2022 2nd International Conference on Intelligent Technologies (CONIT)
Karnataka, India. June 24-26, 2022

A Novel Approach to Reuse Custom Enterprise Software Extensions for Cloud Migration

Garima Srivastava
SAP Labs India Pvt. Ltd.
garima.srivastava@sap.com

***Abstract* - Enterprise Software has been the backbone for running businesses since last many decades now. It's a common practice for the consumers to enhance these software as per their specific business needs. In general software development terminology, such enhancements are called custom extensions of the software. Depending upon the specific business needs, the magnitude of custom extension may vary from customer to customer.**

With the advent of cloud computing and digital transformation, organizations are paving their path towards cloud in all sectors. This is also applicable for organizations developing and consuming enterprise software. Although cloud computing brings a lot of value to the software eco-system, however it also presents a challenge towards disruption in businesses where the existing custom extensions cannot be reused while moving to cloud. In this paper, we present a novel approach to automatically enable custom extensions from an old On-premise installed Enterprise software to new version of the software hosted on a Cloud Platform.

Keywords – enterprise software, migration, cloud computing, software extensions, software enhancements

I. INTRODUCTION

A. Enterprise Software and relevance of Extensions

Conventionally, organizations have been using on-premise enterprise software [1], which are installed and run-on computers or servers on the premises of the organization. With on-premise enterprise software, the software is downloaded or installed directly on the computers or servers that are executing the software. An organization may commonly take a standard vanilla enterprise software from an application provider and enhance the same with custom extension/add-on code ("extension/add-ons") to tailor it to the organization's needs [1] [2].

As a reference business example, take various countries and organizations in the oil & gas, chemical or mining industries, that may work with hydrocarbon products. Hydrocarbon products generally expand or compress with variation of temperature, pressure, density, etc. and it is desirable for these industries to perform their business according to standards governed in their country or according to standards defined globally. For example, diesel loaded in a container in Germany at L15 (Liters at 15 degrees Celsius) may be unloaded in France at L30 (Liters at 30 degrees Celsius) or unloaded at a non-standard conversion of BBL (Billion Barrels). When such products are moved from one place to another, volumes differ and hence cost calculation may occur at the unit being used at the destination, which may vary from the unit at the source. So, for organizations dealing with hydrocarbon products, considering these factors to calculate exact volume may be very important. Across the globe, there are several regulators (e.g., American Petroleum Institute [5]) who have already defined these conversion standards, while some industries have defined their own conversion routines. These organizations may use a standard software application for tracking the products. However, as performing the conversion of the hydrocarbon product (on the basis of temperature, pressure, etc.) may be legally binding and requires an in-depth knowledge of the domain, the organization may want to tailor the standard software application to execute the conversion. To that end, the organization may build an extension for the standard software application to perform this conversion.

In this process, organizations make a significant investment with the creation and operation of these extensions/add-ons, as they include hundreds of lines of code that might be written in a particular format and/or language to be operable with the vanilla version upon execution [4].

B. Transformation to Cloud

In recent history there has been an effort for organizations to move their computing to cloud platforms, using a Software as a Service (SaaS) model in which software is licensed on a subscription basis and is centrally hosted [6]. When an organization wants to move their existing on-premise software to a cloud platform, the extensions/add-ons may not be directly transferrable [3].

For example:

- different software may be used in the cloud versus on-premise applications
- the data which the on-premise application uses may not be available on the cloud
- the development paradigm used to create extensions/add-ons on on-premise might be very different on development for cloud.

The creation of these extensions requires an investment of time and finances, when an organization moves their computing to a cloud platform.

Hence, systems and methods are desired which support an intelligent migration of on-premise applications having custom extensions/add-ons to a cloud platform.

2022 International Conference on Intelligent Technologies (CONIT) | 978-1-6654-8407-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CONIT55038.2022.9848049

II. RELATED WORK

In the last decade, a lot of research has been carried out to analyze the enterprise solution architecture. Additionally, there have been specific studies that have analyzed the reuse of codebase of these enterprise solutions. According to our literature survey, the following research works are most relevant in regard to our proposed framework:

[1] is a review of all the published work on Enterprise resource planning (ERP) until 2010. It provides a good summary of literature on ERP and gives reference to critical concepts like importance of ERP solutions, details of implementations, exploration uses, and the significance of extension needs. It explains why some companies have the need to extend the ERP solutions.

