Writing Applications with Strakt's CAPS

Alex Martelli

This talk's audience...:

- "fair" to "excellent" grasp of Python and OO development
- "fair" to "excellent" grasp of business application development and deployment
- wants to learn more about: Python for business applications, CAPS

This talk doesn't cover...:

- **all** the details...!

→ Come ask at Strakt's booth

Neither does it cover...:

- Strakt's business-model

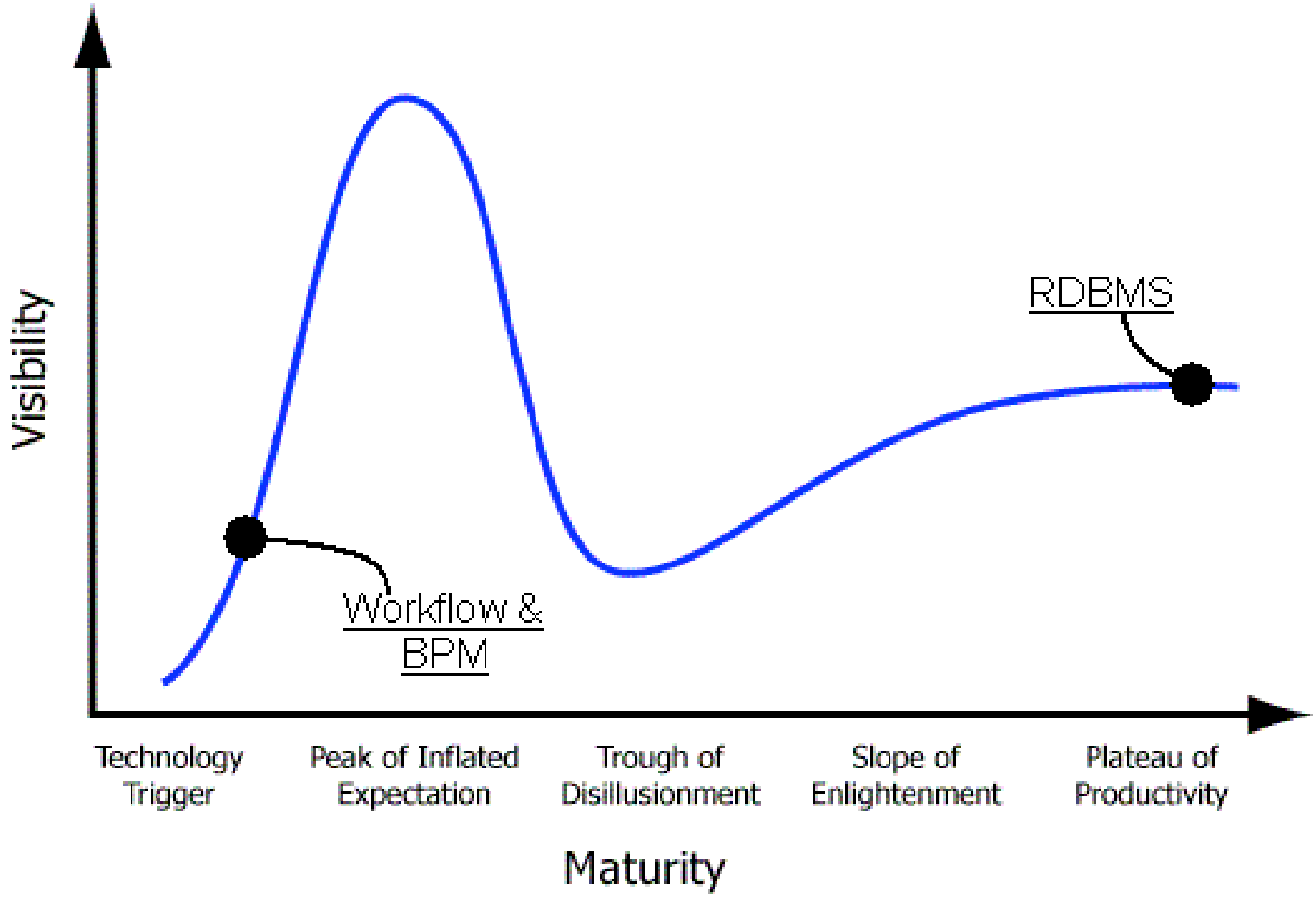
→see **www.strakt.com**

→ask at Strakt's booth

CAPS

- Collaborative Approach to Problem Solving
- focuses on **real-time** business applications characterized by **collaboration** among knowledge workers and the handling of **workflow** issues
  - management of projects and/or resources
  - relationships with suppliers, customers, employees
  - help-desk case-tracking and handling
  - coordination of R & D activities

A warning about "workflow"...



