The use of Plone for enterprise solutions

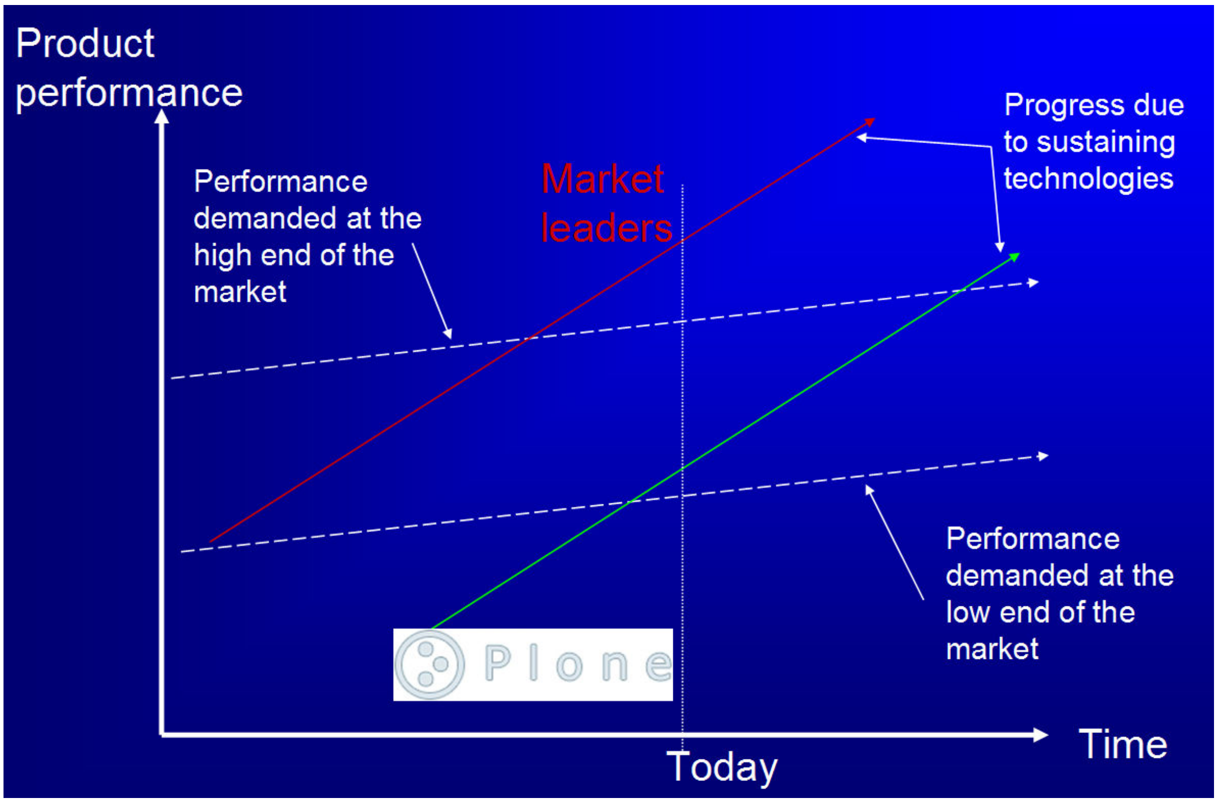
Dr Alexander Samarin, WWW.SAMARIN.BIZ

A few words about me

A good description of my professional specialization and interests is *how information technology can be used to transform business operations to improve significantly their effectiveness and efficiency*, or in other words "Do the right things" and "Do the things right". I have over 9 years experience working with Enterprise Content Management (ECM), Business Process Management (BPM) and, in particular, Livelink from OpenText. I successfully architected and delivered several Livelink solutions for different needs ranging from a heavy-duty production process to a federated distributed organization.

The idea behind this talk

I consider Plone to be a disruptive technological innovation in the ECM area. Plone typically delivers the performance demanded at the low end of the market, as shown here in an illustration taken from the book "The innovator's dilemma" [1].


Product performance
Performance demanded at the high end of the market
Market leaders
Progress due to sustaining technologies
Performance demanded at the low end of the market
Plone
Today
Time

Let us assume that Plone targets also the high end of the market. It is a fair assumption as there is a project underway involving Zope 3 and many of the Zope-based Content Management System (CMS) projects. So, the idea behind this talk is to share my experience of how to "prepare" and "arm" Plone for its move into the ECM and BPM markets and into the enterprise environment. There Plone will be treated as an "adult" and it must manage to survive in this hostile environment.

Plone implementations as socio-technical systems

Practically all implementations of Plone are socio-technical systems. This type of system is designed for large audiences; but your client (the buyer of your services) is not the user of your services. Sometimes, you will never even get the chance to talk to the end users. Instead, you will only have contact with the stakeholders of the system; and very quickly, you will find that there is real tension between the stakeholders.

The book "The art of systems architecting" [2] gives us two good insights. The first one is the "motivation" insight – when there is a change who benefits, who pays, who provides, who loses? The second one is the "resource-limited situation" insight – the <u>value</u> of a service or product is determined by what a buyer is willing to give up to obtain it.

In addition to these two insights, which usually increase the tension amongst the stakeholders, there is another important factor – it is not the facts, it is the perception that counts. As a result, the most important heuristic for designing a socio-technical system, and sometimes a bitter lesson, is "*How* you do something may be more important than *what* you do".