[3] defines extension points as a placeholder where customization of framework can be done. It explains how passing a framework related object, as an argument in an Application Programming Interface (API) call can be considered as customizing a framework. This study proposes a graph mining approach for managing the extension points. Although this study does not cater to enterprise software directly, however it gives good insights on process of identification of extension points.

[2] enhances the concept of identification of extension points by introducing a framework known as FEMIR ((Framework Extension Point Miner and Recommender). This framework executes static analysis of the source code for identification of extension points. It uses code completion infrastructure of the Eclipse integrated development environment (IDE), for publishing information about identified extension points.

III. THE CUSTOM CODE CLOUD ENABLER (C3E) FRAMEWORK

In this paper we present a framework called Custom Code Cloud Enabler (also known as "C3E") that identifies custom extension code, that has not been enabled on a cloud platform during migration of its associated on-premise application to the cloud platform. Once C3E identifies the custom extension code, it generates proposed adjustments for the custom extension code to successfully execute on the cloud platform. The adjustments include C3E framework's search for corresponding cloud extension points that may be used with the custom extension code so that it can be executed on the cloud platform.

A. High Level System Architecture for C3E Framework

C3E framework identifies custom extension code related to a software development extension, that has not been enabled on a cloud platform (and may be considered an "error") during migration of its associated on-premise application to the cloud platform. Once the C3E framework identifies the extension code, it generates proposed adjustments for the code to successfully execute on the cloud platform. The adjustments include the C3E module's search for corresponding Cloud Extension Point or an API (Application Programming Interface) that may be used with the custom extension code so that it may execute on the cloud platform.

In addition to identifying specific errors, the C3E framework identifies the number of issues or errors in the extension code once the code is migrated on the cloud platform. This identification may provide a collected view of the overall issues presented with the migration. The framework also classifies the errors encountered in the migration of the extension code. This classification may be used for upskilling developers, or to gauge effort expenditures to address the errors. Regarding the upskilling, for example, consider a scenario where there were dynamic programming calls in the extension code made which now need to be converted to Cloud API based calling mechanism. Knowing how many such instances need to be corrected or re-implemented may help plan for Cloud API upskilling.

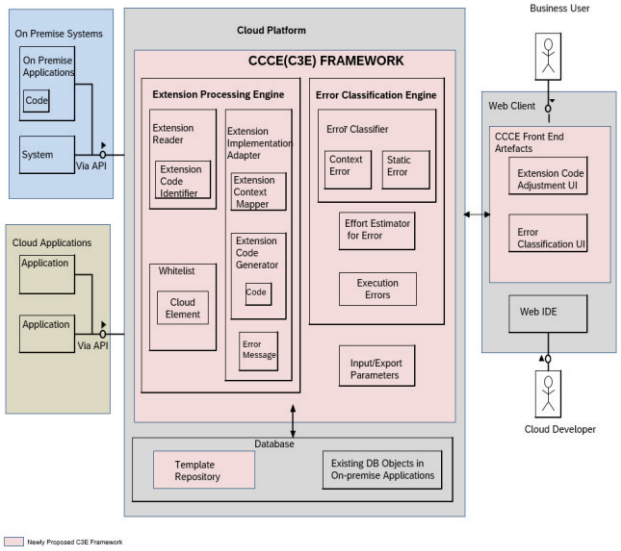

Fig. 1. C3E Solution Architecture Depiction

As shown in Figure 1, The overall solution architecture depicts a client/end user, a web client including a Web Integrated Development Environment (IDE) editor, a front-end artifact of an error classification user interface (UI) and an extension adjustment user interface (UI), a cloud platform, a custom code cloud enabler(C3E) framework, an on-premise system and cloud applications. The architecture includes a depiction of cloud platform. The cloud platform provides any suitable interfaces through which clients (e.g., web client, end user via web client, on-premise system and cloud applications) can communicate with the C3E framework.

1) C3E UI Artefacts:

The Error Classification UI requests and receives data from an Error Classification Engine in a backend of the C3E module. The Error Classification Engine is responsible for classifying all the errors via an error classifier in the extension code with respect to severity. For instance, for some errors, depending on the severity and amount of effort in adjustment of the extension code generated by an effort estimator, a High, Medium or a Low Classifier may be assigned to the extension code.

