Abstract—Replacing legacy systems with off-the-shelf enterprise resource planning (ERP) software involves complex requirements analysis. To minimize risk and enable smooth execution of subsequent software development phases during implementation, we propose an approach that uses bipartite graph(s) to formalize ERP requirements. An industrial case study suggests that this type of lightweight formalism can be useful for large-scale, volatile projects such as ERP implementations. Our model is applicable to activities like product line engineering, domain analysis, and test-driven development, and facilitates reusing knowledge throughout the development lifecycle. Results indicate that bipartite matching can be advantageous for new and existing ERP projects that involve upgrading, merging, or transitioning.

Index Terms—Requirements engineering, enterprise resource planning, formal methods.

I. INTRODUCTION

Research surveys have identified enterprise resource planning (ERP) implementations and upgrades as top IT priorities [1], [2]. These studies suggest that commercial off-the-shelf ERP systems are likely to survive the trend towards cloud and app-based software, e.g., FluidUI. Porting existing legacy software to a new system has been an important factor in the success of ERP implementations. Proper migration of both functionality and data is necessary to satisfy key stakeholders, which include clients, vendors, and third-parties. However, the high complexity of ERP makes developing these types of systems non-trivial.

Requirement engineering (RE) seeks to develop tools and techniques that support the definition of correct, complete, consistent, and unambiguous software specifications [3], [4]. RE is recognized as a vital aspect of software projects since the quality of requirement specifications impacts all phases of software development. Formalizing RE activities can help to identify requirement problems early, and avoid long delays, missed delivery deadlines, and budget overruns.

With the high complexity of ERP systems and the potential benefits of formal methods, one would expect that there would be many formal approaches to ERP requirements specification, development, and validation. However, formal modeling and validation of ERP systems continues to be underdeveloped in the research community [5]-[7]. Furthermore, there is little adoption of formal methods for ERP software development in industry.

In this paper we present a formal approach to product-line engineering for ERP systems to facilitate dynamic evolution and reuse across software projects. Our approach aims to guide requirement engineers in the specification, analysis, and estimation of the ERP customization effort. The major contributions of the paper are as follows:

- Describes an approach that uses bi-graphs to assist software engineers in mapping legacy functionality to ERP features,
- Presents a small case study on an event bidding system to demonstrate the feasibility of the formal specification and analysis approach,
- Discusses our experiences using the approach in an agile product-line development cycle in industry.

The rest of this paper is organized as follows: the next section contains the case study. Section VII provides related work, and Section VIII concludes the paper and discusses future work.

II. BACKGROUND

This section contains background material on enterprise resource planning, fit gap analysis, and bipartite matching as it related to requirements engineering.

A. Enterprise Resource Planning

ERP, a term coined by Gartner Research Group in 1992, is a type of Commercial-off-the-Shelf (COTS) software that seeks to provide an integrated optimized solution as a bundle for businesses to take care of functions including all internal and external operations necessary. The breadth of this definition encompasses the complexity of large-scale integrated systems and the cost associated with implementation and maintenance. ERP systems allow management through integration of all aspects of a business including production planning, purchasing, manufacturing, sales, distribution and accounting. ERP solutions allow the aforementioned operations to be performed cohesively [8], [9].

The ERP implementation process consists of the following phases: a fit and gap analysis (FGA) to identify business processes and customization requirements, which act as the
principal decision making phase for customization needs; system design and development; data conversions from legacy system; testing of the new system; training of the super-users and end users; and deployment. These activities may vary with the nature of the project based on if it is an upgrade or merger. ERP is generally distributed in two forms [10]: Generic or Preconfigured. Generic ERP software targets a variety of businesses and industries, and therefore must be configured early. Preconfigured ERP software is tailored to a specific industrial sector. However, both forms need customizations.

B. Fit Gap Analysis

During requirements elicitation, interviews with users during various phases of a project are a necessity. At project initiation, stakeholders are interviewed to determine scope, rationale and core gaps in requirements. During analysis, meeting users help discover and document requirements for enhancements, customizations and reengineering needs. Sessions with stakeholders throughout the testing phase helps to validate and verify that gaps have been closed. In agile development, these activities are iterative and incremental, and therefore fit-gap analysis (FGA) represents a plausible solution.