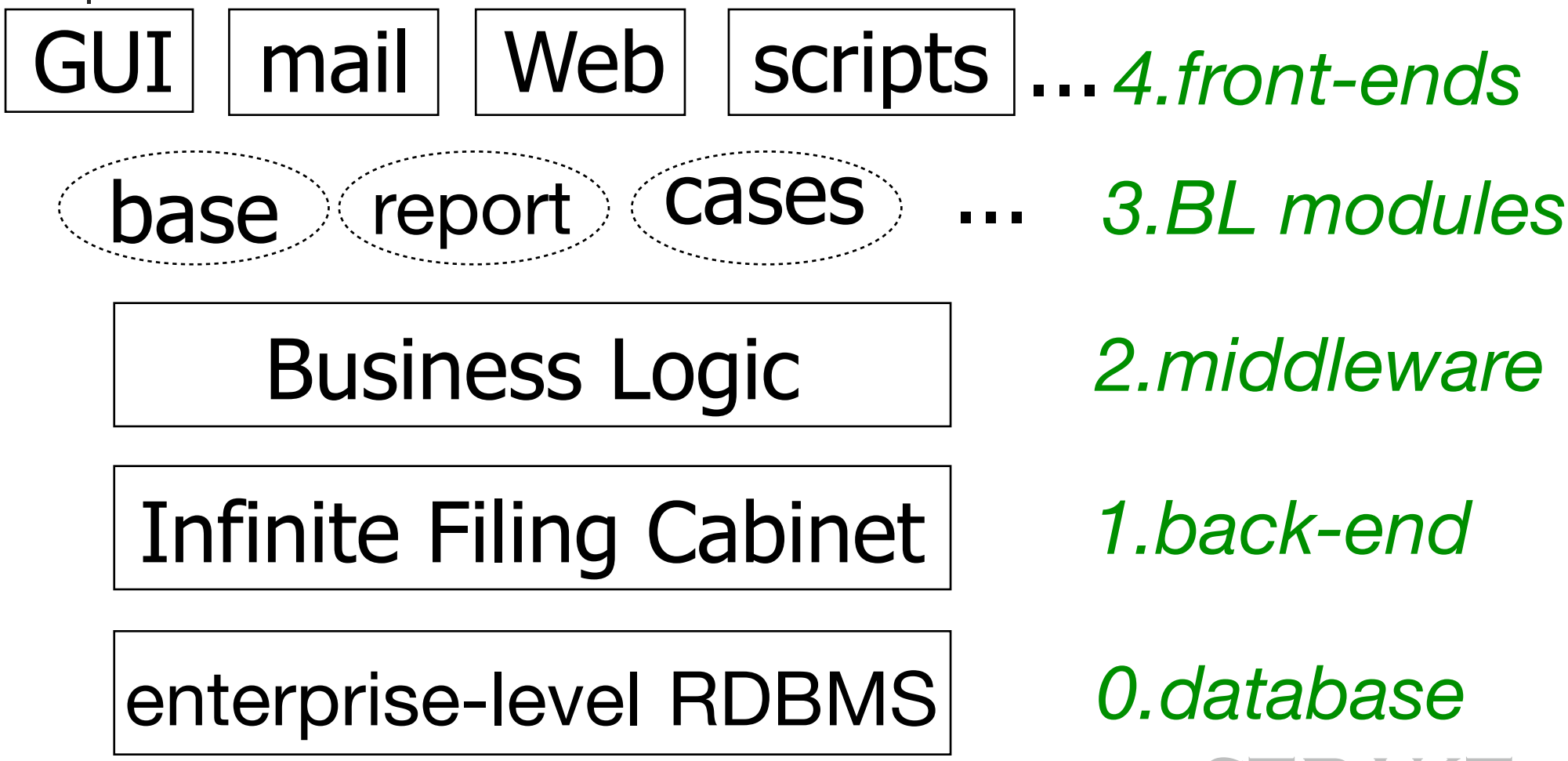
Application-frameworks

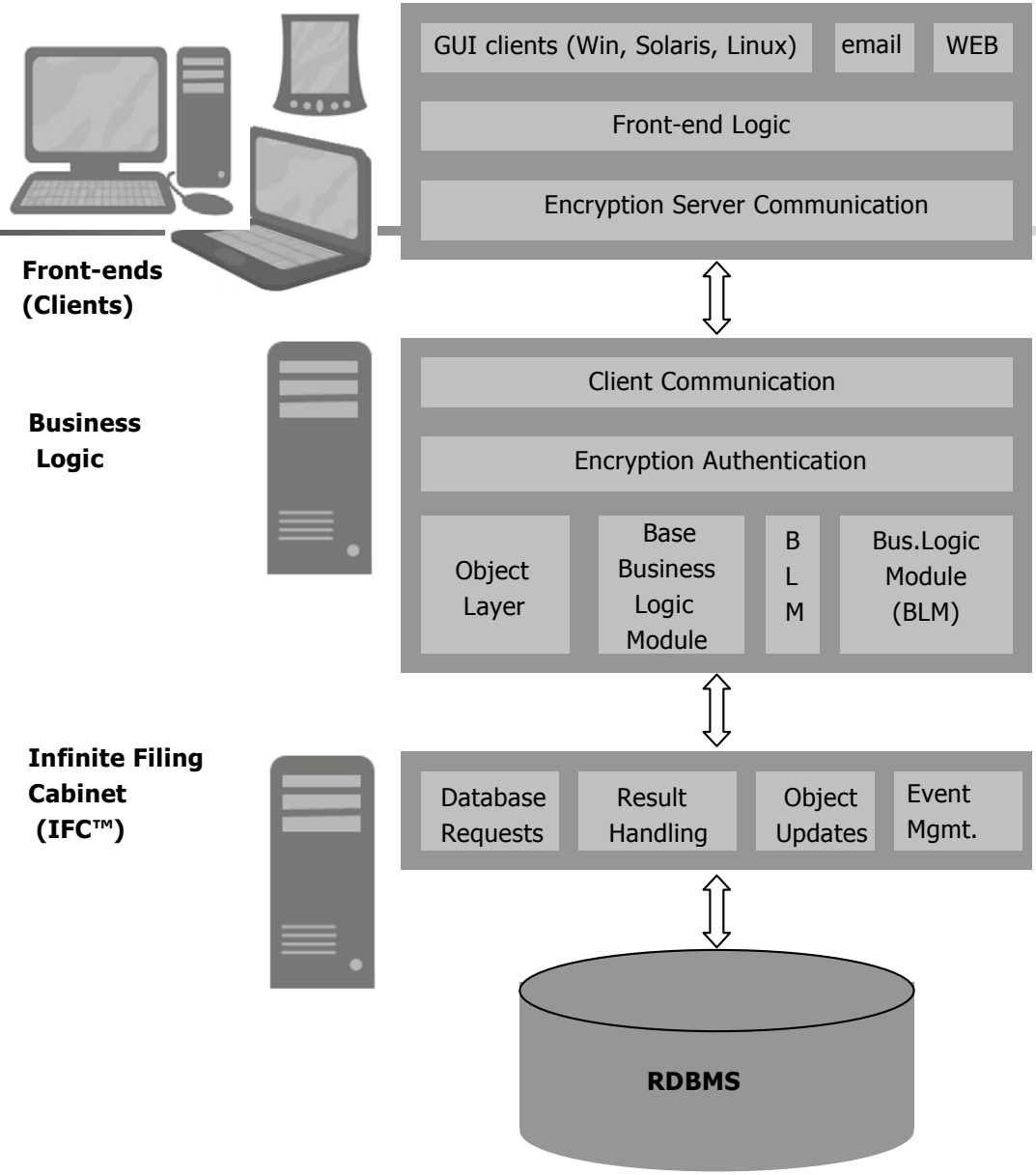
- like a **library**, a **framework** includes substantial amounts of reusable code
- a framework also <u>structures</u> the applications that you develop with it:
  - embodies architectural and design expertise
  - may be oriented to specific application areas
  - may include special-purpose mini-languages, code generation, integration with other technologies, and other programming-in-the-large architecture-patterns

CAPS' strengths: mile-high view

- centers on collaboration through the sharing of **information** (not just **data**)
- real-time updates, event-driven (threads and processes used "behind the scenes", seamlessly, when and where needed)
- multiple front-ends (GUI, scripts, web, mail, …) operate on a shared information-base
- N-tier flexibility eases integration, scaling, deployment

