Upgrading OpenACS 5.4 to 5.10
A case study at the City of Milan

Antonio Pisano

EuroTcl 2023 – Vienna University for Economics and Business

Introduction

What this presentation is about

- OpenACS in the context of an important public administration
- Successfully upgrading a long running, productive, OpenACS application
  - Costs
  - Challenges
  - Lessons learned

OSAPI

What is OSAPI

- Web-based ERP for Public Ground Occupation.
- Created to manage the entire process of issuing authorizations related to the use of public land (an unavailable asset) under the various offices of the City of Milan.
- In use since 2012, owned by the City of Milan and maintained by PHI Software S.r.l.
- From the beginning looks at public administration as a collaborating community on a single shared database containing spatial, accounting, control and resource optimization information.
- As a shared, community-based application, it makes it easy to implement services to the citizen, monitor transparency and efficiency of administrative processes and implement data views and tools for planning and policing.

OSAPI

What OSAPI does

- The main use-case is citizens or companies requesting a permit to occupy a portion of the public city ground. The request is processed and, in case of approval, normally involves the payment of one or more State or local taxes.

- There are <u>149</u> different types of permits and <u>100</u> tax calculation formulas.

- Permits, dossiers and other application artifacts are implemented using a in-house workflow system. Multiple groups of users with different levels of privilege interact with the system according to the workflow specification.

- Groups of users span across different offices in the city administration, responsible for different types of authorization, the local police department and the public.

- Typical examples of permits are those for construction sites, driveways, weekly markets or restaurant tables located on the sidewalk.

OSAPI

Main goals

Traceability of processes to derive indicators of the efficiency and effectiveness of the workflows in place and the work done.

Schedules showing lists of proceedings that have exceeded the acceptable processing time

Productivity studied by territory, by office, by user, by type of permit.

Profitability by permit type, by year period, by event classes, by territorial centers.

OSAPI IN NUMBERS

Permits and revenues

Filed applications	75.500
Granted permits	70.000
Pending permits	1.500
Denied or archived permits	1.000
Other inactive permit applications	3.000
Tax revenue from open-ended permits	€ 34.000.000
Tax revenue from short-term permits	€ 33.000.000

OSAPI IN NUMBERS

Permits and revenues

Short-term permits revenue forecast 2023	
Public Events	€ 1.251.804,45
Driveways	€ 24.597,84
Scaffoldings	€ 20.150.581,59
Local Police	€ 7.227.778,25
Escavation works	€ 4.866.227,27
	€ 33.520.989,39

OSAPI IN NUMBERS

Application size

OSAPI	
.tcl source files	1.318
Procs	438
.adp source files	766
.xml source files	* 303
Printouts	269
DB Tables	~ 450
DB Dump Size	25GB
ACS Objects	1.5 M
* workflow states (every file is basically to a program)	

OPEN ACS	
.tcl source files	5.080
Procs	3.331
DB Tables	906

OSAPI IN NUMBERS

OpenACS upstream packages

- acs-core
- categories
- file-storage
- rss-support
- oacs-dav
- attachments
- schema-browser
- monitoring
- acs-mail

OSAPI IN NUMBERS

OpenACS downstream packages

- maps: map support
- submit: workflow system
- tosap: main application logics
- z-customization: here are the customizations to upstream api

OSAPI IN NUMBERS

Users, groups and collaboration

COOPERATION	
Cooperating Offices	115
Active Users	631
Yearly office interactions	30.456
Average daily office interactions	138

The Upgrade

Future-proofing OSAPI

- In 2022 the City of Milan renewed its commitment to OSAPI.
- This came with the decision to upgrade the complete application stack, including:
  - Underlying OpenACS version
  - Postgres DBMS
  - Operative system
- I am contracted by PHI Software S.r.l. to povide support to the project.

The Upgrade

Starting point

- Application stack versions
  - Operative system: Ubuntu 8.04 LTS
  - DBMS: Postgres 8.2
  - Web Server: AOLserver 4.5.1
  - OpenACS: 5.4.3
- Development process
  - Version control system: cvs for in-house codebase only
  - Upstream codebase practically immutable

The Upgrade

Main goals

- Application stack versions
  - Operative system: Ubuntu 20.04 LTS
  - DBMS: Postgres 12 (OS packaged version)
  - Web Server: NaviServer (latest release)
  - OpenACS: 5.10.0
- Development process
  - Version control system: git for in-house and upstream codebase.
  - Allow for continuous update

The Upgrade

Roadmap

- Spring 2022: upgrade plan
    - Size the new machine
    - Install the new stack
    - Import database, codebase and settings
    - Setup git repositories
    - Connect upstream packages with upstream mirrors
- Summer 2022: test upgrade
    - Perform the upgrade in a test environment
    - Test the resulting system and validate the upgrade