In FGA, we deal with two domains. The first is the domain that already exists, which is the source domain of the business needs and basic non-functional requirements. This is commonly termed as the legacy system. The second domain is the demonstrative ERP solution. The ERP demo environment is the target domain which will be customized and adjusted with required configuration. At the beginning, a high-end workable prototype is placed for analysis. Analysts can therefore choose to work on different modules of functionalities to: identify the fitting business functions, discard obsolete legacy features, suppress additional ERP features, and/or prescribe necessary customizations.

The principal goals of FGA include: creating a workable prototype of the target domain; assigning legacy experts and domain consultants for business areas; engaging associated technical experts of both legacy and target domain in analysis; documenting and specifying functional needs for each requirement; and iteratively refactoring specifications according to an agile development life-cycle. Non-functional requirements such as the minimum hardware and platform vary based on the vendor software and release version.

C. Bipartite Matching

Formal specifications use rigorous mathematical notations and techniques to precisely describe the properties a system should have, without unduly constraining how to achieve them [11]. In requirements specification, analysis, and design, formal methods can play a significant role towards success [12]. Using formalisms force the software engineer to ask questions that may have otherwise been postponed until implementation [11]. This allows early fault detection in documentation, and may be the distinguishing factor between a successful or failed project. Horowitz and Munson first stressed the need of formal approaches towards reuse [13] basing the content, concept and context of a domain.

In graph theory, a bipartite graph or bi-graph may be defined as a graph \( G = (U, V, E) \) where: \( U \) and \( V \) are disjoint sets whose elements make up the vertices of \( G \), and every edge \( e \in E \) connects a vertex in \( U \) to a vertex in \( V \) [14]. In other words, \( U \) and \( V \) are independent sets that partitions \( G \) into two. Applying this model to ERP requirements engineering involves matching functional requirements into the two aforementioned domains as relationships on a requirement-to-requirement basis. A one-to-one or many-to-one mapping may be used to associate legacy to ERP. One-to-many relationships can be ignored since it will be contradictory to both reusability and portability. In other words, the probable transfer will be shifted to a more robust and cohesive product than the one proposed if the latter is prominent. Thus legacy features are ensured of not being omitted or incorrectly mapped.

III. RESEARCH PROBLEM

Portability is special type of reuse, which can be determined from rigorous domain analysis [15]. More specifically, it is the ability to reuse some aspect of the software within different operating environments and contexts. For ERP implementation we redefine portability considering the product line engineering approach, combined with the agile manifesto [16].

Replacement of a legacy system with ERP software requires that at minimum the desirable qualities of the existing system be ported to the new implementation, both for system features and underlying data. If there are no significant improvements, the acceptability of the new ERP environment would be questionable. In the worst case scenario, the client could retract to the previous system in spite of heavy losses. Here we apply the two systems using bipartite graphs to determine matching functionalities through analysis.

A. Functional Challenges

Developers of ERP product line software are faced with the task of analyzing the portability of existing legacy functionalities, which have already met the users’ satisfaction criteria, and mapping them to corresponding ERP features that should also meet client needs and expectations. If the approach to tackling this problem is not exercised during initiation, monetary commitments may fall short or the project life cycle may cross the originally planned timeline.

Some aspects of the legacy functionality that may make attaining the same (or greater) level of satisfaction in ERP difficult can be if the desired feature is: operating smoothly; heavily cross-linked and distributed; flexible and extensible to cheaper COTS; 100% compatible and requires no/minimal support and maintenance; capable of achieving a very high level of user satisfaction. In the research problem under investigation, the targeted functionality is performing satisfactorily, and we address the question of how requirements engineering can be used to support porting this functionality to a new product line implementation while satisfying all business and data constraints.

B. Data Challenges

Another aspect of the research problem is determining how the variations of data in the current system will be represented
in ERP. ERP implementations already have their own underlying database, which is well-defined for most of the known modules of the target business function(s). Beside interconnection and embedded data operations fulfilling generic requirements, almost all implementation projects will end up specifying a handful of customizations that must be synchronized among the systems implemented. This increases complexity during version control, upgrades, new acquisitions and significant change management operations.