2) C3E Backend Extension Processing Engine:

An Extension Processing Engine identifies the custom extension code that needs to be adopted to be enabled on the cloud platform, as described further below. The Extension Processing Engine includes an extension reader (including an extension code identifier), an extension implementation adapter (including an extension context mapper and an extension code generator).

The extension reader identifies the places in the software that have custom add-ons/extensions. The extension implementation adapter analyzes the software objects identified by extension reader. In this analysis, the context mapper tries to map the point/hook where the custom extension was built (in the on-premise version) to the available point/hook in the cloud software stack (commonly referred as whitelist [8]). The extension code generator may generate the custom add-on/extension code for the hook identified by extension context mapper.

3) C3E Database Artefacts:

Database stores an extension definition template repository and existing DDIC Objects. These are Data Dictionary Objects (Tables that are created by software provider and Custom extensions created on these tables by user) which are used by the C3E module during the execution.

B. Deployment Options for C3E

The proposed C3E framework can be deployed in at least one of two manners. As a first deployment manner, it can be deployed as a SaaS offering so that it may be easily integrated with cloud-based applications and be consumed on an on-premise stack (e.g., Enterprise Resource Planning Systems installed on computers on the premises of an organization). With this deployment manner, the extension code may be adjusted by C3E once the extension code is installed on the cloud platform.

As an optional deployment manner, the C3E framework can be used as a built-in solution with the cloud platform, where cloud relevant objects from the cloud platform may be used to create/deploy the extensions. With the second deployment manner, the framework can be considered to be provisioned with the cloud platform. In this case, when the user installs his extension code on the cloud platform, user can use this framework to adopt the extension. It needs to be considered that the first deployment option may be preferrable in a case where the user wants to prepare his extensions in the on-premise stack and the second option may be preferable in a case the user may be ready to completely install his software from on-premise hosting to the cloud platform.

C. Details of Extension Concept and its definition in Software Design:

Figure. 2 illustrates how an enterprise software may be designed with a Super Package including two Sub Packages, where each Sub Package includes a plurality of Objects. Here the packages may be entities formed from a collection of one or more objects [13]. The objects may be a software artifact (e.g., a Class, or a Method in a class). It should be noted that each package can have a hierarchy of its own (e.g., packages within packages, etc.).

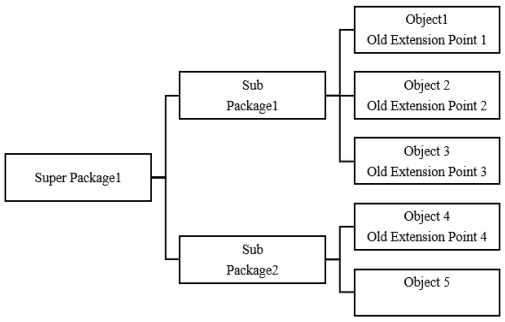

Fig. 2. Depiction of Packages & Extension Points

Figure 3 depicts how the extension points can be utilized to create custom code extension code for enhancing a software [3].

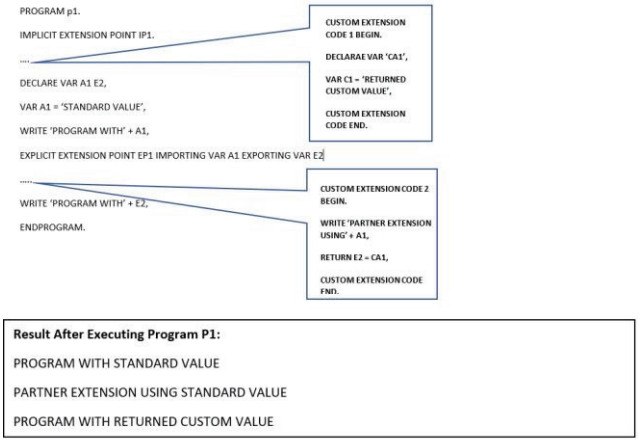

Fig. 3. Depiction of Usage of Extension Points

It illustrates a standard program (PROGRAM p1) having a custom extension code created at both an explicit as well as implicit extension points. With respect to the implicit extension, the extension code cannot be input anywhere in the on-premise application. Rather, the on-premise application includes limited areas where custom extension code may be included (e.g., a beginning or end of a program).

