A Dynamic View of ERP System Metrics


Abstract—This paper analyzes the dynamics of different metrics that are specific to ERP systems. Facing the problems of the economical crisis this technical quantitative approach to ERP systems development and maintenance provides an agile approach context. ERP systems usually have huge numbers of tables and related classes, with a growing complexity that easily gets difficult to manage. The problem is to identify what are the things that are used often, what are the things that not used, and which ones are worthy of investing our time in improving. To determine these points precisely from technical perspective this paper analyzes the data from their dynamics trough time and from several ERP specific metrics.

Usually an upgrade is worth up to 1/3 of initial investment and this approach has helped us achieve productive ERP system development environment, and enabled us to perform an ERP system upgrade to second next version, skipping a whole version, which is even more worthier.

The analysis that was performed is shown to be advantageous in the way of quantitative technical approach, in comparison to previous, more soft skills oriented, approaches.

Keywords—agile software development, enterprise resource planning, software quality and metrics.

I. INTRODUCTION

ERP systems connect most of the departments of an enterprise, supporting the information flow inside the organization, and also connecting it with external partners. As such, these systems can be extremely large and complex.

The idea of this paper is to switch from the prevailing, almost standard soft-skills-oriented approach to development of new functionalities for ERP systems and the maintenance of the existing ones giving a vague outcome especially in the times of an economical crisis, by restoring a scientific, methodological approach. This paper will introduce a precise technical perspective into analysis of ERP systems and try to connect the data from the analysis with future requirements for development and maintenance and in an agile way plan for change. A special focus will be given to dynamics of values of this metrics in time.

While before the crisis the companies were ready to invest in technology to enable growth, now they delay or very carefully enter new ERP projects. Even with identified need for ERP upgrade or change, companies are focused on maintaining current implementation.

Using this approach helps a company to easily maintain the current ERP implementation and support new request, and be fit and ready for an upgrade. It has even enabled doing an upgrade to second next major version of ERP (from version 3 to version 5) skipping one whole version, which is considered a very worthy move in ERP world.

Section 2 introduces us to ERP systems and outlines the current state of the art in development, maintenance and upgrade of ERP systems. It also explains basic agile concepts. In section 3 quantitative analysis of software engineering state of an ERP system is performed from six aspects and each of them is discussed. Section 4 connects this exceedingly beneficial information with agile approach. Section 5 explains the results that have been achieved using this approach. In the final section a summary of paper is presented.

II. PREVIOUS WORK

Enterprise Resource Planning (ERP) systems are information systems which out-of-the-box cover most of the business needs of a typical enterprise. Since these systems originated from the manufacturing field [1], this is usually the business area that is supported most. For some other areas there might be a need to customize the ERP system by modifying the existing functionalities or creating completely new ones.

Some of the ERP systems that have the largest market share are shown and compared in Fig. 1. Other common way is to compare them by how they support the key functional areas. An alternate way is to compare them from the technical side, and the authors of this paper did an alternate comparison by performing one of the first comparative analysis of the technical aspects of the ease of maintaining ERP systems [2].

Fig. 1 Gartner's Magic Quadrant for ERP for Product-Centric Midmarket Companies as of December 2010 [3]
As a growing number of companies adopt ERP systems, ERP systems implementation and upgrades are identified as one of the top five IT priorities among global CIOs according to independent surveys conducted by Morgan Stanley [4] and Deloitte & Touche/IDG [5].

Much of the pressure to add new modules to their ERP systems seems to arise about six to 12 months after an organization has gone live with its initial ERP implementation [6]. In that analysis [6] the authors listed some of best practices for ERP upgrade. Most of them are outside of the core software engineering area, and are related to IT management in general. The finding that was listed as a big influence is the focus to un-customize the customizations and to focus on the usage of the out-of-the-box functionalities. As it has been indicated [7] this is not always possible (e.g. a heavily customized ERP systems which can happen for many reasons like company working in country or business area that has not been localized yet from process side).

Based on the previously published research, it can be said that the area of technical analysis of ERP systems in the sense of the structure of the ERP model has not yet been covered. Most of the previous work focuses on the soft skills factors, instead on the general software engineering principles analysis.

One of the principles of agile software development [8] is “Responding to change over following a plan.