ERP systems contain two sets of data: 1) setup data, and 2) transactional data. Set-up data represents system constants and universal data, e.g., department identifiers or position numbers. Transactional data is the data that gets newly inserted, modified, discarded, batched and accessed on a daily basis for business operations, e.g., employee benefit premium, accounts receivables encumbrance value. Experience indicates that communication alone is insufficient for a precise gap estimate in the data requirements if representative or actual data is not utilized. Lastly, integrity constraints must be defined to ensure that there is no data loss or corruption, thereby allowing every piece of required information to be accurately transferred to the new system. For instance, if human resources already passed big bang adoption and financials is on transition, to receive a valid ledger for the business we will still need payroll and related information from the new system. Furthermore, the legacy system needs to be simultaneously operating to avoid errors in the budget year.

IV. ERP REQUIREMENTS AS BIGRAPHES

By mapping existing legacy functionality to ERP features, we can help make a decision upon the degree of customization with a better understanding of the two systems, and focus project efforts on correct construction as opposed to post-hoc analysis [12]. Questions that should be asked during modeling the graph representation are: 1) Does the proposed functionality already exist in whole or as a part of the legacy system, or is the functionality completely absent? and 2) If it exists as part of the legacy system, can it be extended for what is required and/or suppressed to disable unnecessary or unworthy features? Answering these questions can help to identify the basic elements and relationships that should be defined in the model. Similar questions can be enumerated for other scenarios such as upgrades.

![Bipartite Graph](image)

Let L represent the set of legacy features, and P be a set of ERP features (disjoint from L) available before customization.

It is possible to construct a bi-graph \( G = (L, P, E) \) where the elements of L and P make up the vertices of G, and every edge \( e \in E \) connects a vertex in L to a vertex in P. Our formalization provides a means of analyzing the similarities and dissimilarities between the functionalities of the two systems (Fig. 1). Some elements of L will have related elements in P, others will not. If the relation R describes the mapping between L and P, then \( L \setminus \{l \in L \mid R(l) \neq \Phi \} \) yields the features that need to be developed for P.

Initially, the relationships between elements of L and those in P can be established at a high-level of abstraction, i.e., mapping high-level features from legacy to ERP. Next we refine to analyze sub-feature relationships with use-cases. If requirements for both systems have been specified as use cases, we can first identify relationships between each legacy use case and that of ERP. Analysis of relationships between use case flows (discrete interactions) can help decide whether the feature is fully or only partially covered. Once the commonalities are precisely determined in between the two domains the non-cliquing vertices needs to be addressed which can be termed as non-portable functionalities.

The ideal situation would be to have a balanced bi-graph where \( |L| = |P| \). This would mean that the existing features could be completely ported to the new environment without having to perform any customizations. However, in most cases the client would be requesting more features and thus we expect to have \( |L| > |P| \). It will help the client determine the cost-benefit ratio of the project, and select among the large number of functionalities typically provided as add-on modules. Our problem is now reduced to listing the bi-cliques by associating features with left and right-hand side vertices, respectively, and then enumerating all bi-cliques with all legacy vertices and at least \( \alpha_m \) (the minimum set of) right-side vertices. A vertex cover of G is a subset of vertices \( C \subseteq V \) such that for each \( (u, v) \in E, u \in C \) or \( v \in C \) holds. The complement set \( V \setminus C \) is an independent set in G. The graph obtained by removing a vertex \( v \) from G and all its adjacent edges is denoted as \( G \setminus v \) [17]. For a decision to suppress feature or reduce the scope for monetary or security grounds, we use the covering with vertex removal.

Exterior covering further allows decomposition of a bipartite graph into a core part and an inadmissible part [18]. The core of the decomposition consists of a number of irreducible sub-graphs and at most two tails. Each of the tails is the union of a number of minimal semi-irreducible sub-graphs. The decomposition has no tails if and only if G is an \( n \times n \) bipartite graph and has a transversal of order n [14]. The core part can then be further decomposed into irreducible parts to produce a canonical reduction of the graph. Applying exterior covering to the matching problem, we introduce edge labels that associate the relationships from L to P with values. An edge \( e \) mapping a legacy feature l to an ERP feature p (denoted \( e_{lp} \)) has the value 1 if the functionality of l matches or exceeds p, otherwise \( e_{lp} = 0 \). The exterior cover then is defined using a pair of sets \( [A, B] \), where A includes all the features of L, and B includes all the features of P, at any point of operation. In graph theory, \([A, B]\) are referred to as the minimal exterior pair (MEP) [14].