Let us imagine that there is an opportunity for implementing a Plone-based solution in an organization. Either the top management ordered the IT department to chose a good ECM solution via a formal RFP-based selection, or someone from the organization is pushing Plone. In both cases, you have to earn the confidence of the stakeholders in a very short time. During the evaluation of a Plone-based solution, you need to address most of the stakeholders' concerns (you should determine these concerns first); you do not increase the existing tensions and you offer a prototype that implements about 30-40 % of the desired functionality.

In general, the main tension is between the business units (your potential users) and the IT unit (your potential client). If the business units are actively involved in the evaluation, then you may play the role of "solution architect": confidence from the business units (you are the users' agent) and in-depth understanding of the IT (you are a helpful resource) is a winning combination. In the case where the IT staff talks 95 % IT and only 5 % business, you have to keep a 50-50 balance.

The "human" part of the business environment is often the most difficult.

For the sake of clarity, and to span the differences between cultures and countries, I will use a simple classification of IT staff members by making an analogy with the construction industry:

- technicians … maintain buildings
- developers … construct buildings
- architects … architect/design buildings
- solution architects … plan a city

<u>Technicians</u> are usually very conservative – they do not want "yet another application on their plate". To be accepted, a new application must not need baby-sitting and must allow automation of all maintenance tasks (backup, availability/resource monitoring, etc.).

<u>Developers</u> do not understand why someone in the organization wants an external application – after all, they the developers can deliver everything using Java, LAMP, PHP, etc. To be acceptable to them, a new application must, as a minimum, reproduce exactly their layout and their styles.

<u>Architects</u> may admit that they have no clue about Plone. In general, you should be ready to describe how you plan to integrate Plone with existing systems. Moreover, please, do not give the impression that somebody will be obliged to learn Python!

Solution architects may ask you how Plone fits into the Zachman framework [3] and TOGAF (The Open Group Architecture Framework) [4]. If you have not been asked such a question then consider yourself lucky not to have met such a person.

Other staff members may be described as follows.

The project manager will ask you for examples of similar Plone installations and about Gartner Group recommendations. As Gartner mentions Plone in their publications at least a few times, I would recommend reading the most recent ones. Be prepared that in some places Gartner has been seen as the safe choice, akin to the old days "you can't be fired for choosing IBM". Another potential landmine is the in-house project management practice – clearly explain why you plan your work as an agile project.

The IT manager is concerned about business continuity, for example who will the maintenance contract be with? Are there other people/companies who can maintain your solution? Another important point to bear in mind is the manager's personal integrity – do not demonstrate that Plone easily resolves some problems which constitute the main part of the IT budget.

The business process owners' main concern is how quickly a new application can be adapted for new business practices, procedures, etc. Will each adaptation require significant resources from the IT unit or a third party? Can some work be delegated to the business users? Can the business users manage the evolution of the application without the systematic involvement of the IT unit? In summary, the business users must feel "ownership" of the new application. So build a system for the users, via small incremental steps and at the users' pace.

All users are important since they are the business process owners; some will probably be "super users". The perception of a new application often comes from them. You may hear "I like this set of colours" or "My son created a similar Web site for himself – can you just deploy it?" To improve their perception, visit the users often and systematically, like a farmer checking his herd ("fait un tour des vaches"). Also, be prepared that other stakeholders may actively prevent you from visiting the users.

The "system" part of the business environment is usually a complex system of systems that has evolved over several years. With a very high probability, you will no doubt find good examples for some of the following anti-patterns:

- large-scale in-house IT development – the business is considered as "special"
- implementation of departmental rather than organizational solutions – too difficult to agree on a common solution
- running highly complex IT environments with multiple duplication and redundancies
- creating serious integration and migration problems for IT projects – existing applications have not been properly architected to address these issues
- co-existing competing development approaches – developers work "their way"
- lengthy, often unsuccessful, IT projects – no critical analysis of internal IT projects; failure is considered experience and a potential for success next time.

To survive in such a hostile environment, a Plone-based solution should be a good citizen in the enterprise. The best way to fulfil this mission is to be easily adaptable to practically all aspects of the organization including

- the policies, priorities and organizational structure,
- the existing data, IT systems and equipment,
- the level of computerization within the company,
- the constantly changing business processes,
- the computer knowledge and culture of the users,
- the available budgets, and

- the size and complexity of problems to be addressed.

In general, the delivery of highly adaptable solutions is no longer a technical problem. All required technologies and tools are available and industry experience has demonstrated that the key enabler for an adaptable solution is an enterprise-wide, service-oriented, architectural framework which focuses on business process management, i.e. its modelling, automation, control, measurement, optimization, etc.

Whatever degree of architecture is employed, the following heuristics will always be useful.

- Integrate systems via an interface – integration, data uploading, etc. can be carried out with in-depth knowledge of Plone.
- Operate as a service – easy to find, integrate, validate, replace, etc.
- Make minimal modifications to Plone – follow the spandex principle "Just because you can, doesn't mean you should" since if you customize Plone you become responsible for all subsequent problems.
- Welcome project modifications – this is one way to win over the users who are usually worried that they have to specify everything in advance; sometimes, their original description of the problem is not necessarily the best or even the right one.
- Keep a good audit trail – be ready that someone may change something in Plone without telling you in advance, or even without admitting to it.
- Implement external monitoring – try to be clever with such monitoring; although testing of ports is necessary it is not sufficient to ensure that the system can do the job.
- "Maintainability" is crucial – if you are serious that a Plone-based solution will work without you and won't cause you a headache, then prepare it for power cuts, incorrect data, migrations, etc.
- You may break any heuristic *provided that you master it* – remember this if you are accused of not following your own rules!