With an implicit extension point, there may not be specifically defined import/export parameters, unlike an explicit extension point that does include specifically defined import/export parameters. Explicit extension points also give the benefit of defining custom extensions at desired places within the provided software.

D. Process Flow for C3E Execution:

Figure 4 illustrates the process flow of C3E execution at runtime. The process initiates by receiving application code that needs to be ported to cloud platform.

Initially, a user initiates an extension processing engine. Initiation can be done via selection of an extension

adjustment selector on an extension adjustment user interface.

Next, application code of the on-premise application is received at C3E module. Then, a whitelist of cloud elements is provided by the Cloud Enabler Module. The cloud elements on the whitelist, by virtue of being included on the whitelist, should have been approved by the cloud platform as being operable with the cloud platform. The cloud elements can be at least one of an Application Programming Interface (API) to integrate the on-premise application code with the cloud platform without the need for direct function calls or a cloud-provided source code plug-in to enhance existing application code.

Further, an extension reader of the C3E module analyzes the application code to identify any extension points via an extension code identifier. The extension reader identifies the extension points object-by-object and performs the analysis as for the object. As described above, the extension point may be one of an explicit extension point or an explicit extension.

The C3E Module then determines for each identified extension point, whether a suitable cloud element exists on the whitelist that may implement the extension code.

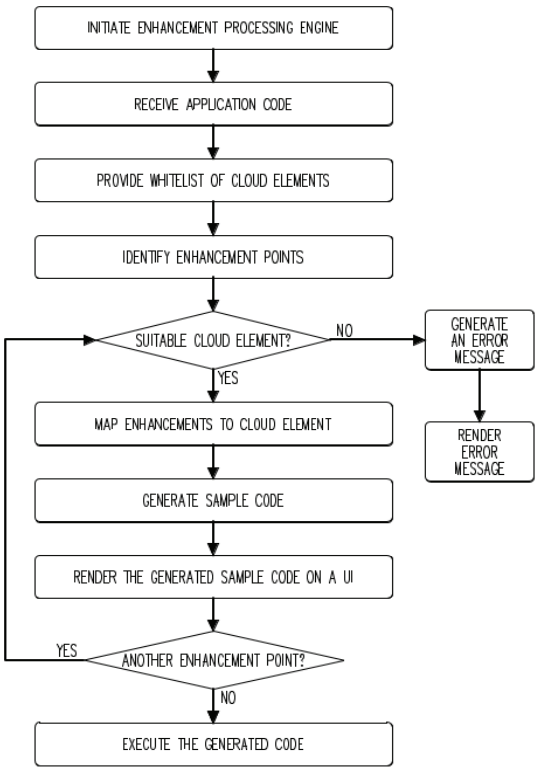

Fig. 4. Process Flow for C3E Framework

Additionally, to determine a suitable cloud element exists, the C3E Module selects a first cloud element from the whitelist, reads a structure definition (import/export parameters in the case of an explicit extension or lack of import/export parameters in the case of an implicit extension point) of the extension code in the extension point, and compares it to the structure definition of the selected cloud element.

As implicit extension points do not have parameters and are available either at the beginning or at the end of an object, the user can write any code that he wants to execute before/after the standard code is run. In some instances, the whitelist can include one or more generic elements that might be used as a generic hook (i.e., having no mandatory import/export parameters) for cloud extensions. Hence mapping of an implicit extension point to a cloud element on the whitelist can be a mapping of the implicit extension point to a cloud element on the whitelist that does not have explicit import/export parameters. For example, if there is an extension implemented using an implicit extension point at the beginning of the object (e.g., class, etc.), it may be mapped to the cloud element on the whitelist that does not have explicit parameters defined in the cloud code and is also called in the beginning of the object.

The C3E Module selects a given cloud element from the whitelist sequentially (e.g., it may select the first-listed cloud element from the whitelist first for comparison, and if needed, it may then select the second-listed cloud element from the whitelist). In a case the structure definitions match, the cloud element is a suitable cloud element for the extension code at the identified extension point. It is to be noted that with respect to the import/export parameters of the structure definition, for there to be a "match", at least the mandatory parameters are the same (e.g., the optional parameters do not have to match).