The Upgrade

Roadmap

- Winter 2022/2023: upgrade rollout
  - Agree on a suitable maintenance window: 2023-01-28 → 2023-01-29
  - Import latest database and code changes
  - Perform the upgrade
  - Validate the upgrade
  - Switch the production environment to the upgraded system: 2023-01-30
  - Post-upgrade support

The Upgrade

Repository structure

- Upstream packages have been replaced with upstream versions
  - Added GitHub mirror as a remote for the package
  - Checked out closest upstream “relative” oacs-5-4
  - The subsequent release branches have been checked out at every upgrade step
  - Packages without release branches (e.g. schema-browser) were checked out and updated just once
- Downstream code has been made part of the main acs-core repository
- The resulting setup has 3 main branches: development, production and upstream. This allows to implement a release cycle concept and receive upstream code regularly.
- The main server configuration files are also tracked via git in the core repository

The Upgrade

The OpenACS upgrade – 1st Step

- The starting database had to be ported to Postgres 8.4 first
  - Installed Postgres 8.4 from source and restored the original dump
  - Upgraded stepwise 5.4 → 5.5 → 5.6 → 5.7 → 5.8
- Issues encountered:
  - packages/acs-admin/www/install/index.tcl → had to comment the code trying to connect to cvs, as it was not accessible by the machine
  - acs-subsite/sql/postgresql/upgrade/upgrade-5.8.1-5.8.2.sql → the constraint that is eliminated here can be named differently on an old database

The Upgrade

The OpenACS upgrade – 2nd Step

- The upgraded database was dumped from Postgres 8.4 and restored in Postgres 12
  - Upgraded stepwise 5.9 → 5.10
- Issues encountered:
  - acs-kernel/sql/postgresql/upgrade/upgrade-5.10.1d8-5.10.1d9.sql → Again a dropped constraint was named differently
  - acs-tcl/tcl/apm-procs.tcl and acs-tcl/tcl/apm-install-procs.tcl → code was expecting a cache to exist that was not there yet (fixed upstream)

The Upgrade

Pitfalls, regressions, fallouts

- Command lines used via exec
  - trml2pdf: not packaged on Ubuntu anymore. Replaced from current GitHub repo. Current version behaves slightly different (e.g. page orientation).
  - Some utilities were overlooked and had to be re-instated.
- Postgres
  - Postgres 8.4 removed many implicit type casts. Multiple queries in in-house codebase had to be adapted.
- OpenACS
  - The “inform” ad_form widgets do not pass their values on form submit. Many pages had to be reworked.
  - Logics to compute the system URL are much stricter in later versions.

The Upgrade

Other issues

- OpenACS
  - Deprecated api, e.g. “with_catch” vs “try” had to be replaced in many files to keep warnings under control.
  - Some local changes to upstream code had to be reinstated.
  - Various specific configurations and setups had to be ported.
- JQuery UI
  - Datepicker regression on Internet Explorer (still used in may offices). Solved by updating the widget assets.

The Upgrade

The costs of upgrading

- Upgrade process
  - **PHI Software** - hardware requirements analysis tests and coordinations with the City Administration, backup and restore of data and sources, configurations and validation: 96 person hours
  - **Me –** upgrade design, development workflow design, git setup, backup and restore of data and sources, upgrade rollout: 64 person hours
- Tests and Post-upgrade support
  - **PHI Software** – tests, user-support, bugfixing, post-upgrade coordination with the City Administration: 328 person hours
  - **Me –** port-rollout fixes, git training, git setup tuning: 20 person hours
- Tackling all regressions post-upgrade took roughly 2 months

Takeaways

What we learned along the way

- Testing testing testing!
  - Testing and post-upgrade support was by far the costliest part of the project
  - Having a good automated test suite is also costly, but worth the effort, at least for core functionalities.
- Upgrading OpenACS installation was the easy part (at least for OSAPI)
- Upgrading regularly is better
  - Code is kept up to date (deprecated api, best-practices...)
  - Regressions can be absorbed gradually
  - Ensures best features and security
- Git repays a rather steep learning curve with much more control over the codebase

Acknowledgments

- Many thanks to “Comune di Milano”, in particular to Laura Giuseppina Colombo, for investing in this project and in the future of OSAPI in Milan. She also provided many information and figures used in this presentation. Thanks also to Ubaldo Salerini for his usual kindness and patience.

- Thanks to PHI Software, Stefano, Luca and Serena for involving me in this project and for much heavylifting endured during these many months. Thanks also for being here with us to tell you about it!

- Finally, many thanks to Hector Romojaro, whose work I have sometimes “borrowed”, in particular the git setup, largely inspired by the one in use at LEARN.

Thanks for watching!

Questions?