An exterior cover during ERP project initiation can be useful for certain critical decisions which may impact the
overall success: 1) identifying ambiguity in requirements which can hinder the establishment of co-relationships; 2) assessing the degree of similarity or dissimilarity between the two systems; 3) evaluating additional properties of the problem using bipartite theories of canonical decomposition, 4) help domain analysts to make decision on non-enhancing possible portability features to reduce dynamic evolution of variable artifacts, 5) allow exact customization information being available at any point of implementation life cycle.

V. DECISION ON ENHANCEMENT

In our discussion let K be a subset of L×P, representing the cartesian product of the related functionalities (edges) from each system, corresponding bi-graph representation thus would be of finite exterior dimension. Since the legacy system exists and is functioning, we know that a number of requirements for it have already been specified. In addition, the practical norm of software development is that customers are delivered a contracted number of features per release, and evolution of the product is controlled through versioning. We can therefore conclude that K is a matrix (which is also a forest for very large scenarios) where both L and P have a finite or countable number of elements ordered l_1, l_2, l_3, ..., and p_1, p_2, p_3, ... and can be represented a_i j = 1 if the pair (l_i, p_j) is an element of K, otherwise a_i j = 0. An element in K is said to be inadmissible if it is not an edge of any disjoint sub-graph K*, such that E(K*) = E(K); other-wise the edge is admissible. For our problem, K is of finite exterior dimension, the subset K_i consists of all admissible edges of K, which is referred to as the core. K can be decomposed into 3 regions, R_1, R_2, R_3 by canonical decomposition [18].

In Fig. 2, R_2 is shown in the upper left-hand corner, and R_3 in the lower right. R_1 is the region separating R_2 and R_3 and will be discussed with cover properties. Alterations to K, and the role played by the core K_i, is our tool for identifying mismatching functionalities (inadmissible edges), and help forecast probable success or failure of our project. Following cover properties were identified:

1) If K is altered by the addition or removal of edges from R_2 the resulting graph has the same core as does K. Each of R_1, R_2 and R_3 remain unaltered indicating features in legacy that have equivalent features in ERP. A scenario with such a property contributes to a fit. However, all of these features may not be needed by the client so should be limited to the core requirements. As mentioned earlier removal from R_3 will hold the property G\{v\}.

2) If edges in the region R_1 are added to K, the resulting graph will have the same decomposition of L×P as does K, and hence each added element is admissible. The addition of edges from this region will continue to reduce the gap between the two systems, without affecting project parameters such as development, testing and maintenance. More edges from this region means increased portability and flexibility to change. In our case study this corresponds to moving a feature (variable artefact) from application domain to core during product-line evolution. For example, Strategic Sourcing was not a core module before PeopleSoft Financials and Supply Chain Solutions v9.1 came out and was handled by domain specific customizations as application artefacts with reuse capability.

3) Edges can be removed from K intersection R_1 without decomposing L×P. Some legacy features may have replacements in ERP that are functionally equivalent, but are more streamlined and efficient. Such paths within the non-disjoint graph can be eliminated without affecting the graph as a whole [17], [18].

4) If K is altered by adding edges from the region R_3, the resulting graph will have an entirely different decomposition of L×P. This represents a widening of the requirements gap, and should be controlled as the project expands with increased client familiarity with the new product. Thus an agreement should be put in place to limit the rate of expansion proportionally to cost, time, and resources. Our case study identifies the bid confirmation number as a custom feature to be added to an already core version of Strategic Sourcing module. As we proceed through our system development life cycle we continue to remove edges from R_1 and keep adding more edges in R_2. Some of these features can then become part of the core during the next implementation as a preconfigured portion of the demo solution, and reside safely in R_2.

VI. CASE STUDY

As ERP implementation consultants for five state projects along with many others, we experienced phases where even a fine-grained piece of functionality can offset the allocated time, resources, and budget of an entire project. To better manage ERP engineering challenges in these projects, methodology toolkits and solution portals for managing the system development life cycle have been developed.