As depicted in Figure 5, the C3E Module compares the structure definition of the explicit extension point of an object 1 to the structure definition of API 1 in the Whitelist.

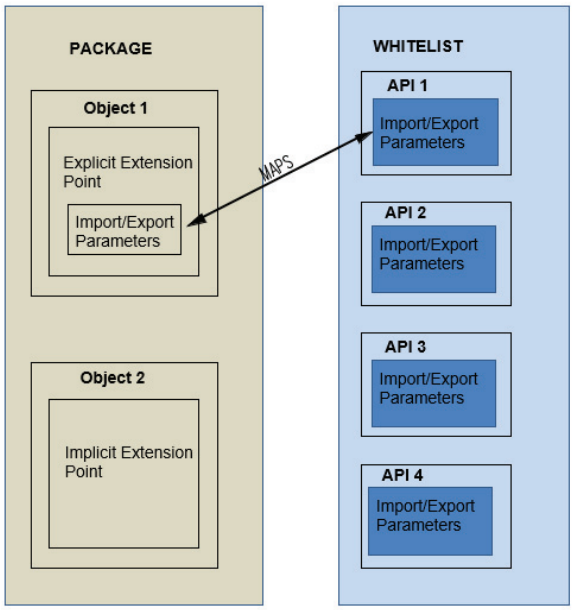

Fig. 5. Mapping of Object to Cloud relevant artefact

In a case the structure definitions do not match, or the match is absent, the C3E Module selects a second cloud element from the whitelist for comparison. The C3E Module may continue to check the cloud elements in the whitelist until at least one of a suitable cloud element is located, all of the cloud elements in the whitelist are selected, a pre-set amount of time has elapsed, a pre-set amount of cloud elements have been selected, or any other suitable stop point.

IV. EVALUATION STRATEGY

The aim of this study is to come up with a model to identify contextual extension code related to a custom software development, that has not been enabled on a cloud platform. It then generates proposed adjustments for the extension code to successfully execute on the cloud platform.

For the purpose of evaluation of the model, we have computed the magnitude of custom extension adjustments done via the proposed model and compared it to the adjustments done manually for three software applications to be enabled on Cloud Platform. For the purpose of this study, we have considered Google Cloud Platform (GCP) as the platform where the applications need to be migrated.

To estimate the efficiency of the model, we calculate the variance between values from the model and manual adjustments for the following factors:

- Total Time taken for adjustments
- Number of Places the custom code was adjusted by the model

A. Evaluation Metrics and Analysis Procedure

The data received from the model was considered for the above two factors for each software application and the variance from manual adjustments were determined. Manual efforts for adjusting the custom extensions were calculated for this case study by observing the time recorded for software adjustments by employees working in the upgrade project from On-premise software to cloud platform. The development tasks made use of the project management software JIRA for recording times and tasks. With the help of JIRA, developers can account the actual time taken in a project for adjusting the custom extensions. As mentioned in [11], the project data is stored in the JIRA repository which could store information of numerous ongoing projects within an organization.

V. EVALUATION RESULTS

A. Cases Considered

The cases considered for this study were three different software applications in the areas of Procurement, Manufacturing and HR where we were involved as part of the application team that manually adjusted the custom code for these applications. We approached multiple developers to calculate aggregated time required for manually adjusting each software application.

Table 1 shows results of the C3E model being applied to the custom extensions that were required to be adjusted for transition to cloud platform. After executing the proposed framework, we arrived at the estimated efforts for each application shown in the below table.

TABLE I. ESTIMATES OF ADJUSTMENTS VIA C3E MODEL

Upgrading Application	Custom Code Adjustment Efforts (Hours)	Number of Places Adjusted in the software
Procurement	60	8
Manufacturing	78	12
HR	92	19

Table 2 shows results calculated by the manual adjustments by using the data from JIRA and inputs from the developers working on the applications to be upgraded to cloud platform.

