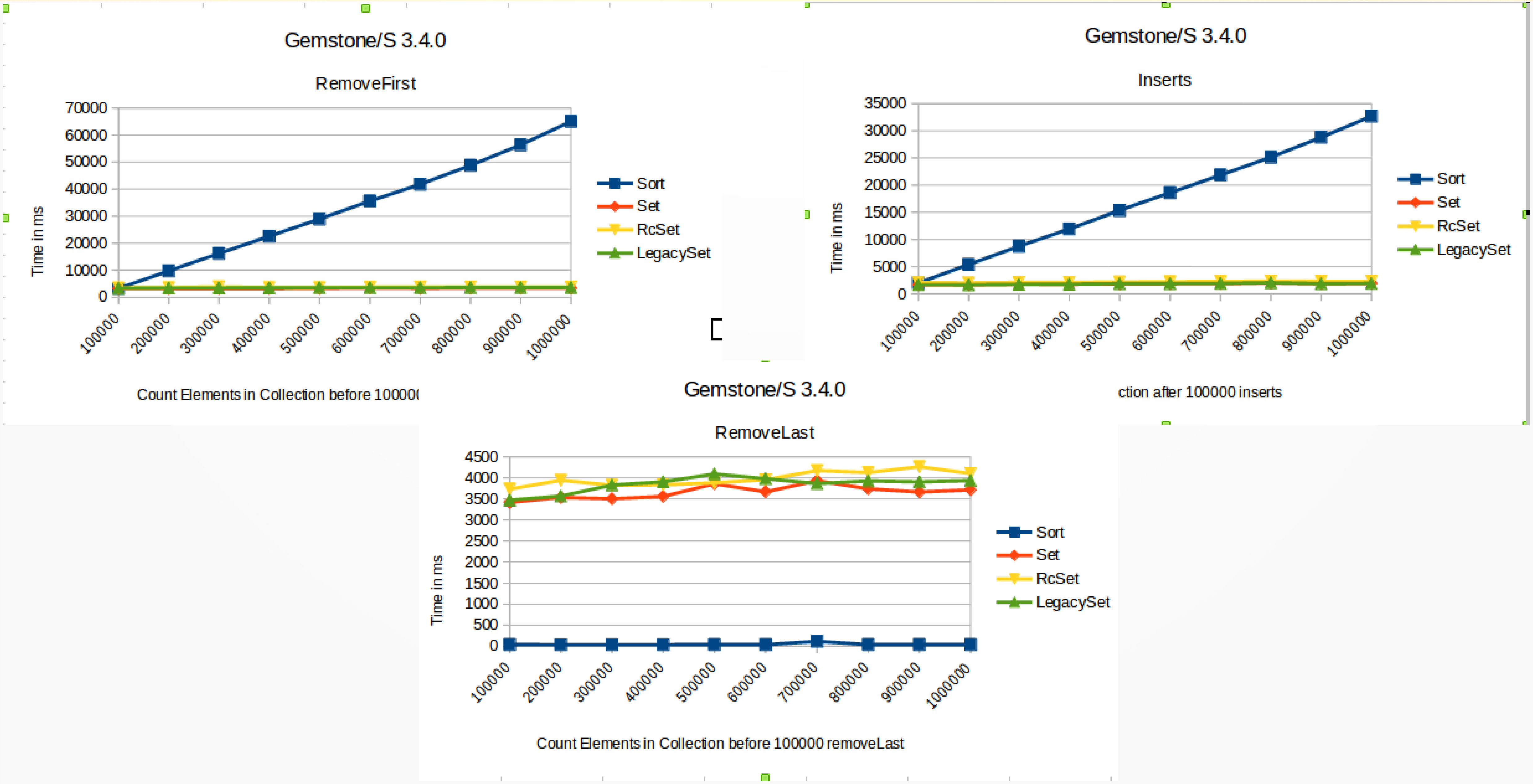
Load Balancing

- API-Call
  - Try to group api calls in different pools of similiar time-behaviour
  - Longer API-call may block short calls
  - Adding more Gemstone/S answering tasks is not always a solution
  - 2-core license is a limitation here
- Apache/2, different algorithms for load balancing
  - Bybusiness – not default

Gemstone/S – SortedCollection

- Workhorse
  - Speed relates with size
  - Enables presorted structure (Speed access !)
  - Very fast operations possible
    - Browsing, Buffered Browsing
  - Consider reverse logic (work at the end of the collection)
- Alternative: IdentitySet with index support
  - Slower
  - Constant speed

Gemstone/S – SortedCollection



Gemstone/S

- Interest in Gemstone/S increased ? No. Why ?
  - Offers no support for developers in other language
  - Offers no basic programming model or database API
  - Consider Smalltalk as a datascripting language
  - Everything has to be built from scratch
  - Others choose PostgreSQL, SQLite, MySQL or MariaDB or MongoDB
    - Usable from all other languages out of the box
    - User need mostly basic stuff from database: queries, storing etc …
  - No Windows platform for development

Modelling – Features

- „Search-Path“
  - Define search paths from the root to a persistent object
  - e.g. Address
    - Customer (customerID)
    - Project (projectID)
    - AddressContainer (containerID)
    - Address (addressID)
  - Pretty boring for API-developer
- Switching to „OOP“-Number under Gemstone/S