In order to adapt this principle one must plan to change. One useful artifact in this approach is the product backlog which basically shows the ordered list of requirements for the system or product being developed by business value for the which basically shows the ordered list of requirements for the system or product being developed by business value for the organization has gone live with its initial ERP implementation [6]. In that analysis [6] the authors listed some of best practices for ERP upgrade. Most of them are outside of the core software engineering area, and are related to IT management in general. The finding that was listed as a big influence is the focus to un-customize the customizations and to focus on the usage of the out-of-the-box functionalities. As it has been indicated [7] this is not always possible (e.g. a heavily customized ERP systems which can happen for many reasons like company working in country or business area that has not been localized yet from process side).

Based on the previously published research, it can be said that the area of technical analysis of ERP systems in the sense of the structure of the ERP model has not yet been covered. Most of the previous work focuses on the soft skills factors, instead on the general software engineering principles analysis.

One of the principles of agile software development [8] is “Responding to change over following a plan.

In order to adapt this principle one must plan to change. One useful artifact in this approach is the product backlog which basically shows the ordered list of requirements for the system or product being developed by business value for the company [9]. It is also adapted for change and evolves as the product and the environment in which it will be used evolve (management constantly changes it to identify what the product needs to be appropriate, competitive, and useful). This is why the presented approach focuses on dynamics nature of metric values.

The ERP system that is analyzed in this paper is Dynamics AX. The application layers of Dynamics AX are based on the architectural principle that enables fine settings and extending definitions of model elements (tables, classes). From the definitions of model elements, this is transferred to structure and behavior. Application layers are shown in Fig. 2.

When a new version of an application is published by the vendor (Microsoft) all the elements of the model, which define out-of-the-box application, are located on the lowest level in SYS layer shown in Fig. 2.

The partner modifications are usually in the middle layers (BUS and VAR layers in Fig. 2), and the end-user modifications are in the top layers (USR and USP in Fig. 2). USP is a patch layer. The patch layers exist for most of the layers, but for brevity are not shown in Fig. 2. They are designed to make incorporating updates easier.

If a modification of a model element is made, then the definition from the lower level is copied to the current level and the modifications are saved as a model element at the current layer (over-layering). The run-time combines the definitions of the elements considering the top definitions first. The object created by the run-time, can be considered an occurrence of a dynamic type definition composed from the model elements from several layers. The object instances are created either on the server or on the client, depending on the element definition.

### III. METRICS FOR ERP SYSTEMS

Metrics that we have identified as ERP specific are based on technical notable technical aspects of ERP system such as:

1) Number of model elements of different type
2) Number of model elements by layers
3) Number of model elements by modules/business areas
4) Usage frequency

In the following sections we give a more detailed view of these aspects through a detailed analysis of each of this aspects with the sample data shown for a retail/wholesales company operating in several countries and a brief analysis of classical software metrics application with ERP systems will be presented.

#### A. Number of model elements of different type

First technical aspect is easiest to understand. It gives us a rough idea about the size of implementation. Table I shows numbers of elements by few different element types in the sample. The first column shows general record type, since some types were grouped for brevity. Second column is the number of elements of this type.

<table>
<thead>
<tr>
<th>General Record Type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Method Element</td>
<td>122729</td>
</tr>
<tr>
<td>Table Field Element</td>
<td>59393</td>
</tr>
<tr>
<td>Table Method Element</td>
<td>22448</td>
</tr>
<tr>
<td>Class</td>
<td>7871</td>
</tr>
<tr>
<td>Extended Type</td>
<td>7250</td>
</tr>
<tr>
<td>Menu Element</td>
<td>6280</td>
</tr>
<tr>
<td>Table Element</td>
<td>4795</td>
</tr>
</tbody>
</table>

Class method elements include instance and static methods of classes. This is the most common element type. Usually they are created and modified when there is a process change. In the second row there are different kinds of fields of tables.
(fields, field groups, indexes, foreign key relations, etc.), which are usually created and modified to support data model change that is in connection with process change. The third row shows methods of tables which are almost the same as methods of classes from first row. In fourth row it is shown that there re almost 8 thousand classes. Extended data type in fifth row is kind of lookup mechanism. In sixth row are different kinds of elements related to menus. The next row shows that there are almost 5 thousand tables in this sample. Other element types have not been shown for brevity.

Tracking the dynamics of values of this metric over time doesn’t provide any special value, since this is introductory metric.

B. Number of model elements by layer

Most of the existing research (e.g. [6][10]) suggests the importance of avoiding customizations as much as possible. Our goal is to identify where the customizations are and to see how we can plan for change.

In order to do so, an aspect which focuses on identifying model elements that have the most customizations is introduced. Based of layer location of specific model element (see Fig. 2), we can see if the element is vendor original (SYS layer) or partner (VAR) or end-user (USR and USP layers).

