Structural Estimation Methodology for Estimating Enterprise Application Integration Projects

Manjula Natarajan, Infosys Ltd.

Abstract. Enterprise Application Integration (EAI) projects aim to integrate a host of application systems built on disparate technologies. The integrating solution should offer a platform to achieve interoperability seamlessly, thereby improving business process efficiency. This article summarizes the structural software sizing approach and its use for sizing and estimating EAI projects. The structural estimation methodology considers functional requirements as well as technical implementation styles of application integration projects.

Introduction

The worldwide Application Infrastructure and Middleware (AIM) space is ever-growing and according to Gartner, the AIM software revenue market totalled $19.3 billion in 2011[1].

In our attempts to measure the size of integration projects to precisely study productivity aspects, traditional software sizing models fell short of addressing the key requirements involved in such projects [2]. Extending traditional models to size integration projects [2] involves approximating the units/weights for every additional processing and systemic requirement that cannot be addressed directly by the base reference model. This also triggers constant validation of the units/weights additionally assigned, in comparison with the recommended size unit. For example, one extended Cosmic Function Point (CFP) should be validated with one standard CFP.

Integration projects are characterized by a host of factors such as the participating systems, the underlying technologies, data interfacing complexities together with the ability to prepare and present data, either synchronously or asynchronously for the participating applications, application of additional business processing logic, and so on, and essentially, making all of these possible at runtime.

While it is to an extent possible to size part of functional requirements in an EAI scenario in terms of data exchange requirements, the implementation aspects of data exchange and processing requirements ranging from i) direct product configurations to ii) extended custom logic, with different shades of these two implementation types, should not be overlooked.

Hence the associated challenges in sizing EAI applications are twofold, namely:

- To assess application complexity related to synchronous or asynchronous data exchange requirements and data processing requirements.
- To assess implementation characteristics, while being able to size the application upfront during requirements gathering stage.

Traditional estimation models fail to integrate the factors of software architectural requirements along with the functional (integration) requirements.

It therefore necessitated the development of a sizing model suited for estimating integration projects from all aspects of functional, technical and systemic requirements. The structural estimation methodology thus had its genesis to provide realistic estimation across the lifecycle stages of enterprise application integration projects.

The essence and the uniqueness of the model lies in its ability to capture the integration project complexities associated with the run-time data exchange and data processing requirements

The structural estimation approach, covering the functional, technical and systemic requirements is equally applicable and extensible to the following application areas demanding various implementation complexities:

- Business process integration and BPM-SOA
- System integration solutions
- Network-based integration services
- Product-based configure and build solutions
- Package implementations

The structural estimation model for EAI projects has been applied and validated on 10 application integration projects. The accuracy of estimation made in planning stages, was assured during design stages, and confirmed during closure. The model also enabled performance comparisons of these integration projects.

This article presents the model, examining the systematic approach for estimating EAI projects and covers the following:

- The integration application complexity comprising of run-time data exchange complexities, data processing complexities, and additional systemic complexities.
- The EAI sizing procedure following the software structural elements and the associated complexity factors.
- The approaches taken to validate the model and the derived business benefits.

The Model

The purpose of software sizing is to determine the cost of development and implementation. It addresses both the business functionality being implemented (what) and the technical implementation of the business functionality (how).

Backfiring methods are used by some organizations to bridge the gap between the functional requirements and technical implementation to derive the size. But these methods lack clarity in combining these aspects to derive the size units, and in the majority of cases, the methods do not consider the implementation approaches.

The structural estimation methodology addresses this gap between what functionality is to be built and how it is to be built, to derive the cost estimation. The model considers the software architectural layers as the focal point. In an EAI scenario, the software architectural layers include the host of integrating systems, and the EAI layer itself comprising of interfaces and data structures needed to integrate the external systems.