Case 1 – Research department business system

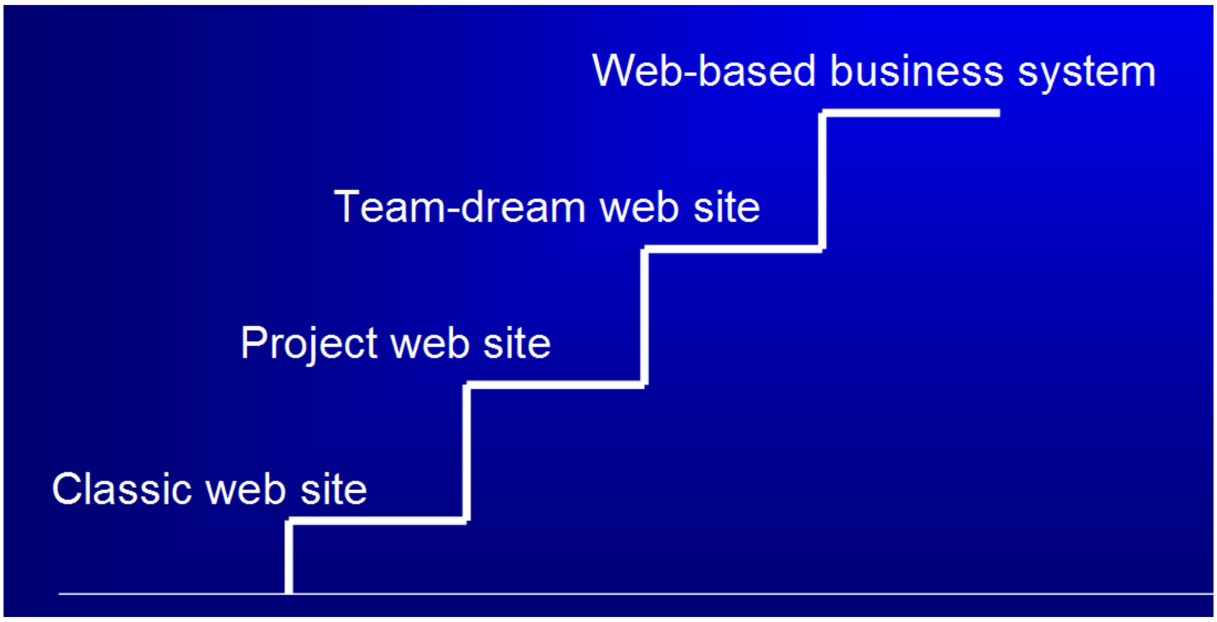
Recently the Theoretical Physics Department of the Research School of an Australian university carried out a feasibility study for creating a departmental web site using Plone. Of course, the Research School already had an official web site which offered reasonable functionality, but not everything that was wanted. The desired features were

- people management (tracking of many visiting fellows),
- management system for Ph. D. students (housekeeping of their research, lectures, etc.),
- personal visibility on the Web for staff members (should be easy for a professor to maintain his/her own web site),
- web-based environments for grants (temporary collectives with well-defined objectives), and
- publications management (ease of generation of personal/departmental/group/etc. lists of publications, ease of use with BibTeX and LaTeX).

It is well known that the academic environment is very federated and that local groups exercise a relatively high autonomy. At the beginning of this pilot we discovered that we were allowed to create a departmental web site, but only if the official web site was mimicked in both presentation and layout. In addition, we learnt that many similar bottom-up initiatives had died a death following a typical pattern: a student installed a collaborative system, supported it and then it disappeared when he/she left.

The problem with the publications and their bibliographical references was quite a common one – information is very decentralized over several repositories, there are personal files, personal web sites, etc.

To clarify what type of collaborative system we were going to build, I employed the ladder of collaboration concept (see illustration below).



Classic web site: one writer / manager and many readers; simple library with read-only access; possibly use of notifications.

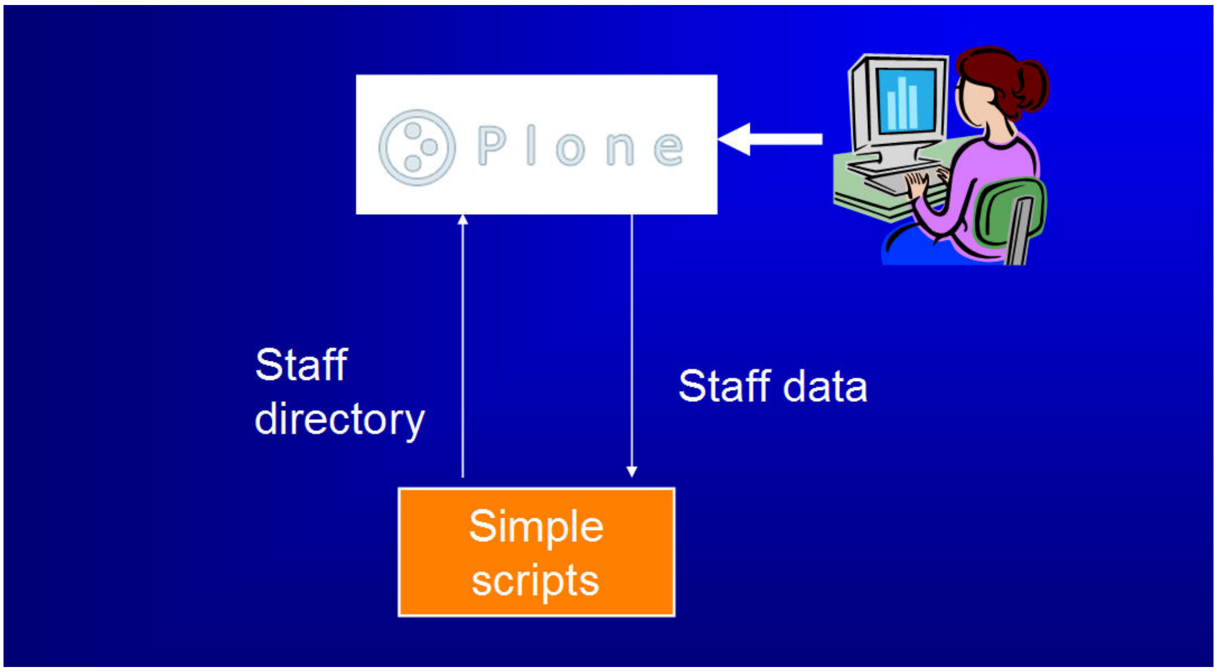
Project web site: one manager and many writers / readers; library with protected documents / folders to keep project-related information and documentation; task assignment (for advanced projects); discussion (depends on group culture).