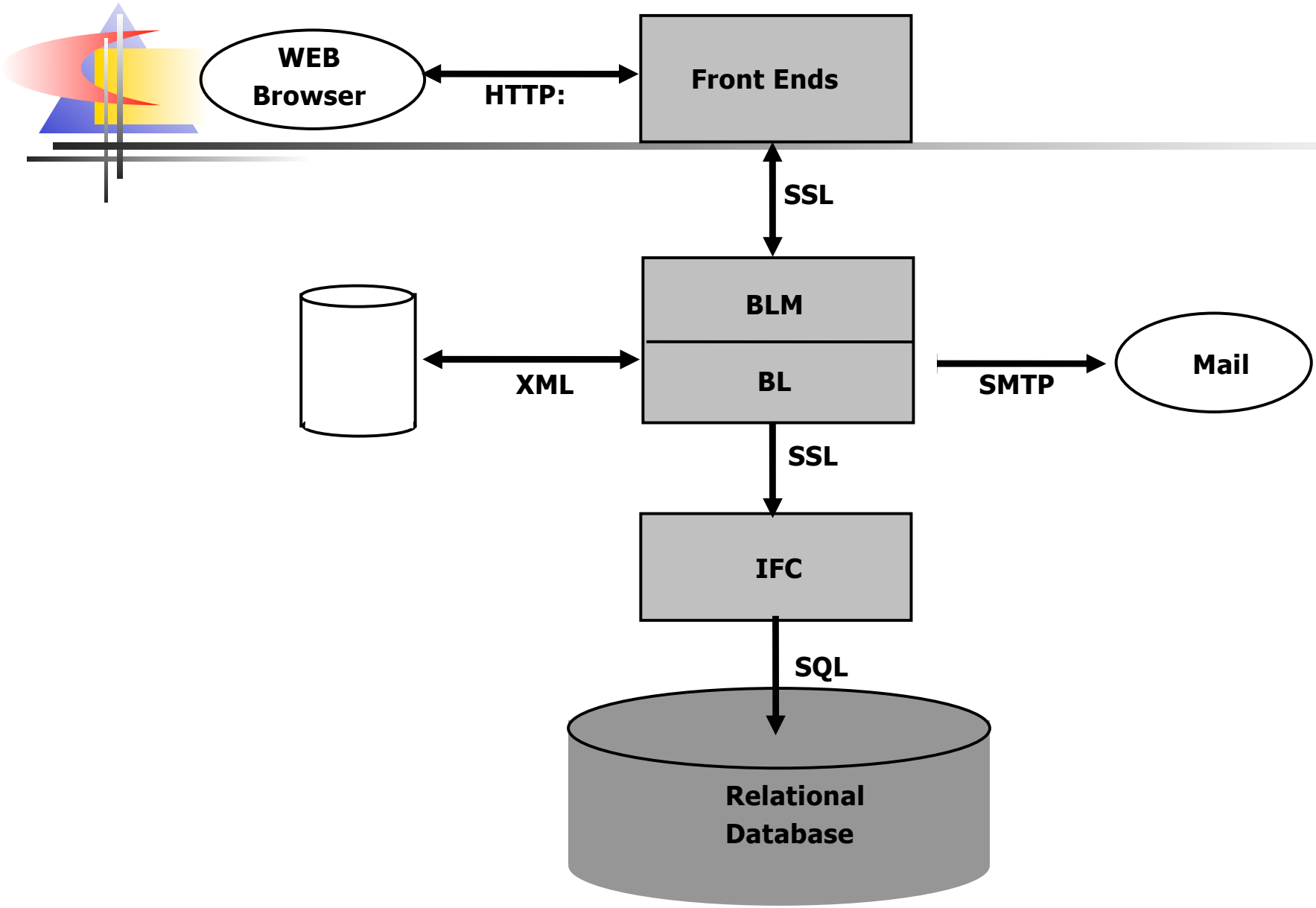
CAPS layers


GUI
mail
Web
scripts
...
4.front-ends
base
report
cases
...
3.BL modules
Business Logic
2.middleware
Infinite Filing Cabinet
1.back-end
enterprise-level RDBMS
0.database


GUI clients (Win, Solaris, Linux)
email
WEB
Front-end Logic
Encryption Server Communication
Front-ends
(Clients)
Business
Logic
Client Communication
Encryption Authentication
Object
Layer
Base
Business
Logic
Module
B
L
M
Bus.Logic
Module
(BLM)
Infinite Filing
Cabinet
(IFC™)
Database
Requests
Result
Handling
Object
Updates
Event
Mgmt.
RDBMS

Communication between layers

- BL and BLMs are only logically separate: the BL loads the BLMs (from the IFC) into the same process (so, 2↔3 is "intrinsic")
- 0↔1: SQL via... *(depends on the RDBMS)*
- 1↔2, 2↔4, 3↔4: SSL (secure, distributed)
- **no other** communication between layers
  - though each layer may further communicate with external systems (mostly, parts of layer 4 do), e.g. w/ SMTP, HTTP, XML, ...

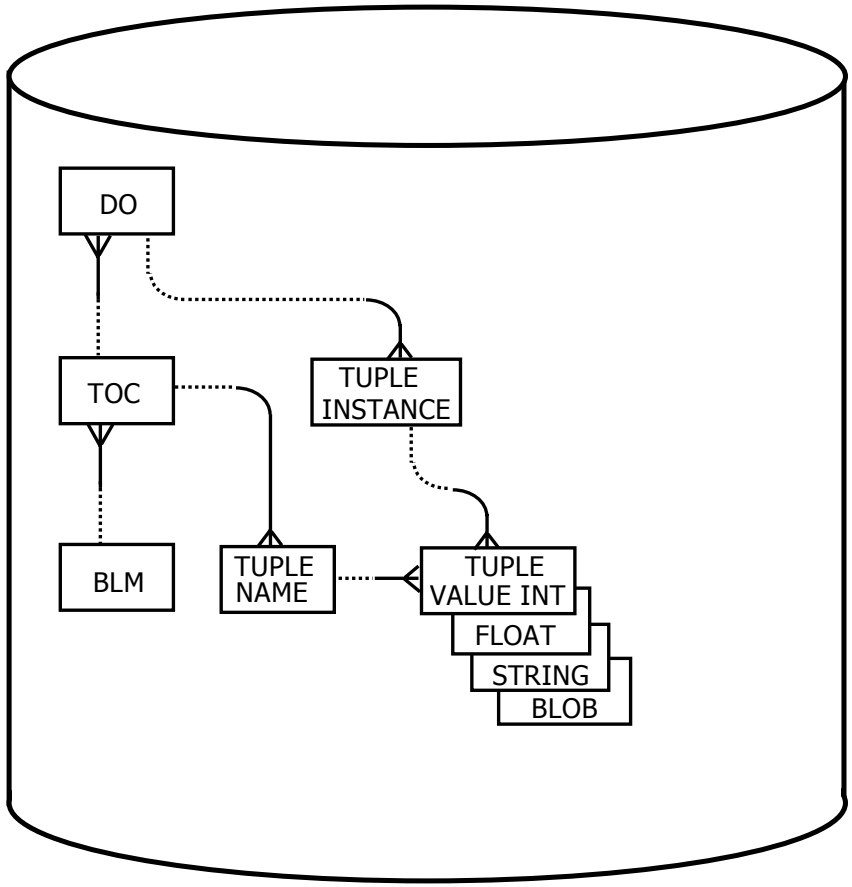

WEB
Browser
HTTP:
Front Ends
SSL
BLM
BL
XML
SMTP
Mail
SSL
IFC
SQL
Relational
Database

RDBMSs for CAPS applications

- PostgreSQL
- Oracle
- Microsoft SQL Server
- ...any enterprise-level (full-function) RDBMS, i.e one w/all standard features:
  - transactions
  - views
  - outer joins
  - ...

