Adaptive Version Control in ERP Environments

A. Selmeci
Óbudai Egyetem AMK, Székesfehérvár, Hungary
E-mail: selmeci.attila@amk.uni-obuda.hu

***Abstract*—The modification of anything brings changes in a system. If we consider only the IT world we can think about hardware and software elements, components or even huger systems built from other elements or components. An Enterprise Resource Planning system, which is the central heart of a company today, is not a single, simple solution, but a very complex, integrated environment. The integration takes part internally, like database system or web servers; externally like interface based communications with other solutions; and technically like using storage, network, hardware, etc.**

The modifications on different level build versions and these versions should be applied occasionally or in a regular manner. The goal in this paper to show the possibility of hardware-software solution, which is adaptive enough to obscure the modifications brought by versions. In the same time we introduce the availability of analogous versioning as well.

Keywords: **ERP, version management, multi-tier, storage, virtualization, redundancy**

I. INTRODUCTION

The software applications can be grouped according to the size, functionality, architecture, etc., but almost all has software implementation bugs. These bugs should be corrected some times and the software developers provide new releases, versions, or even bug fixes, patches for the given software.

In the real life the machines (simple or complex) have also components, mounting parts, which should be repaired or sometimes replaced by a new one. We can think about corrosion, physical damages, using of pure material, inappropriate design and manufacturing, etc. Many cars, or other machines are recalled on a designing bug, which comes out only in the special cases occurring in daily usage.

The hardware elements around the software applications can be regarded as such special products, like cars. These can have issues, problematic building elements or any component, which could fail.

The producer, manufacturer repairs the bug, or issue, or even problem within the original software application, machine, or hardware. They should internally test it and provide the solution in a patch, or in a new release. Sometimes the new releases are not really implementable, but the new one should step pin the place of the original one. In software solutions would we can ask for a migration path between the two versions, but in hardware environments or among used machines total cutover is needed as well.

The next chapters of this paper describe the versioning types, occurrences in business application environment and the possible theoretical future and implementation of a sustainable, well-managed version control.

II. ERP SYSTEMS AND THE VERSION MANAGEMENT

In case of a business application, like Enterprise Resource Planning (ERP) systems, we have more complex environment and co-existing sources of errors. An ERP system has nowadays at least three-tier architecture. [4] It contains a database engine, an application layer, where the business logic runs and a frontend layer. In case of web application there could be more tiers. [5] Each tier has its own purpose and internal architecture. Fig. 1 visualizes a standard three-tier ERP and a four-tier web enabled ERP architecture.

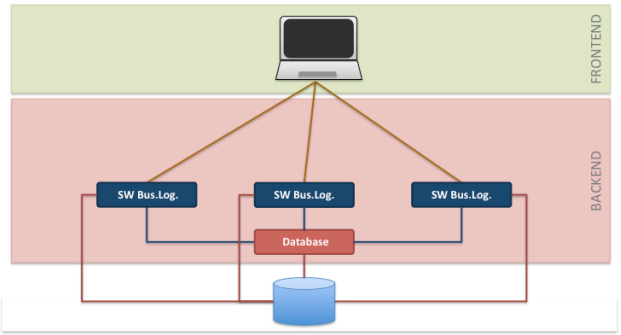

Figure 1 – ERP simplified architectures

These components of the multi tier architecture can be application specific or general-purpose applications. Let us consider for example a web-based software application or ERP solution having three or four tiers. To store the data many ERP systems use a database management system (DBMS). Some installations are database dependent, like applications using open source technologies (e.g. MySQL). Others are database independent or rather they can use, support not only one database management system, but several ones (like SAP does with Oracle, MSSQL, DB2, Informix, MaxDB, Sybase ASE or even HANA DB). For the web enablement there should be somewhere a web server, which provides the web contents, stores the mime objects and handles the HTTP requests. The same is true for this web layer as for the database layer. There are some special ERP applications, which contain their own internal web server implementation, but others use one from the market (e.g. Microsoft Internet Information Server, Apache HTTP Server). These standard, public solutions (DBMS and web server) are separately developed, tested, distributed and supported. They have their own version management and version control inside.

The Fig. 2 below sketches the main software and hardware layers, component under an ERP system.