Team-dream web site: role-dependant functions and permissions; content management practices are managed in accordance with a recognized quality management system such as ISO 9001; typical business procedures (e.g. maintenance of publications) are formalized and executed as workflows; some automation.

Web-based business system: CMS is an interface to the business system; workflow is the business process integration tool; access to centralized restricted information; integration with existing applications.

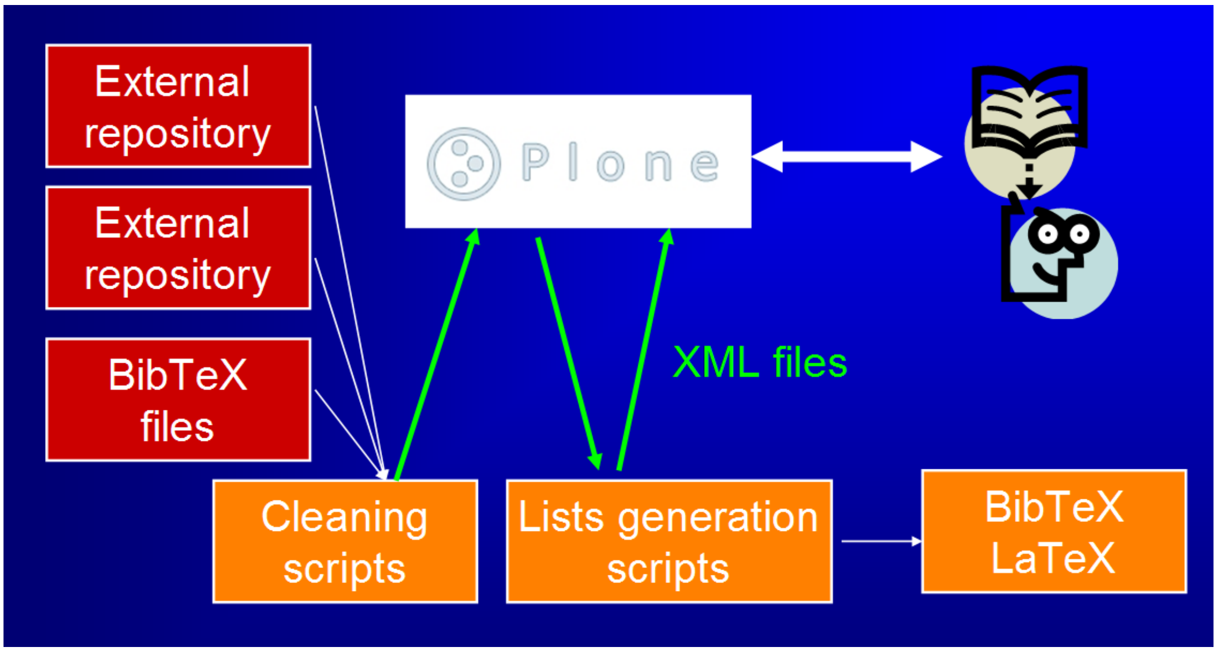
We found that the different groups and individuals within the department targeted different steps on this ladder and also had a different pace in advancing up it. So, we understood that using Plone we had to implement a departmental business system which could be maintained by the theoretical physicists (only one person among them knew Perl) with minimum help from the Research School's IT group.

Two areas required automation: people management and publications management. For the management of people, we provided an archetype to keep the data needed and a few simple external scripts to extract people data and generate a staff directory, and then upload it to Plone as shown in the illustration below. An Administrative Assistant carries out all necessary work.



The management of publications is more complicated because the bibliographical information is originated and duplicated in many different resources including on-line publication servers, personal records, etc. In addition, all existing resources provide different metadata and demonstrate a different quality of data. Any effective management in this case implies a centralized solution – a repository with all necessary metadata and guaranteed quality. By definition, such a repository must allow a round-trip as people want to reuse validated bibliographical data in their daily work (e.g. in publications).

We provided a couple of archetypes and a few external scripts for the processing of incoming information, to upload/download information, to generate different lists, etc., as shown in the illustration below.