CAPS apps in the RDBMS

- CAPS uses **its own schema**…:



Infinite Filing Cabinet (IFC)

- real-time, transactional **OODB**
- built on top of the level-0 RDBMS
- "infinite" because by design the IFC does not remove "old, superseded" information
  - **marks** deleted/updated info as "deprecated"
  - ensures perfect audit trail
  - supports future data-mining, reporting, ...
  - obliteration (e.g. for legal requirements... or very tightfisted, myopic customers:-) via other admin itf

IFC: Query vs Request

- "Query": give me the set of objects satisfying condition ... [[as numeric IDs]]
- "Request": for this set of objects [[as numeric IDs]] give me the values of these attributes ... [[lists of approp. datatypes]]
- each can be transient or subscription
  - upon subscription, IFC sends update packets (asynchronously, as needed) until canceled

IFC: Events

- "normal": send alert when certain specified attributes change...
  - ..."to" given values, or...
  - ..."away from" given values, or...
  - ...from given "entry" values to given "exit" values
- "timed": if certain attributes have specified values at a specified time
- "rule-based": when objects are created or modified and satisfy certain conditions

The Business Logic

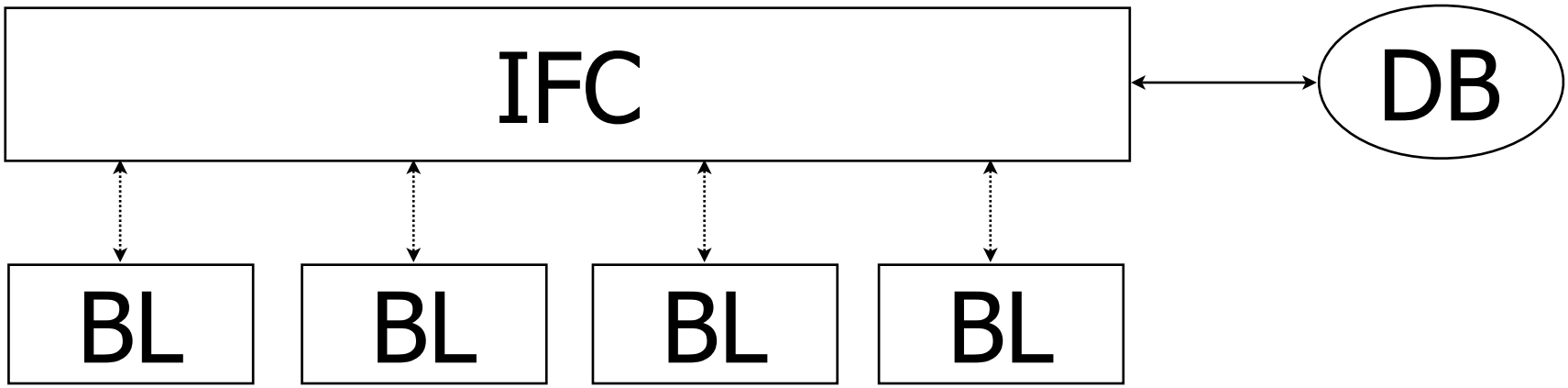
- the **BL** middleware subsystem builds full-fledged business objects on top of the IFC's fundamental OODB facilities
  - objects that respect "business rules" and have all the appropriate behavior
- the BL mediates all communication to and from all of the front-end subsystems
  - thus, front-ends always deal with business objects, never with the "raw" underlying ones