Table I has shown different model element types. Here their distribution over different application layers is analyzed in order to see layer location of customizations.

Fig. 3 shows different model element types and their distribution over layers. The number of elements for each type is shown as vertical axis. Types with most elements in total are shown as the ones on top of Fig. 3 (i.e. Class methods). The layers are shown as horizontal axis and have been grouped and sorted in hierarchical order, as used by the application layer system mentioned before. For brevity reasons, some element types which do not have customizations are not shown.

In the first row, there are the tables that don't have relevant attributes that determine business area properly set up. A company can try to remedy this on appropriate application layers where it has access (e.g. end-users at USR/USP layers). Most of these tables are in vendor (SYS) layer. Second row contains tables which are in basic area which supports all other areas.

The third row contains tables related to ledger (general ledger) area, which form a core of financial transactions module. Fourth row contains inventory related tables which form core of inventory management or SCM (Supply Chain Management) module. Next row contains tables related to Human Resource Management (HRM).

One very valuable insight comes from noting the ratio of number of tables in certain business area across different application layers. E.g. it can be seen that the total number of tables in the Ledger area, which form a core of financial transactions module. Fourth row contains inventory related tables which form core of inventory management or SCM (Supply Chain Management) module. Next row contains tables related to Human Resource Management (HRM).
tables in end-user layers (USR/USP) and partner layers (BUS/BAR) in customer area (or Accounts Receivable) and vendor area (or Accounts Payable) is highest compared to number of tables for this area in vendor (SYS) layer. This indicates that the end-user company in this sample had most modifications in these modules. Other notable business area that was heavily customized is inventory management.

Monitoring dynamics of this metric values provides very valuable information to the company. A new developer can come and based on finding from this aspect have the idea about how the ERP system developed over time.

D. Usage frequency

This aspect identifies whether a part of the system has been used and how often it has been used.

For this purpose the original tool was modified to perform the analysis of the whole ERP system including all layers.

Table III shows an analysis of a number of forms and their run count since ERP was installed and the layer where this form was last modified. Only a few sample rows have been listed, alphabetically ordered, save for last few rows which show forms without usage.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Usage Count</th>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>Address</td>
<td>2275</td>
<td>USP</td>
</tr>
<tr>
<td>Form</td>
<td>AddressCountryRegion</td>
<td>74</td>
<td>SYS</td>
</tr>
<tr>
<td>Form</td>
<td>AddressCountryRegionGroupBLWI</td>
<td>0</td>
<td>SYS</td>
</tr>
<tr>
<td>Form</td>
<td>AddressSelect</td>
<td>2</td>
<td>BUS</td>
</tr>
<tr>
<td>Form</td>
<td>CustTripJour</td>
<td>0</td>
<td>USP</td>
</tr>
<tr>
<td>Form</td>
<td>CustTripJourTable</td>
<td>0</td>
<td>USP</td>
</tr>
<tr>
<td>Form</td>
<td>CustVendCreditInvoicingLookup</td>
<td>0</td>
<td>SYS</td>
</tr>
</tbody>
</table>

Elements that have no usage (i.e. where Usage Count is zero) and which are created at appropriate application layers (e.g. USP for end-users) can probably be safely deleted, and will remove the amount of customizations and make upgrade process and maintenance easier. In Table III this would be the forms CustTripJour and CustTripJourTable.

There are situations where there are elements that are from the lower layers (SYS) and which are not used (in Table III this would be the forms from last two rows). This indicates that there are some out-of-the-box functionalities of ERP system that are not needed. By identifying the parts of ERP system which are not used better view for future development and maintenance of ERP system is achieved. Big saving on upgrade effort is created by not analyzing these parts of the ERP system during the upgrade process. Other way for the company to leverage this information is in vendor/partner negotiations; dropping modules which are not used and appropriate licenses reduces upgrade costs. Module licenses which are not used can even be traded for other modules with some vendors.

This metric has dynamic values which can be analyzed in two ways. First if there is a prediction that usage of some elements is going down fast, this is a signal that there is a problem with its functioning (e.g. bugs, bad performance) or that company is abandoning the business approach that relied on this elements. Second signal is usage increase. This is a reason to pay special attention to model elements in question and improve their performance if needed.

E. Classical software metrics and best practices

Software metrics are used to measure relevant attributes of a software or its parts (i.e. code and tests).

In OO languages, a special attention should be given to object-oriented metrics which are focused on objects, their relations and properties. There is a standard set of object-oriented metrics which were proposed in [11], later implemented in many metrics tools (e.g. Weighted Methods Per Class, Depth of Inheritance Tree, Lack of Cohesion in Methods, Coupling between object classes, etc.).