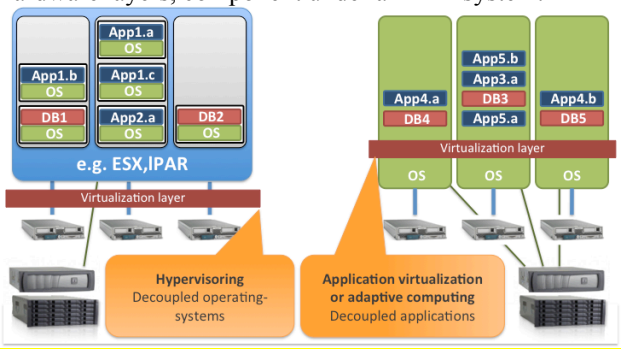

Figure 2 – Undelaying components supporting ERP

Each of the applications should run on a computer or node, which can be physical or virtual as well. There are standard open source operating systems and proprietary ones. The non open-source operating systems mainly have their own hardware as well (like AIX on IBM machines, HP-UX on HP computers (earlier PA-RISC, nowadays Itanium based), Solaris on SUN/Oracle servers (SPARC), MacOS on Apple Macintosh machines). When the manufacturer designs and produces the machine and the operating systems as well, it can exploit the hidden features of the hardware in the operating system as well. This combination can bring a good performance. From version point of view the open-source operating systems are refreshed very frequently bringing mostly new features into the solutions. The vendor-based operating systems have support strategy for the version and patch management. Only bug fixes are provided between the planned version delivery dates

Almost the deepest level of the architecture is the persistent data store, the disks or storage. Here we have again the possibility to choose disks and build our own storage layer or buy a storage system from a vendor. Depending on the internal architecture of the storage among the others different behavior, feature set, and performance are provided. The storage attachment to the computing layer is a huge question because of the different aspects, like speed, reliability, flexibility, redundancy, etc. In version and component management we have to think about component exchanges (e.g. simple disk, disk type, or even controller), software updates or exchange as well.

The lowest remarkable layer is the network. If we think about network we calculate with active elements, like routers, switches, security control component (e.g. intruder detection systems, firewalls), access points, etc. The other part of the network is the cabling. The active elements are again combined software and hardware components; the wires (e.g. coaxial cable, optical fiber cable, twisted pair) are pure hardware elements with special plugs (endings). The version control of networks can be started at configuration changes, like modification of firewall settings, but the replacement of a cable or restructuring the network can also be mentioned as version problem.

III. VERSION CONTROL, QUALITY GATES

Everything is changing in the real world and in the IT systems as well. Our research goes into the direction of analog changes of the systems. It means that the required changes after sufficient test can be smoothly implemented into the corresponding layer without any stop, restart of the system or interference on the end-user side.

Currently the changes are collected and built into a so-called version, which can replace the previous state (or version). By implementing a version we are not really go forward analogously, but rather jump to the next version level in a discrete manner. Our goal is to design environments with more analogous versioning behavior.

The version control is almost in each case a system, which contains the documentation of the changes. It should be attached to adequate quality control as well. In our research we plan to extend this version control with recommended functions and procedures as well. The version control and version management is mixed many times. In our case we speak about version management, which is the mechanism in a software solution how it handles the versions to be implemented.

IV. SOFTWARE ELEMENTS

The applications, especially the business applications like ERP have more software layers. The end-user, who works on the user interface of the system, does not know anything about the system architecture, used hardware elements and network. But he or she should be always on the safe side, so they should not be disturbed. The main behavior of a system in his or her mind is the stability, which is represented by the followings:

- Same outlook of the user interface (only some enhancements are excepted)
- Availability (when they want to use, it is working)
- Acceptable speed (sometimes can be “slower”, but any increment in response-time is regarded as natural)

Depending on the distance from the end-user to the changes the end-user feels different the modification. E.g. some minor changes on the user interface can cause significant problem, but the whole exchange of the storage system could remain unknown. (Check Fig. 4.) The deeper level we work in an IT environment, the more redundancy is required.