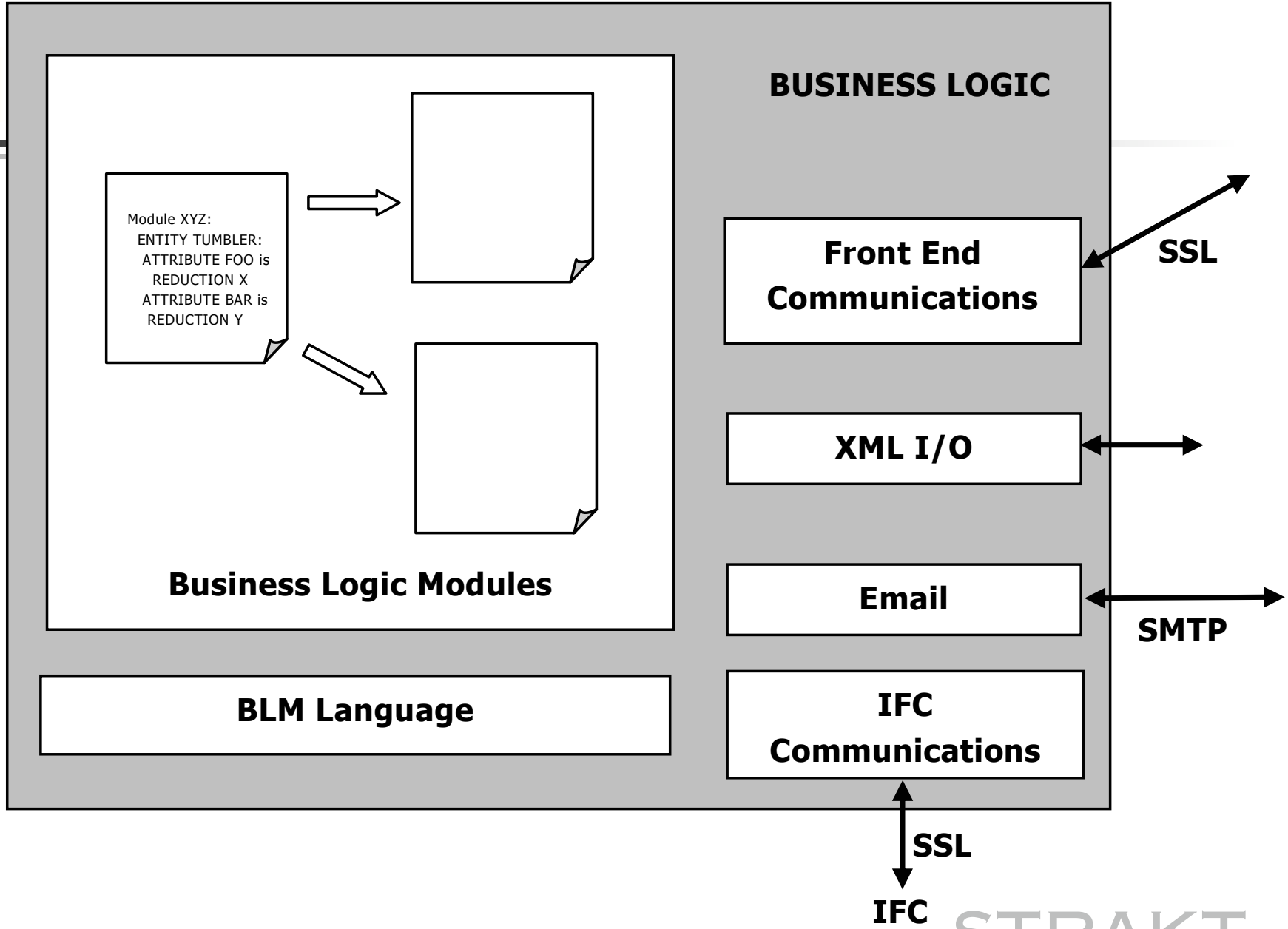
BL: access privileges

- the IFC trusts the BL (once properly connected and authenticated) and lets it perform all kinds of accesses and updates
- the BL (in the "base" module) implements "users" and "groups", and grants highly selective access privileges accordingly
- BL↔IFC needed trust level is thus high, while BL↔FE trust level can be quite low

1 IFC ↔ multiple BLs

- ...not currently supported, but...
- ...architecturally OK:
  - BL caches can be refreshed or invalidated upon a BL receiving subscription updates (from others' changes)




BUSINESS LOGIC
Module XYZ:
ENTITY TUMBLER:
ATTRIBUTE FOO is
REDUCTION X
ATTRIBUTE BAR is
REDUCTION Y
Business Logic Modules
BLM Language
Front End Communications
SSL
XML I/O
Email
SMTP
IFC Communications
SSL
IFC

Business Logic Modules

- BL loads from IFC and runs multiple **BLM**s
- the **base** BLM, always present, supplies:
  - infrastructure: TOI, session, settings, event, ...
  - security model: user, UG, AC, OPE/PAE, BLAUTH, ...
  - auxiliary: macro, search, query, mailqueue, ...
- **auxiliary** BLMs, plugin-like (reporting, ...)
- **application** BLMs: application information model, business rules, specific actions, ...

The BLM Language **blam**☺

- each BLM is coded in CAPS' own "**BLM Language**" (very **un**officially **blam**☺)
- declarative, corresponds to Entity-Relation analysis of the information-model
- a rather Python-like syntax:
  - significant indentation, uncluttered
  - comments, docstrings, a few parentheses
  - its own set of keywords
- embeds Python code for procedural parts

BLM Language concepts

- **toc** ("class") and **toi** ("instance") ↔ **Entity**
  - tocs form a single-inheritance tree
- **type**: string, number, ...; can have further
  - **restrictions** (on quantity, changeability, ...)
- tois (and whole BLMs) can have:
  - **methods**, may have parameters and results
  - **attributes**, stored or computed, may have **on** clauses, restrictions **pre**, **post**, **both**, default values, ...
  - **relations** to/from other tois
  - **on** clauses (create/update of the whole toi)

blam ☺ blm, import, type

```
""" module docstring """
blm name version "string"
    some Python code here
%%
import a_blm_named_foo as bar

type nameString is LimitedString
    """ docstring for the type """
    restriction quantity(1)
```

blam☺ code, attribute

```
code
    import time
%%

attribute now is Timestamp
    "The current time"
    restriction quantity(1)

    computation code
        return [ time.time() ]
    %%
```

blam☺ method, restriction

```
method createFoo(bar) is ToiReference
    restriction quantity(1)
    restriction toiType(Foo)
    param bar is nameType
    code
        foo=services.new('Foo', b=bar)
        return [foo]
    %%
```

blam☺ toc, on-clauses

```
toc Foo(Bar)
    """ an example toc """
    attribute b is nameType
        on update code
            newinitial = value[0][0]
            existing = self.b[0][0]
            if newinitial != existing:
                raise BlmException('?!')
        %%
```

blam☺ relations

- relations are a first-class concept, w/ constraints (e.g. arity) as restrictions:

```
toc Tree
  relation parent to Tree.children
    restriction quantity(1)
  relation children from Tree.parent
```