TABLE II. ESTIMATES OF ADJUSTMENTS VIA MANUAL PROCESS

Upgrading Application	Custom Code Adjustment Efforts (Hours)	Number of Places Adjusted in the software
Procurement	118	8
Manufacturing	123	13
HR	189	21

Table 3 shows the calculated and manual actual efforts incurred (in Hours) in our case study along with the variation

TABLE III. EFFICIENCY IMPROVEMENT IN TIME TAKEN FOR ADJUSTMENTS

Upgrading Application	Efforts using Manual Process (in Hours)	Efforts using C3E Model (In Hours)	Efficiency Improvement
Procurement	118	60	196.67%
Manufacturing	123	78	157.70%
HR	189	92	205.43%

Table 4 shows the number of places where the custom code was adjusted using the C3E model as well as manual adjustments. It depicts the accuracy of identification of the custom code adjustment done via the proposed model.

TABLE IV. ACCURACY IN CUSTOM CODE ADJUSTMENT

Upgrading Application	No. of places adjusted manually	No. of places adjusted by C3E Model	Accuracy in %
Procurement	8	8	100%
Manufacturing	13	12	92.30%
HR	21	19	90.47%

VI. OUTCOMES/CONCLUSION

In this paper, we have proposed a model that would automatically enable custom extensions from an old On-premise installed Enterprise software to new version of the software hosted on a Cloud Platform.

To compute the efficiency and accuracy of the proposed framework, an empirical study was conducted analyzing three applications that were considered to be enabled on a cloud platform. The three applications were in the areas of procurement, manufacturing and human resources (HR). The factors for evaluation were the time taken to do the custom adjustments and the number of places where the code base was adjusted. The proposed custom code adjustment framework was helpful to enable enterprise software applications efficiently on cloud platforms by reusing custom code created as extensions.

REFERENCES

[1] Addo-Tenkorang, Richard, and Petri Helo. "Enterprise resource planning (ERP): A review literature report." Proceedings of the World Congress on Engineering and Computer Science. Vol. 2. 2011

[2] Asaduzzaman, Muhammad, et al. "FEMIR: A tool for recommending framework extension examples." 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017

[3] Asaduzzaman, Muhammad, et al. "Recommending framework extension examples." 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2017.

[4] Haines, Marc N. "Understanding enterprise system customization: An exploration of implementation realities and the key influence factors." Information Systems Management 26.2 (2009): 182-198.

[5] Shepard, Francis Parker, Fred B. Phleger, and Tjeerd Hendrik Van Andel, eds. Recent Sediments, Northwest Gulf of Mexico: A Symposium Summarizing the Results of Work Carried on in Project 51 of the American Petroleum Institute, 1951-1958. American Association of petroleum geologists, 1960. Model for Software Development", 2016 23rd Asia-Pacific Software Engineering Conference, 2016

[6] M. Hajjat et al., "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud", SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, pp. 243-254, 2010.

[7] Onur Demirörs, Neslihan Küçükate Ömüral, Exploring reuse levels in ERP projects in search of an effort estimation approach, 2018 44th Euromicro Conference on Software Engineering and Advanced Applications, 2018

[8] Pareek, Himanshu, Sandeep Romana, and P. R. L. Eswari. "Application whitelisting: approaches and challenges." International Journal of Computer Science, Engineering and Information Technology (IJCSEIT) 2.5 (2012): 13-18.

[9] Jahchan, Georges J. "application Whitelisting." Information Security Management Handbook 6 (2012): 193.

[10] J. Hizver and T. Chiueh, "Cloud-Based Application Whitelisting," 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp. 636-643

[11] Srivastava, Garima, Yeshwant More, and Jenifer Sam. "Effort Estimation Model for an Enterprise Software Upgrade." 2020 International Conference for Emerging Technology (INCET). IEEE, 2020.

[12] David L. Olson and Fan Zhao, "Critical Success Factors in ERP Projects", Research and Practical Issues of Enterprise Information Systems, eds. A. Min Tjoa, Li Xu, Sohail Chaudhry., (Boston: Springer), pp. 569-578.

[13] En.wikipedia.org. 2022. Package (UML) - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Package_(UML)> [Accessed 14 February 2022].

[14] Brehm, Lars, Armin Heinzl, and M. Lynne Markus. "Tailoring ERP systems: a spectrum of choices and their implications." Proceedings of the 34th annual Hawaii international conference on system sciences. IEEE, 2001.