A. Event Bidding Process

Our experience indicates that one of the primary reasons for failures along with negligence in Critical Success Factors as described in [19], the inability to define a disciplined, quantifiable customization plan with future release features and upcoming patches kept in mind. A formal research method can be the answer to reduce this huge risk. A case
study of an event-bidding, such as undertaken in one of the projects mentioned in [20], can illustrate the practical application of our formal approach.

**Legacy:** The state asks bidders to propose a price for a specific purchase event. The legacy system achieves this function via newspapers, circulars, message boards, internet advertisements, and regular mail. Bidders respond with their proposals, which can include multiple files, design and contracts. This information is mailed to the specified response address, and once received a confirmation receipt is sent to the bidder. The event management office selects a winner, and notifies all the bid participants through mail.

The method above has limitations which include: a potential bidder can miss the circulation and fail to meet the deadline; mailed responses may be delayed, lost or damaged; and after an offer is placed, the bidder has little means to modify or improve it as even if a new bid package is sent it may not meet the deadline. The integrated ERP solution of the event-bidding system is described as follows:

**ERP:** The state asks bidders to propose a price for a specific purchase event. The ERP system creates the event within the integrated system. Suppliers can instantly view events where a bid can be placed. Bids are made online, attaching the necessary documentation and a confirmation with a unique code is generated. The event manager logs into the system and can see all the bids placed and information on each of the bidders. Once the event deadline passes, the event manager can analyze the information and select a winner. All the bidders get instant notification of acceptance or rejection.

**B. Analysis**

Whereas the legacy system only provided basic messaging and tracking of purchases through data entry, ERP provides a completely integrated online solution with Rich Internet Architecture (RIA), in addition to keeping referential integrity zeroing in the chances of human error. The implementation used is PeopleSoft Enterprise Strategic Sourcing [21], which does not provide a mechanism for generating confirmation numbers. The corresponding ERP functionality therefore had to be developed through customization, and then was integrated into the system so that event managers and bidders could use it to view confirmed bids. This discrepancy was well-judged and the correctly recognized as a gap during FGA.

In our formal model the bid confirmation number is one of the features that exists in L, but not contained in the ERP feature set P. In the legacy system, this number provides information on the bidder’s identity, whereas the pre-customized ERP system can only generate incremental integers per bid submission. Bid confirmation number that conglomerates the bidder identity was a missing edge in the bi-graph of the event-bidding problem. An additional edge was then added in the R₃ region of its canonical decomposition, which enforced further testing and enhancement of the system as it changed the decomposition of L × P. The identified gap resulted in major changes to the delivered data repository, underlying record, and code related to the panels, fields, and other component. It also changed the way querying and message call was intended by the original client.

**C. Agile Product-Line Implementation**

Early formal representation of matching and exterior covering thus can help analysts decide on quality of such design and based on the volume, change management can help practitioners choose between options. Furthermore, in other planned changes for related modules or their delivered regular business functions also need consideration. Proper mitigation effort is a necessity to trace back down to all related components and objects in the system. Matured solutions in the market be it ERP or any other packaged...
system generally provides a flexible tool for user-friendly upgrade with backward compatibility. The mainstream reasons of bug-fixes, patches or upgrades can be: version control of database management systems; change in legislature, business procedures and/or policy; dynamic evolution of a beta released module; and a new version of the implemented solution which is more suited with the current business trend.

As customizations happen in the application layer, change in domain layer of system can prove that such modifications are now become boiler-plate extensions adversely affecting the operation both qualitatively and quantitatively. Efficiency of system and people involved can largely reduce as frustration and unhealthy segmentation among stakeholder’s increases out of this. Classifying customization as leaves of a core, similar to basic product line engineering helps the constraints on market size, window and penetration, along with time, budget and resources available. The other factor is the delivery date, as most cases ERP implementations fall short of their project duration due to possible dynamic evolution. Using the model as represented in Fig. 3 we were able to achieve faster elicitation within an agile implementation cycle.

D. Results

For proprietary reasons we are going to apply code names for our clients. Projects N, F, and FP will respectively represent a project which used common RE (N), formal FGA and RE as described (F) and formal FGA and RE with Product-Line Engineering as shown in Fig. 3 (FP).