- there are **also** `ToiReference` attributes, usable kinda like "2nd-class" relations

blam ☺ "2nd class relations"

```
toc Club
  attribute members is ToiReference
    restriction toiType(Person)
toc Person
  attribute clubs is ToiReference
    restriction toiType(Club)
    computation code
      return services.findObjects(Club,
                                  members=self)
    %%
```

blam☺ the 1st class equivalent

```
toc Person
  relation clubs from Club.members
  attribute name is nameType
toc Club
  relation members to Person.clubs
  on update code
    for p in value:
      if p.name == [u'Groucho']:
        raise BlmException('no way!')
  %%
```

blam☺, ACID, multiwhatever

- "real" transactions should always be:
  - Atomic
  - Consistent
  - Isolated
  - Durable
- for speed & scalability, CAPS is event-driven (with threads/processes behind the scenes)
- ...it might be challenging to code to this...

blam☺ might be challenging...

- ...so you don't have to!
- (we're not quite there, but closing in...):
- transparent ACID
  - "collision detection", automatic restarts, transactions
- (we do have already, kinda...):
- transparent multiwhatevering
  - code plainly and linearly (no locking/unlocking, no explicit "waits", callbacks, deferreds, ...)
  - sequencing & synchronizing are implicit

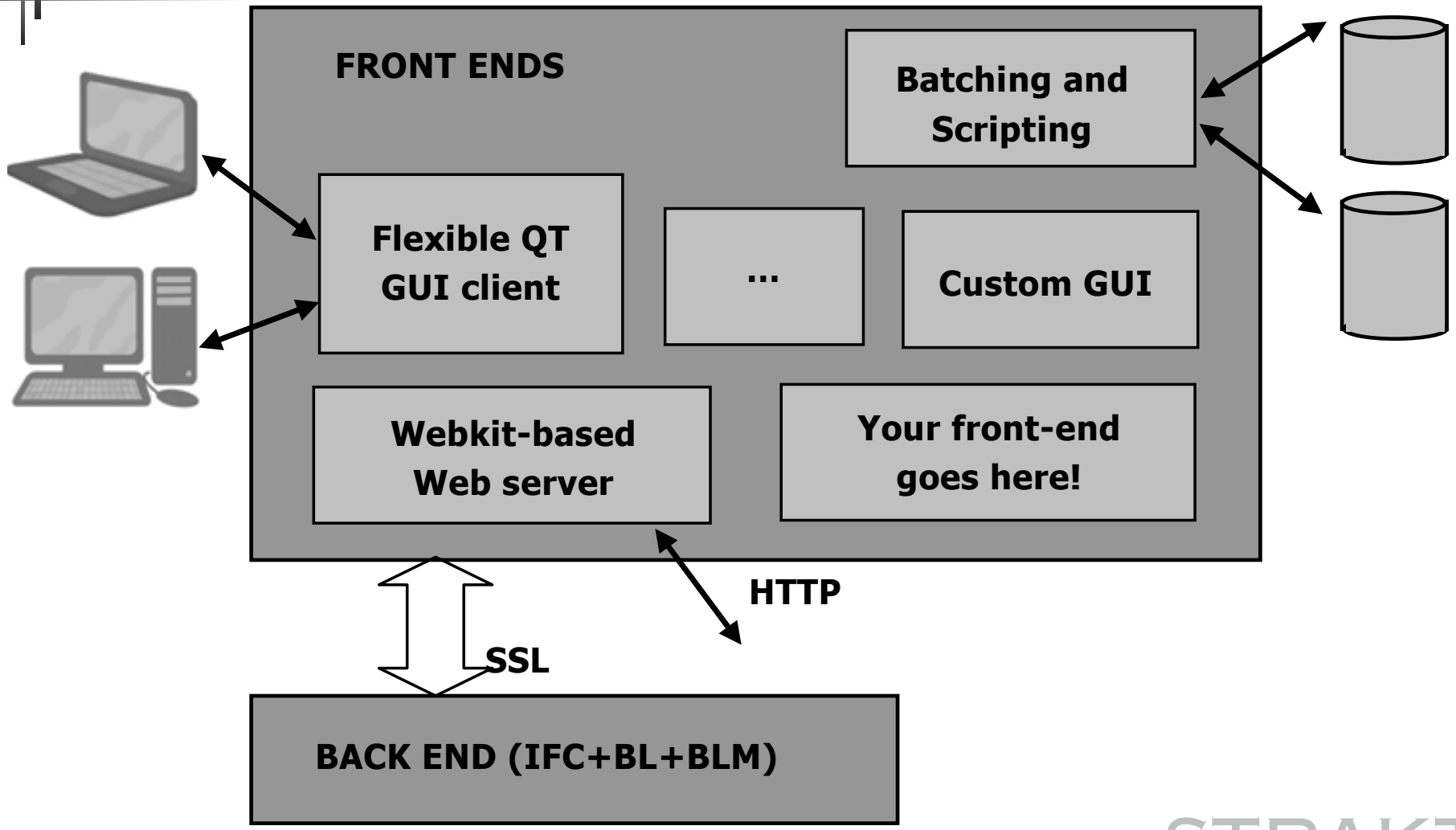
blam☺ & presentation issues

- presentation-driving metadata are set by separate **Presenter** sub-modules
- currently coded as plain Python classes
  - procedural, not declarative
  - mostly bunches of self.setThis, self.addThat calls
  - methods & constants supplied by Presenter baseclass
  - some Template-Method DP usage
  - currently only for GUI front-ends (e.g. web front-ends currently use on-disk HTML template files)
- ...clearly space to grow here