As the overview in chapter II described in an ERP system or other business application not only the application solution contains software elements, but the hardware components contain as well software parts. The below list gives an incomplete overview on the usage of software in hardware element:

- User interface
- Configuration
- Internal logic (optional)
- Firmware

This kind of software elements are HW-dependent components and they are not redundant, but live once in a hardware element. The redundancy and availability cannot be managed on this software level.

The application level software components can have their own load balancing and redundant architecture. Unfortunately the version management is not able to work with this kind of features. The bigger systems have their own enhancement model. [3] These models offer different level of changes, enhancement of the software. Fig. 3 below illustrates the different level of software changes.

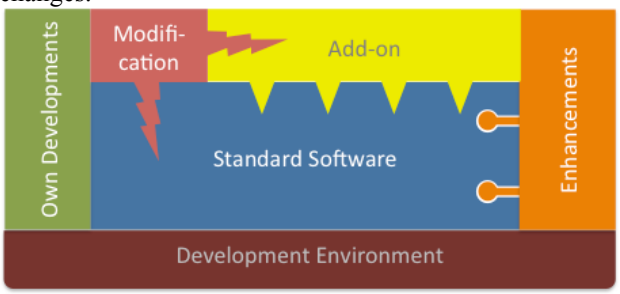

Figure 3. Software modification levels

During software modification the following four types can be significant:

- Configuration (not mentioned on Fig. 4, but can strongly influence the software behavior and outlook as well)
- Modification (changes of delivered standard components)
- Enhancement (soft changes of the standard software; these modification are implemented in predefined entry points without changing the original, only adding functions and logic)
- Add-on implementation (standard extension modules bringing new functionalities)

The own development is a special case, because it can be a totally separated program, applet, which does not harm the whole system, only gives new functions for some end-users.

Depending on the development strategy and model different approaches can be used in developments. One of these paradigms is the usage of so-called reuse elements. This reusability is a programming technique avoiding unnecessary thinking, designing, and programming efforts. If a task requires e.g. a special routine, it should be so designed and specified that other tools could it use later as well. (Well-defined signature, well-designed layers, etc.) Next time the routine should not be developed again to use it for other purposes. It brings less coding and faster development.

From version control point of view this kind of programming can be problematic if we should change some deeper object behavior, which is widely used in the software. Fig. 4 shows a similar dependency as described at the beginning of this chapter.

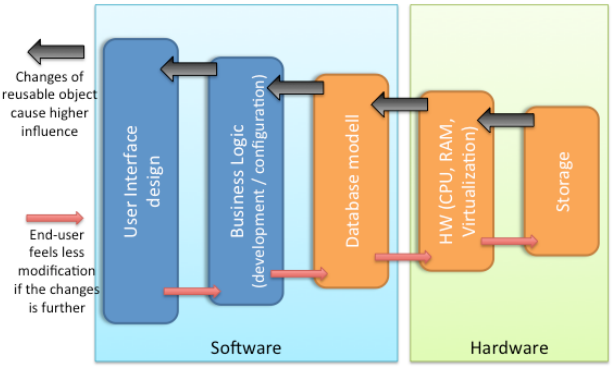

Figure 4 – Distance dependencies

The Figure 4 explains in one chart the dependency of influence

- On the end-user when changes are implemented on different level of the solution and
- On the software, when changes are implemented on different reusability level of the solution.

Similarities are conspicuous, because they effect in opposite direction.

Software can handle the version changes smoothly only if it has online version management. Under online version management I refer to a mechanism, which manages the running processes using current versions of programming units, objects, etc. in a manner, that the code (or object) versions are connected to the running sessions. After the new version is implemented the system records the new version, without removing the instances from the old ones. It keeps the running sessions using the old version alive. If a new session starts, it uses the already the new version. The online version management work strongly together with the session management, because it check when an old version is not used anymore and notifies the version management to deactivate the old version.

With this online version management behavior the software components can be updated smoothly.

V. Hardware Elements, Operating System, Database Management and Virtualization

As described in the previous chapters an ERP system is not running alone, but some other components are needed as well:

- Database management system for ERP data store
- Operating system for running the ERP business logic machine and the DBMS
- Hardware elements, which run and manage the operating systems

Today's virtualization and cloud computing project a direction for us to follow. [6] In global thinking the virtualization can be followed, because we use it in many layer without checking the implementation (like network card teaming, RAID arrays for disks, etc.).