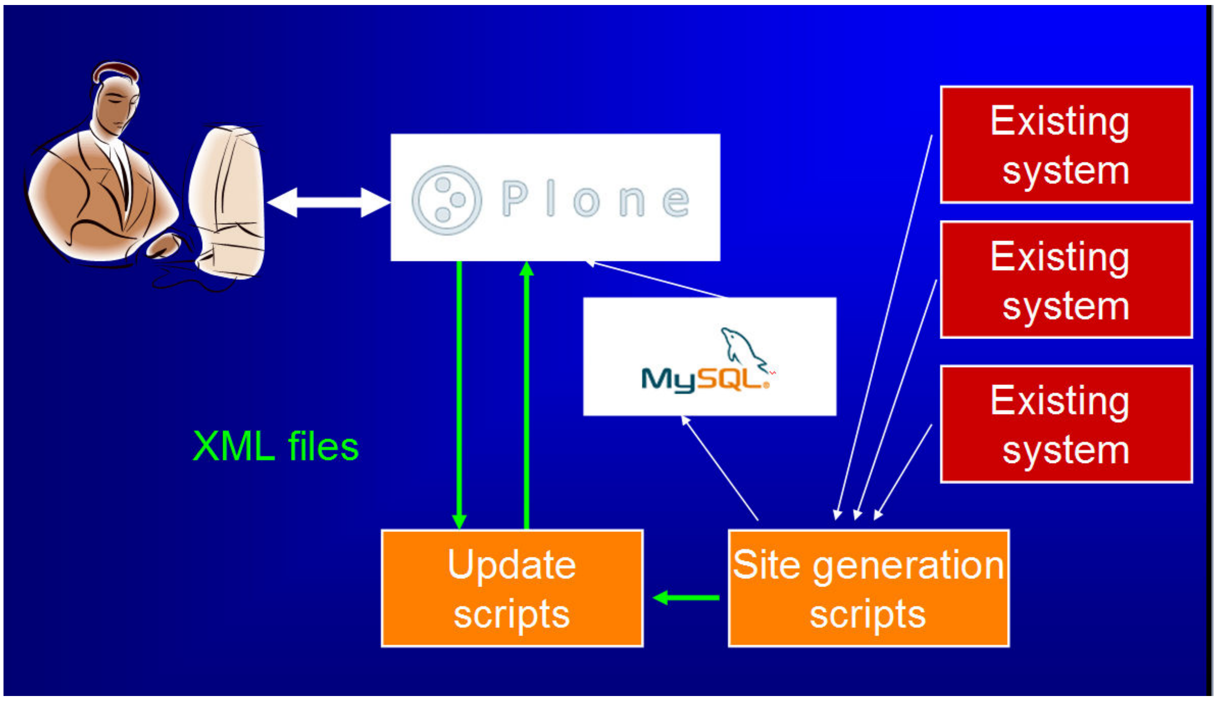


Case 2 – Garage client management

Recently, we were asked to provide a customer-relation management system for a group of three garages. As a temporary solution (the mother car-manufacturing company had promised to provide a proper solution *soon*), the group wanted to have more transparency in the management of clients by their salespersons. The information about all active clients (about 24 000) and all sold cars (about 33 000) was available in several different old-fashioned applications. However, the information about prospective clients was handled manually by salespersons in quite different ways.

The Plone-based solution had to integrate all the available information and to keep some additional information such as the clients' hobbies, interests, wish lists, etc. In addition, it had to provide some business intelligence for matching a client's wishes with the available cars. Such a matching could be quite straightforward – just a fuzzy search in a list of second-hand cars; or it could be a bit more intellectual – offering a new car to an existing client and re-selling his/her car to a prospective client.

Considering that all information had to be aggregated around the user's needs (salesperson, manager, etc.) and that good navigation was essential, we provided a few archetypes to keep client-related information, a read-only car database (in mySQL) and many pre-generated HTML files with different types of list (for example, a list of all cars sold by each salesperson, a list of all clients for each salesperson, etc.). Since we expected that these lists would evolve in logic and presentation, we kept their generation outside Plone. As a result, practically all content of this Plone-based solution was programmatically generated by a few scripts as shown in the illustration below. We carried out such a generation once a day and updated Plone as necessary.



Principles of these solutions

These two cases were implemented based on the same principles listed below:

- use Plone out-of-the-box;
- add a few mature products;
- use few archetypes for the storing of application-specific data types (very simple);
- use a Plone API;
- keep all business logic in external scripts, not inside Plone;
- make the solution easy to maintain without modifying Plone;
- make the solution easy to maintain without knowledge of Python.

The Plone API (or PAPI) is the enabler for enterprise solutions:

- XML-RPC based;
- adapted to a few XML-RPC implementations (e.g. Perl and Python);
- implements basic operations for archetype-based objects;
- at present, it is just a primitive implementation which needs to be properly re-engineered.

To simplify data uploading to and downloading from Plone, we used a simple exchange XML format built on top of PAPI. The use of such an exchange format simplifies considerably the integration.

```
<server url="http://localhost:8000">
<objectlist type="data">
<object type="PublicationRef" id="prtest" parent="/pubs">
  <title fieldType="Products.Archetypes.Field.StringField">prtest</title>
  <source
fieldType="Products.Archetypes.Field.ReferenceField">c72fd108cd23aab305a8d84768b08fa7</source>
</object>
</objectlist>
</server>
```

Plone and ECM applications

At present, Plone is good for portal-based integration which aggregates information from disparate resources, collects data around users' needs, provides "an umbrella" for existing systems, etc.