Complexities, Representation and Sizing

The structural estimation methodology covers the functional factors attributable to each such layer, and the associated implementation complexities. Size is derived from combin-

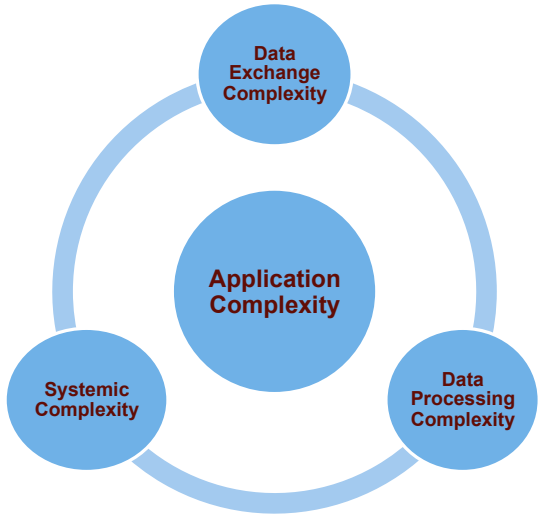

Figure 1. EAI Application Complexity Representation

ing the following requirement complexities: a) data exchange requirements, b) data processing requirements and c) additional systemic complexities. The application complexity categorized with these requirements is represented in figure 1.

The implementation complexities are then studied for each of these requirements and addressed with appropriate weights.

The steps involved in deriving the total EAI application size are represented in figure 2 and described below:

Step 1: List the Software Architectural Points

This includes the host of integrating systems whilst the EAI Layer that integrates the external sources has its own building blocks comprising interfaces, data structures, etc.

For an example, in an EAI project aiming to integrate four different systems using two different interfaces, the software architectural points include the four external systems as well as the two interfaces of the EAI layer.

Step 2: Itemize the Data Exchange Requirements and Processing Requirements of Each Building Block

Each building block or the software architectural point will have an associated data exchange requirement—to feed-in data to or to subscribe to data from, other interfacing points. These form the data exchange requirements and typically include:

- Data received from external sources (internal storage)
- Data to be published to external sources (internal storage, logging requirements)

For logical collection and synchronous or asynchronous exchange of processed data, the following data processing requirements might apply: Data mapping, enrichment, transformation, extraction, encryption, decryption, synchronization, data validation, business processing, etc.

Consider a source application sending messages (data exchange) that are to be processed/enhanced (data processing) and transmitted (data exchange) to a subscribing application. Here the messages are received by the EAI interface, processed/

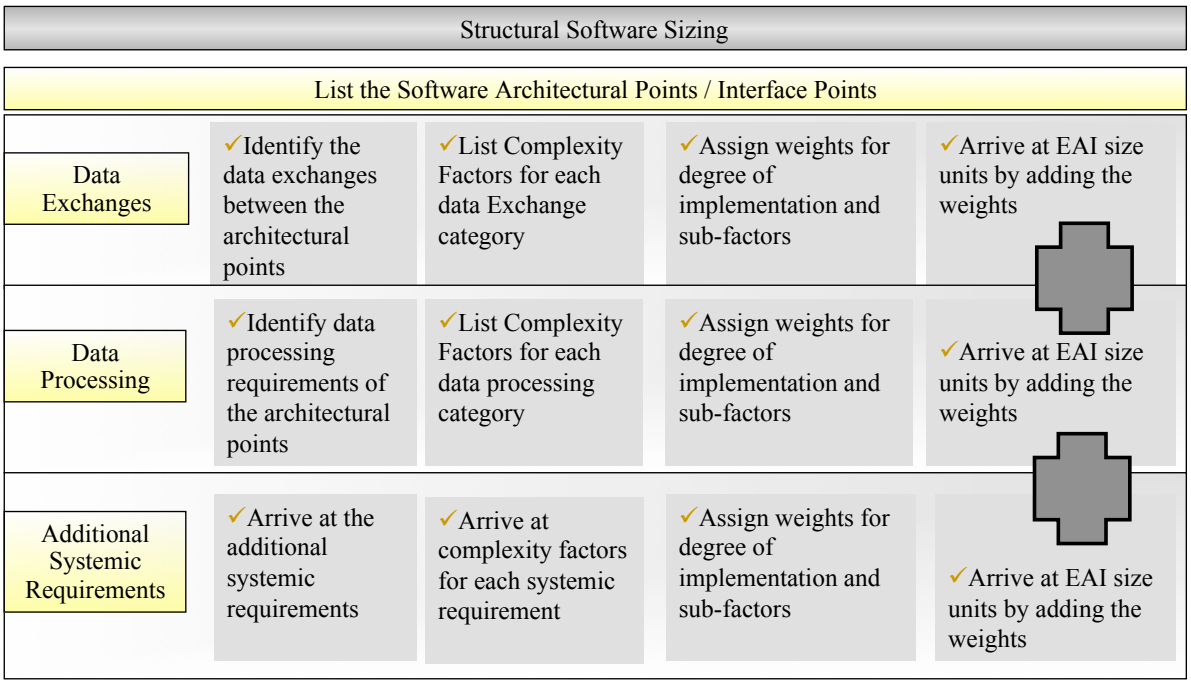

Figure 2: Steps in sizing EAI Applications using Structural Estimation Methodology