More important is the redundancy in the hardware layer, because it is the basis for virtual design. Our research goes in storage and database management systems redundancy and reliability as well. We expect that with the current available hardware elements we can build fast, reliable, redundant and cheap solution. For that own

defined storage nodes can be introduced with special file system offering. The Fig. 5 illustrates this idea:

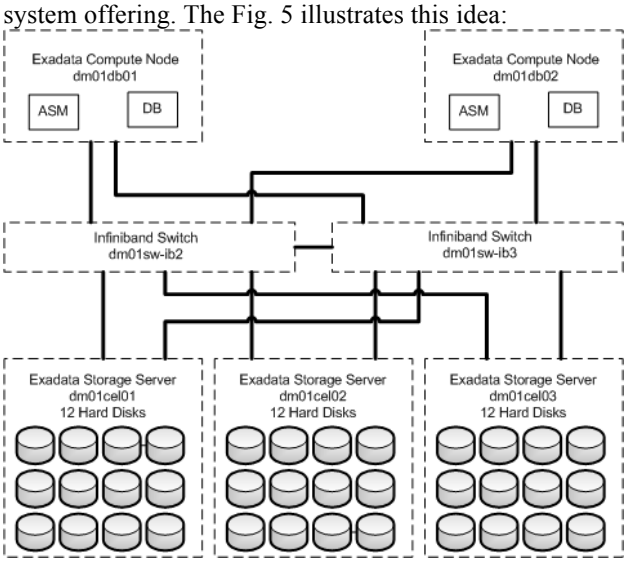

Figure 5 – Oracle Exadata architecture diagram
(Source: http://blog.oracle-ninja.com/)

I copied the Fig. 5 from the above-mentioned source, because it describes similar solutions with Oracle Exadata solution.

It is recommended to detach the applications from the operating systems, like in a high available cluster solution. It brings the availability and load distribution of the software higher level.

VI. CONCLUSION

This paper collects our current researches around building sustainable, stabile ERP environment using the current innovations.

I could figure out that the main point of the sustainability is the redundancy and the layer based exchangeability.

I have introduced and used the following definitions:

- Version control: not only a software managing the documentations of the changes, but rather
- Version management: software solution build in the application managing the software deployment and versioning
- Analogous versioning: continuous version update without stopping the system or components, and unknown by the end-user
- Online version management: mechanism to manage the old and new versions together in a software application

I have introduced a simple, not detailed mechanism, and procedure for adaptive versioning in this paper. These technologies and theorems can be used in education as well to design smaller application.

REFERENCES

[1] A. Selmeci, T. Orosz, Gy. Györök: *Innovative ERP virtualization*, Intelligent Systems and Informatics (SISY 2013), IEEE 11th International Symposium, Subotica Serbia; ISBN 978-1-4799-0303-0 pp., 69-75

[2] A. Selmeci, T. Orosz, Gy. Györök: *Potential of dynamic development in ERP environments*, 5th IEEE International Symposium on Logistics and Industrial Informatics (LINDI 2013) Wildau, Germany, ISBN 978-1-4799-1257-5

[3] A. Selmeci, T. Orosz: *Modification free extension of standard software*, 12th IEEE International Symposium on Applied Machine Intelligence and Informatics (SAMI 2014), Herl'any, Slovakia ISBN 978-1-4799-3441-6

[4] Orosz Gábor Tamás: *Integrált vállalatirányítási rendszerek. SAP: üzleti folyamatok és programozás*, ISBN 978-615-5018-52-7

[5] A. Selmeci, T. Orosz: *Trends and followers in GUI development for business applications with implications at University Education*, 10th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2015), 21-23 May 2015, Timisoara, Romania, ISBN 978-1-4799-9910-1, pp 243 – 251

[6] DANIELS, Jeff. *Server virtualization architecture and implementation.* Crossroads, 2009, 16.1: 8-12.

[7] GOEL, Ms Shivani; KIRAN, Ravi; GARG, Deepak. *Impact of Cloud Computing on ERP implementations in Higher Education*. INSTITUTIONS, 2011, 5: 8.

[8] HASSELBRING, Wilhelm. *Information system integration.* Communications of the ACM, 2000, 43.6: 32-38.