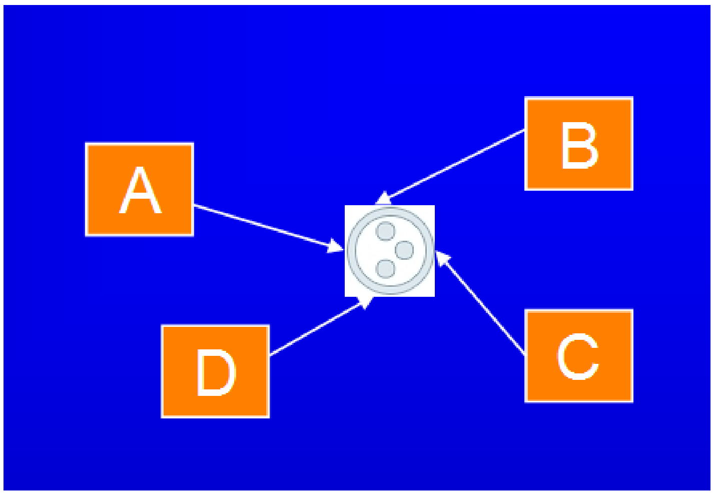
In the general document management area, I would rate Plone as acceptable, mainly because of archetypes, but versioning and a much better permission mechanism are needed.

Plone is still weak in areas involving strong compliance requirements, mainly owing to the absence of proper audit trail functionality and a good permission mechanism. At present, I would not recommend Plone for the implementation of solutions which must be formally certified for compliance with the ISO 9001 Quality Management System. Plone alone is not good enough for records management.

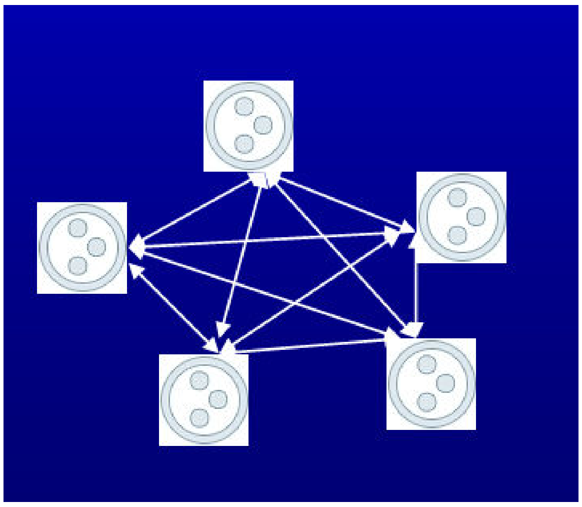
I think that the area of business process management (where Plone is still weak) is the most promising for the successful evolution of Plone, because there is a market need for this and good BPM helps to cure some other problems, e.g. compliance with external requirements.

There are several patterns for enterprise solutions with the current version of Plone.

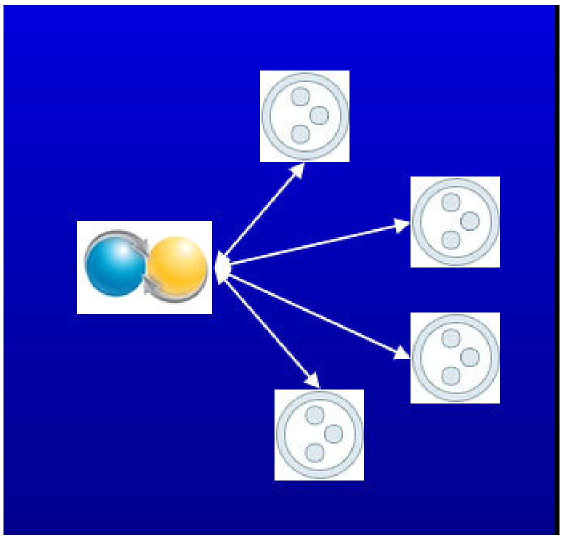
Information aggregation hub or portal:



A group of federated business systems, e.g. many research groups and grants with strong needs for exchange between people and publications:



Satellite partner for a central commercial system:



For example, the central office (a couple of hundred people) of an international organization runs a commercial ECM product which provides comprehensive functionalities including compliance and business process management. At the same time, for its offices, member states and working groups (comprising several thousand people) which are spread over the globe and often do not have a good IT infrastructure, the organization provides a set of Plone-based local solutions. The satellites serve as localized versions, local caches, entry points, etc. The central system helps them to operate efficiently, e.g. provides the management of records, automation, etc.

Plone and the BPM market

Let us define Business Process Management first. I like the following definition: BPM allows you to model, automate, control, measure and optimize the flow of business process steps that span your organization's systems, people, customers and partners within and beyond your corporate boundaries.

Gartner estimates that there are currently over 100 BPM vendors. To compete efficiently with them, we have to offer something different from just a product. I think such an offer could be an architecture (some rules, principles, methods, patterns, practices) or a framework for improving BPM systems for which Plone (i.e. the framework) acts as a coherent set of pre-fabricated building blocks.

Actually, all enterprises have their own BPM system; some enterprises want to change it because it is complex, a function of its history, chaotic, inefficient, etc. Most enterprises want to carry out such a change incrementally. To be successful in such a transformation, they need a structure (or an order or an architecture, or a vision). This is provided by the framework. Application of the framework to a business system with consideration of transformation goals and priorities, immediately shows which parts of the business system are missing or need to be replaced. This is a job for Plone – quickly provide such parts via pre-fabricated building blocks which are easily configurable for typical business processes.

The framework for improving BPM systems is built on the systemic approach and adaptability as the main non-functional characteristics. The framework, from the business viewpoint, is a *generic operational model* for service- and process-centric activities. From the IT point of view, it is an enterprise-wide service-oriented *multi-layer model* for the realization of the generic operational model. The overlap between these two views gives a structure into which new features can be added like pieces of Lego. Having used such a system since the year 2000 [5,6], in my experience it is proven that BPM systems can be improved faster, better and less expensively than is usually perceived. Conceptually, the NetWeaver platform from SAP is similar to the framework, but the framework is more mature and much easier to implement.