Note: The weights are assigned based on a three-point scale, in alignment with the degree of technical implementation—fully configurable, partially configurable and manually constructed. Also the weights vary from one EAI package to another.

enhanced and sent by the interface to the subscribing application. This is a simple example of data exchange and data processing from the three software architecture points, namely the source application, the EAI interface and the destination point.

Step 3: Determine Complexity Factors Associated With the Two Category Heads (Data Exchanges and Data Processing)

For every data exchange requirement, the implementation complexity factors dealing with metadata preparation needed for the data exchange, data exchange/transport mechanism, data routing mechanism and conditions, will be determined.

For example, an EAI interface receives purchase order details from a legacy purchase order processing system. Here the complexity factors to be considered during the data reception shall include: preparation of schema or metadata to receive the order data, size of the metadata as number of data elements, configuration of data reception through appropriate adapter or end point channel, any extraction logic and data parsing conditions.

For data processing requirement, the complexity factors deal with ascertaining if the data processing will be done by directly configuring the integration product or by usage of COTS utilities or by custom logic or a combination of the above three techniques. This will ensure proper size assignment for simple to complex data processing conditions.

Step 4: Determine Additional Systemic Requirements and Associated Complexities Affecting the Overall Application Integration

Usually, the data-subscribing applications in a B2B scenario might require the EAI interface to pass data in an encrypted form. Additional requirements might include passing the data in chunks which will have to be received in a logical sequence and reassembled during execution.

The EAI layer should address such additional requirements which are neither direct data exchange requirements nor data processing requirements. The systemic requirements are those associated with the additional technical requirements applicable for the seamless integration.

Step 5: Assign Weights for Degree of Implementation for Each Factor and the Associated Sub-factors

Additional factors to be considered here include the degree of customization needed, which can be ascertained by the integration product in selection.

For each of the complexity factors considered from Step 3 and Step 4 above, the associated sub-factors need to be analyzed based on how the complexity factor is to be implemented using the integration product. For most of the processing and exchange requirements, the implementation may be facilitated using in-built product features, or by using COTS utilities, or through custom development. The degree of implementation will be studied for each complexity factor and weights assigned based on the nature of work involved.

Based on the above technique the size output is determined by adding up the individual size units, denoted as:

Total Build Size in package-specific "EAI points" =
(Data Exchange Size Units from the assigned weights in "EAI points"

+

Data Processing Size Units from the assigned weights in "EAI points"

+

Additional Systemic Complexities Size Units from the assigned weights in "EAI points").

Validation Approach

The assignment of weights as a unit was carefully made from the multiple iterations of the following steps:

- Determining degree of implementation of each complexities assigned across the three categories.
- Rank ordering the complexity assigned across the three categories.
- Assigning the unit and the weights in scale factors for each complexity.

The sizing model was then validated by applying 10 EAI projects with the integration scope covering the majority of the factors considered, and by using the following approaches.

- Rank ordering of projects based on computed size units and comparing the order with the projects' scope based on expert inputs. The direction of magnitude was confirmed.
- Plotting the size units against the effort consumed for build & unit testing. The observed R squared value was 0.97.
- Checking the size vs. effort relationship at the granular component/interface/complexity level. This was done to validate the approximation of a size unit evenly across the various complexity levels. This step also helped to under stand the consistency of an EAI size unit across EAI projects with varying complexity scope: data exchange rich integrations, data processing rich integrations and complex integrations covering data exchanges, data processing and other systemic requirements.

The methodology usability was verified by conducting a reproducibility exercise for one project with seven expert estimators to determine the size units. The experts were subjected to an initial orientation on the sizing exercise and the project scope for sizing. The insignificant variation in the size units confirmed the usability of the model.