In project N we had state oriented data for approximately 3300 individuals, and the total duration of RE including FGA before entering the development phase took 15 weeks, overlapping our 12 weeks estimation. In project F we first time implemented our model. Although project F contained state-oriented information for total 55000 individuals, the FGA process being formalized was actually ahead of initial timeline projected. The estimated 24 weeks were not necessary before we could enter the development phase, as it was around 21.5 weeks when we started construction. The further benefits achieved along the system development life cycle comparing these two projects can be found in [20].

Our final project FP was a reuse enabled implementation of an upgraded version where we initially developed a methodology tool kit besides configuring the domain artefact of F to enable it to be used for FP. Above case study was a part of FP and as being a financial implementation comparing with F provides much improved results as the entire FGA were downsized to less than a half of the duration required for F regardless of the total size which are closely equivalent in quantity of data and resource. We need to keep in mind that we used all our prior knowledge and reusable artefacts of F in FP which actually provided a quicker response in elicitation, but this streamlining was possible because of the bipartite matching that we started using since implementation in F.

VII. RELATED WORK

In our research of using a test-first parallel processing technique for assuring appropriate migration of data from legacy to ERP, we mentioned about data mapping during FGA [20]. Those mappings were a product of the bi-graph approach discussed in this paper. To the best of our knowledge, our research is the first seeking to formalize the requirements associated with porting an existing legacy system to ERP.

The work by Illa et al. [22] is most closely related to this research. It describes an approach for selecting ERP products using a formal description of their relevant characteristics. The authors extend previous work on the Systematic Help for ERP Acquisitions (SHERPA) methodology [23]. However, the new version of SHERPA is based on a formal language for translating user needs into requirements over the ERP products. In this paper we do not address the issue of selecting an ERP product from a candidate pool. However, utilizing such a methodology prior to applying our approach could help identify the best software candidate to be used for matching up functionalities. This would help to minimize the number of customizations needed from the beginning, thereby reducing the problem complexity and improving the overall approach.

Soffer et al. [24] describe the process of modeling ERP system. They consider a whole range of process alternatives supported by the ERP system and the interdependencies between them. In addition to defining a comprehensive set of generic modeling steps for ERP, they provide desirable characteristics of ERP modeling languages. While they take a more general approach to ERP modeling, we focus in on formal modeling of the requirements involved in migrating from legacy to ERP.

VIII. CONCLUSION

Our research work provided several benefits in various facets of the ERP software engineering process. Though implementation scope grows exponentially, putting all of the domains involved into one single bi-graph definition was not possible, so we used a collection of sub-graphs defined as part of a forest [17]. A bipartite graph $G$ can be factored as a product $G \ast H \ast K_2$ of a graph $H$ with the corresponding complete graph $K_2$ similar to factoring an even integer $g$ into a product $g = h \times 2$, except that for graphs the factorization need not be unique [25]. This trade-off was handy in running iterative enhancement processes.

Future work involves developing a tool able to define multiple domains, input in the features formally, and then generate the intended cover property of customizations. Any views, opinions, findings, and recommendations presented in this paper are those of the authors and do not necessarily reflect those of the Ultimate Software Group, Inc.

REFERENCES

[1] Achieving, Measuring, and Communicating IT Value, Deloitte Touche and IDG Research Services Group Rep., 2002.
[2] D. M. Togut and E. Bloomberg, Morgan Stanley Research Rep., Morgan Stanley CIO Survey Series: Release, vol. 4, no. 5, 2003.
[3] M. von der Beeck, T. Margaria, and B. Steffen, “A formal requirements engineering method for specification, synthesis, and verification,” in Proc. Eighth Conf. on Software Eng. Environments, IEEE, 1997, pp. 131-144.
[4] I. K. Bray, An Introduction to Requirements Engineering, Addison Wesley, 2002.
Tariq M. King is an engineering manager and a test architect at ultimate software in Weston, Florida. He held his Ph.D and M.S. degrees in computer science from Florida International University. In his role at Ultimate, Tariq provides technical leadership and strategic direction in software testing and quality assurance. His research areas are in software testing, autonomous and cloud computing, model-driven engineering, and computer science education.