The generic operational model

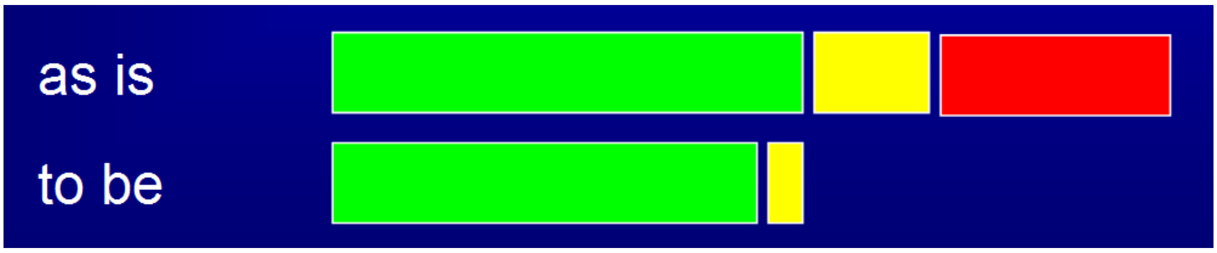
The business process is driven by *business events*. For each event, there is an associated *business procedure* to be executed. A business procedure includes *business rules* and *regulations*. An elementary or indivisible business procedure is called a *business task*. Each task operates with some *business entities* or *business objects*.

Classification of the tasks:

- intellectual tasks — mainly value-adding activities;
- verification tasks — different types of check that activities have been carried out correctly;
- administrative tasks — process-support activities.

The aims of the business automation are automation of all administrative tasks and as many as possible of the verification tasks, as well as the provision of all information needed for carrying out the intellectual tasks. The automation of administrative tasks, in addition to saving work time and improving the quality of operations, also brings the possibility for heads of service to delegate certain tasks to less-qualified staff members.

In a typical service- and process-centric company, activities may be classified as 50 % intellectual tasks, 30 % administrative tasks and 20 % verification tasks. The deployment of this type of architectural framework can improve IT-based operations by some 200 % over a 2- to 3-year time frame: 10 % reduction in intellectual tasks, 400 % reduction in verification tasks and elimination of all administrative tasks.



The commonest pattern of business procedures comprises a mixture of the various types of task:

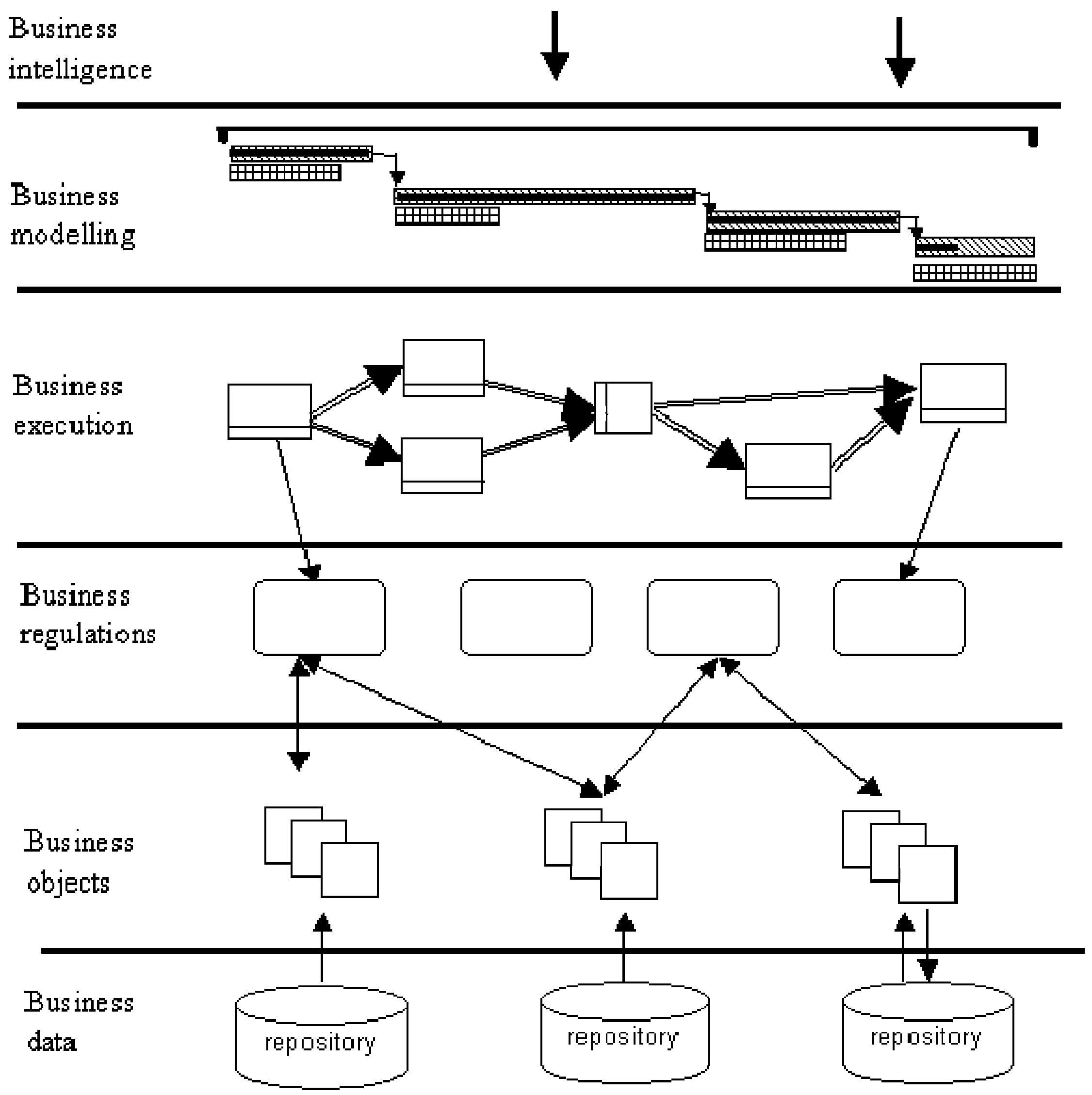
- "get" some business objects from some repositories (administrative task);