There are also metrics which are not focused on objects, such as: number of source lines source of code, comment percentage, cyclomatic complexity, etc.

Most of this metrics are supported in main-stream programming languages, and can be implemented in ERP systems also.

Best practices are a form of static code analysis, which is performed without executing the programs (in most cases the analysis is performed on source code). This kind of analysis enables early detection of potential problems during the creation of code. Simply by focusing on correcting defects earlier rather than later in a project, one can cut development costs and schedules by factors of two or more [12].

Some common static analysis tools are: FindBugs for Java, FxFinder for .NET, PReFex and PReFast for C and C++. The tool for checking the best practices in Dynamics AX is an integrated version of a static code analysis tool and is embedded in the compiler. If set in a proper manner, violations of best practices will block working with the code and the model elements. These checks are based on the cumulative development experience of an ERP system, which in turn are based on general good software development guidelines.

The integrated tool allows running an analysis of code and application model elements to check for best practices violations. This tool includes about 350 rules [13] in many different categories. The authors have modified this tool in order to support the analysis of a large number of model elements at once of different types and layers.

Classical software metrics and best practices violations give a very important aspect of the system that can be dynamically monitored in ERP context too. Both of them should provide indication if their values are going out of recommended ranges, and can become potential bifurcation points.

One specific for best practices violations is if there is constant increasing trend of same type of violation, that there is a lack of education of developers in this field or that there is a need for systematic solution (e.g. architectural change) for this type of violation.

IV. DISCUSSION

The previous analysis (types, layers, modules/business
areas, usage frequency, best practices, software metrics) can present a clear picture with a quantitative backing about the weak points of the software system. For most aspects, investing in correcting all this points could take a lot of time. The presented approach is an agile one. The request priority in the product backlog indicates the priority of improving the previously discovered weak points.

For example, if the next iteration in software development includes working in business areas like HR, we would try to improve some weak points in this area. We would use the modified tools we have created to analyze the whole module for any weak points in form of the best practices violations, bad software metrics, usage frequency, distribution of elements by layers, etc. This approach can provide a good overview of the current state of a part of an ERP system rather fast. The efforts needed to fix the identified weak points can be managed in a better way, and can even influence top requests in backlog in regard to their priority. Making improvements to class hierarchy to remedy this some of problematic situations could take a long time, so it might make sense to leave them 'as is', until we have a request to make modifications to the part of the ERP system that these classes are related to.

Continuous improvement by using the above approach would create long term advantage for upgrade of an ERP system. It would also influence the team to raise their awareness of good software development practices which are directly connected with quantified results from tools that they can use easily, so they would strive to create high quality software solutions that would reduce their work in maintenance.

Another benefit is that the analysis of different aspects can be run on an ERP system or on parts of the ERP system maintained or developed by partner/vendor companies. We could define minimal software engineering acceptance criteria that a partner would need to reach in order for us to accept their solution. In long term this transfer of good practices downwards (in application layer sense) to partners and to vendors would help the company get future solutions which are more just-in-time like (with less time, less money paid and with higher quality) [14].

Doing an upgrade is usually considered to cost 25-33% of the initial ERP implementation investment [15]. Skipping one whole version can be considered an even worthier move that required greater analysis and planning. Using this approach has helped us perform upgrade from version 3 directly to version 5 (Dynamics AX 2009) of the ERP system that we are using.

As the software evolves, the analysis that was presented in this paper helps get benefits and improve the quality of ERP system even after initial analysis which was performed at previous periods. It also greatly helped in motivation of developers, by giving them a very productive environment where they can easily implement most of requests presented by management.

V. CONCLUSIONS

In the recent years there have been many analytic perspectives on ERP systems, most of them focusing on soft skills. The significance of this analysis derives from its different point of view - from technical perspective - and extends to connecting valuable and precisely quantifiable information with an agile development approach of ERP systems, mostly in response to the increased focus to development and maintenance of current ERP system determined by the economical breakdown. In difficult times of a global financial crisis this approach has proven as the most productive and extremely motivating for developers.

Technical perspective was introduced through different technical aspects. Using data from this analysis and benefiting from an agile approach has resulted in an exceedingly productive way of maintenance and implementation of new requests. It also enabled easier ERP system upgrade resulting in a worthy upgrade to second next version (from version 3 to version 5).

The challenge is to continue to develop and maintain ERP system using technical perspective for agile and productive approach in the years yet to come.

REFERENCES