Further, for improved usability and reproducibility, package-specific sizing tools with user interfaces have been created. Users need only to enter the integration requirements, and the tool automatically provides the computed EAI project size in package-specific EAI points. The following graph depicts the linear relationship between EAI points of EAI projects and the associated project build effort.

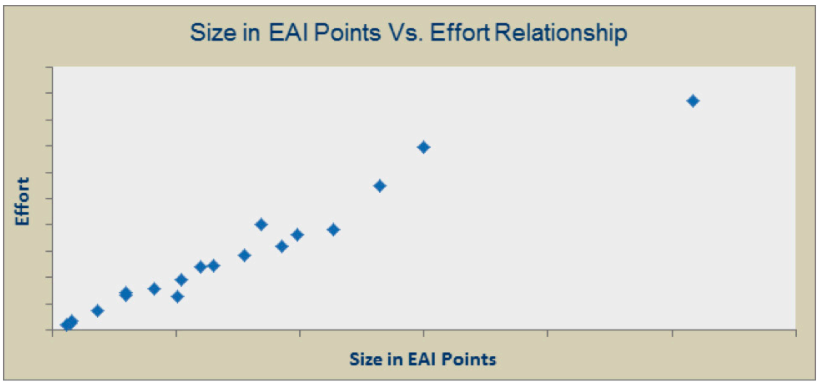

Figure 3. Size in EAI Points Vs. Effort Relationship

Effort Equation From Historical Data

For validating the size units, a linear relationship was established between the size units of each project and the respective build efforts from historical data. The total size units for each past project were listed against their respective build effort and the two variables were studied for linear relationships by plotting the size units against the build efforts. The observed R squared value was 0.97, showing greater linear relationship between the size units and the build efforts. The effort equation thus arrived is used for estimating the build effort for a given size.

Note: The effort equation is dependent on the historical data which largely reflects the standard organizational process capability and hence the baseline performance.

Once the performance baselines are thus established, the projects can effectively estimate the build effort and plan for improved performance leveraging the organizational process, project and risk management capabilities.

Business Benefits

This model is best suited for estimation at early lifecycle stages of EAI projects and provides the following business benefits:

- Improved accuracy in estimation leading to enhanced cost and schedule planning.
- Structural estimation leads to effective management of multi-vendor outsourcing/contractual projects at a possible logical level.
- Facilitates effective project staffing and execution models based on sizing at specific requirement levels.

Conclusion

The structural estimation approach:

- Adopts a scientific approach towards software estimation, covering the integration requirements and the multi-dimensional complexities of the actual building blocks.
- Provides improved accuracy in sizing, thus leading to proper effort and cost estimations.
- Allows effective management of costing and scheduling the work pieces by slicing and dicing and rolling-up the size units at any required level.
- This feature, allows an organization to effectively outsource different pieces of work, and aids in selecting appropriate project execution models and accurate staffing.

Future developments will include studying its fitment for all types of configure and build solutions. Extensions shall be made to derive appropriate adjustment factors for sizing maintenance work in all the applicable areas.

Acknowledgements:

The author acknowledges the contributions of Prakash R, Raghavendra Rao, Rekha Kodali, Vivek Shaurya, Santhosh Maruthaiyan, Senthil Kumar P.R, and Vinayakaraman S, towards applying the methodology across EAI package services. The author thanks the CrossTalk editorial board, its technical reviewers, Haresh Amre, Moses Raj and Anshuman Tiwari for their support and valuable feedback.

ABOUT THE AUTHOR

Manjula Natarajan has completed her master's degree in computer applications and postgraduate program in management. Manjula has more than 16 years of experience in top Indian IT service organizations, spanning multiple roles including development, delivery management, program management, SEPG/QAG and IT process consulting.

She holds the title, Principal – Quality Programs, at Infosys Limited where she contributes towards establishing engineering processes and measurement frameworks, engineering productivity improvement solutions, business outcome and value-based prediction models for project services covering Oracle packages and EAI.

E-mail: Manjula_Natarajan@Infosys.com

REFERENCES

1. Gartner Press Release, "Gartner Says Worldwide Application Infrastructure and Middleware Market Revenue Grew 10 Percent in 2011", Gartner.com
2. Naveen Krishna, "EAI Estimation Challenges – Cosmic FFP Efficacy", Infosys.com