- "do" something with these objects (intellectual or administrative task);
- "check" the integrity of the work done (verification task);
- "put" the objects and / or any new objects into some (probably different) repositories (administrative task).

The multi-layer model

To automate and manage such business procedures a multi-layer model (see the figure below) has been used. Each layer is a level of abstraction of the business and addresses some particular concerns:

- the business data layer comprises information that is stored in existing repositories and traditional applications;
- the business objects layer comprises objects (actually, business data containers) specific for a particular business;
- the business regulations (and rules) layer comprises the actions on the business objects, which must be carried out to perform the business tasks;
- the business execution layer carries out the business procedures (each procedure is an orchestrated set of manual and automated tasks);
- the business modelling (and monitoring) layer analyses the business events, which summarize execution of the business procedures;
- the business intelligence layer implements enterprise-wide planning, performance evaluation and control actions applied to the business procedures.



Each layer has two responsibilities: it uses the lower layer functionalities, and it serves the higher layer. Each layer has a well-defined interface and its implementation is independent of that of others. Each layer comprises many building blocks that can be reused in different activities.

Another practical observation is that different layers have life-cycles of different time scales: typical repositories have a 5- to 10-year life-span while the business requires continuous improvement. Because of the implementation independence of the different layers, each layer may evolve at its own pace without being hampered by others.

As a rule, the existing applications already implement many of the business objects, regulations and events, but in an unstructured way. The events, rules and objects are intermixed, not reusable and need to be implemented again and again. This model is a tool which helps the organization to design the system in business terms, but not in terms of IT tools. In addition, this model doesn't require that all layers be implemented at the same time or even provided in a single project.

A service-oriented architecture

Where in this "architecture" are databases, application servers, graphical user interfaces, XML, web services and many other IT gadgets? By definition, it doesn't matter. In general, a layer, a building block and a version of a building block are services. They could be executed inside a single program, within many programs on a single computer, on an application server or in a distributed environment. Services may be implemented in-house today and outsourced tomorrow, or implemented manually today and automated tomorrow.

Each service is accessible via other services and programs through its Service Programmatic Interface, which can be implemented by traditional API, XML-RPC, CGI, web services, etc. In general, a service may use a persistent storage (a database or file system). It may be executed on an application server and may have its own graphical user interface.

For example, a CMS is a repository for a number of business data. It needs a database, various features of an application server, a web-based user interface and some programmatic interfaces. But, the CMS is just a building block. For migration or dependability purposes for example, we may even need to run several different versions of the same CMS.

In some ways, the traditional 3-tiered architecture is perpendicular to the plane of this framework, as each service has some of these tiers. Sometimes, this architectural framework is perceived as a mechanical extension — just an extra 3 layers — of the 3-tiered architecture.

However, the perception of this as an extension is a trap which must be carefully avoided when explaining this architectural framework to IT personnel. Another trap is to treat all services as reactive or passive components when, in practice, modern complex and distributed systems require more and more active and proactive components, such as agents, to monitor and manage them.

Typically there are elementary indivisible services and there are composite services (constructed from other services). The binding and composition of services are dynamic. Some services may share some servers, and so on.

Implementations of the framework

The table below shows how the framework was implemented using Livelink.

Layer	**Tools**	**Comments**
Execution	Livelink built-in workflow; external automation agents (robots)	Activity-based workflow with graphical interface
Regulations	Livelink forms; Jython	Descriptive logic via forms; procedural logic via Jython
Objects	Livelink forms; Livelink categories	Data or metadata containers with a few access methods
Data	SQL storage; external storage	

The table below shows how the framework can be implemented using Plone.

Layer	**Tools**	**Comments**
Execution	Built-in workflow, internal automation service, external automation agents (robots)	<u>Activity-based workflow is missing</u>
Regulations	Archetypes, Python	Descriptive logic via archetypes, e.g. decision tables; procedural logic via Python; <u>access to external systems via Java API</u>? <u>access to external system via Web Services</u>?
Objects	Archetypes	Data or metadata containers with a few

		access methods
Data	Built-in storage, SQL storage, external storage	

Conclusions

- Plone can deliver enterprise solutions.
- Document management needs enhancements.
- An activity-based workflow is mandatory (embed or sub-contract a good tool).
- A Plone API is mandatory.
- There is need for a community-based "centre of excellence" for enterprise solutions or, even, SWAT teams.
- Combining the architectural framework and Plone allows the move into the BPM market.

Bibliography

[1] Christensen C., The innovator's dilemma, HamperBusiness, 2000.

[2] Maier M. and Rechin R., The art of systems architecting, CRC Press, 2000.

[3] http://www.zifa.com/

[4] http://www.opengroup.org/architecture/togaf/

[5] Samarin A., ISO: integrating the WEB and document management, Proceedings of Documation, Paris, 2001.

[6] Samarin A., Agile SOA for Process Automation and Integration, www.ebizq.net , 2004.