An example Presenter subclass

```
class SessionPresenter(MyPresenter):
    def _addTocPresentation(self, toc):
        self.setViewList((self.vwList, self.vwPick, self.vwSearch, self.vwFull))
        self.addListColumns(None, ('name', 'user_id', 'last_active'))

        self._addToFullView(None, 'name', self.stdFullEdit)
        self._addToFullView(None, 'user_id', _defaultToiRef(toc['user_id']))
        self._addToFullView(None, 'last_active', self.stdFullDateTime)

        self.addToSearchView(None, 'name')
        self.addToSearchView(None, 'user_id')
        self.addToSearchView(None, 'last_active')

        self.setPickView(('name', 'user_id', 'last_active'))

        self.addListViewFunctions(None, (self.vfCreate,self.vfDelete,self.vfFind,
                                         self.vfPrint,self.vfOpen))
        self.addFullViewFunctions(None, (self.vfSubm,self.vfRevert,self.vfEdit))
        self.addSearchViewFunctions(None, (self.vfSearch,))

        self.addToTocPresentation('defaultAttr', 'name')
        self.mkStdViewLists()
        self.mkStdViewInfo()
```

CAPS front-ends ("clients")



CAPS front-ends

- handle all user-interaction:
  - flexible Qt-based "generic" GUI client
  - it's feasible to write custom GUI clients
  - Twisted/Nevow - based web front-ends
  - it's feasible to use "curses" via Python scripts
- can handle many integration tasks:
  - via Python scripting
  - by exposing webservices (XML-RPC, SOAP, ...) to remote clients or (more typically) to other middleware
- many further yet-untapped possibilities

File Edit Searches Tools Settings Admin Help

New Print Edit Copy Delete New notifications Create macro Run macro Add event Add messa

Forward Reply Reply to all Reply to stakeholders

Search for:

230808

- 1. Release to be set
- caps-bugs
- corrected actions
- corrected cases
- created last 2 days
- created last day
- Created last week
- Finished last day
- Inactive bugs
- live bugs
- live maintenance
- Me = Handler
- Me = Responsible
- my actions
- Test
- Perform search

Open Search Template

corrected cases, 43 matches

Name	Last updated	Case number
Export to csv	26/05/2004 12:26:53	254579
Measuring performance	07/04/2004 14:12:27	127565
Message - listview	26/05/2004 10:53:30	254839
New Attributes and Presentation data	26/05/2004 14:47:56	269920
Office integration - Word	26/05/2004 12:22:34	257821
Release object	**26/05/2004 10:49:07**	**276012**
Reply all	26/05/2004 10:53:13	126446
Search result - 'My - email out' on user Per-Åke	**01/06/2004 14:33:16**	**276672**
Actions (1)		
Get list of non-working searches	**12/05/2004 17:28:28**	
Notes (1)		
Messages (0)		
Documents (0)		

Viewing Case 'External initiator - initial search value' (strakt_all)

Overview | Details | Sales | Development | History

Name External initiator - initial search value **Case number** 147355

Category Improvement request Show **Priority** Low

Comments

Comment	Created	Created by

CAPS' General GUI Client

- cross-platform Windows, Linux, Solaris, (Mac), ..., w/Qt+Python+PyQt
- customized per-application/per-site by BLM presentation data and plug-ins
- customized per-user by individual settings
- real-time operation
- macros
- ...

CAPS' web front-ends

- Tweb mini-framework to sit on top of, and merge/integrate, CAPS, Twisted, Nevow
- all "logic" expressed in plain Python code
  - "**client-connection**"; pages; forms; other resources
- all "presentation" expressed in HTML templates (plus CSS, etc)
  - full-page/form templates; fragment-templates
  - standard Nevow data/render methods (**sequence**, ...)
  - additional data/render methods (**attr**, ...)

CAPS scripting front-end

- full Python interpreter
  - interactive environment for exploration/debug/...
  - running top-level scripts for... whatever purpose!
  - (potentially) could be embedded in any application
- **services** object (queries, toi-creation, transaction control [commit/rollback])
- objects for **blm**s, **toc**s, **toi**s (implicit requests and updates via properties)
- completely transparent synchronization

The CAPScase application

- helpdesk management
  - easily adapted to similar case-management workflows
- includes models for people & organizations
  - could be "hived off" to a library-BLM
- centers on the **Case** toc
  - case-metadata (state, priority, dates & times, ...)
  - various relations to people (handlers, initiator, ...)
  - contains **Action**s, **Message**s, **Note**s, **Document**s...
  - automatic journaling of all meaningful changes

The CAPSpro application

- "generic" case-management
  - variants are in use for software-development, manufactured-products delivery, ...
- variant of CAPScase easiest to adapt to many kinds of case-management needs

The CAPSupphandling application

- structured procurement management
  - per Sweden's public-sector regulations...
  - ...which generalize to other EU countries'
  - could be easily adapted to similar workflows
- models for people, organizations & roles
  - pretty similar to CAPScase's
- centers on the **Procurement** toc...
- ...and its rules-based, structured progress along **State**s (prereq's & conseq's)