

Key questions from an enterprise data architect

Plan for project success by taking measurements of the relevant data

Skill Level: Intermediate

[Uche Ogbuji \(uche@ogbuji.net\)](mailto:uche@ogbuji.net)

Partner

Zepheira

06 May 2008

Data is the lifeblood of the enterprise, and the best way to prepare for a development and integration project is to document the characteristics of the data that drive the target applications. Learn the key questions that an enterprise data architect should explore in order to effectively document the characteristics of relevant data and take the most important first step towards project success.

There are many software development methodologies, but all of them emphasize the importance of the analysis portion of the life cycle. During analysis, you look to clearly understand the problem and create use cases. You document requirements and user-acceptance criteria. You document the social and technological environment in which the software is to be used. One part of analysis that's critical to the success of many projects is data analysis. Perhaps you are implementing an enterprise resource planning (ERP) system. Perhaps you are adding sophisticated e-commerce features to an existing corporate Web site. Perhaps you are developing an innovative software as a service (SaaS) product. In any such project, whether you are developing a new application or integrating existing applications, it's essential to understand the nature and flow of the related data.

Developing and documenting the shared understanding of relevant data sets and data characteristics is your primary job as an enterprise data architect. If the project involves integrating existing systems, you have to do a comprehensive analysis of the existing data and governing business rules. Often there is precious little written down clearly, and you have to perform the analysis from scratch. If the project is

creating entirely new components, you might have to create new data models. Either way, the key to getting the right answers about data is in asking the right questions. In this article I present a series of questions you can use to make sure you cover all aspects of data during analysis.

Understand the domain

The first group of questions comes earliest. These are key to understanding the business problem that drives the data. Gathering good requirements is an important prerequisite to this step, because the requirements will lead you to the crucial concepts that you must translate into the data model.

What are the principal terms (nouns and verbs) that come from the requirements and description of the problem space? These terms will become the first data elements in the model.

How do you identify different entities in the problem space? Are people identified by user name? Are products identified by SKU? What are the rules governing the uniqueness of these identifiers? Are identifiers subject to geographical, temporal, or other variations? Note that such identifiers should almost never be used as primary keys in databases or data sets. The very fact that they come from the problem space usually taints their value as stable identifiers.

What is the frequency, volume, and granularity of requests for information in the system? Try to flesh out examples and scenarios to get a sense of the sorts of information users will need to put in and pull out of the system. These can be expanded in the creation of use cases or user stories.

What are the costs typically associated with changes to the data and to the data processing systems? These may be costs incurred in determining, validating, or propagating the changes through the business processes.

In what global regions will the data be consumed? Are the representations of the data subject to translation? Are the representations of the data subject to local interpretations? Are any of the source materials or feeder systems currently internationalized or translated? This is one of the more complicated questions, because it's one of the trickiest topics. But it's a very important topic. Explore these questions not just for their near-term answers but also for likely long-term trends. It's much more expensive to retrofit globalization to an existing system than to build it from scratch, and in today's economy you should always consider globalization unless there is cast-iron certainty that it will not be needed. Remember that globalization is not just a matter of language translation but also of dates, names, financial quantities, measurements, regulatory references, geographical references, and more.

What are the current source materials or feeder systems for information in the project? Try to get samples of existing source material to get a sense of data formats and data quality.

What is the scope of use of the information? Is the data used only within an organization? Within one or more selected departments? Is it shared with customers, vendors, or partners? Is it public?

Establish chain of responsibility

The earlier you can establish the rules, conventions, and patterns that govern data creation, flow, and disposal, the better. Surprises in data governance and responsibility can quickly derail a project. These matters are often the last thing the stakeholders think about, because they're inclined to be focused on the value they hope to gain from the system. So it's your job as data architect to instill the needed discipline. The following questions will allow you to understand who is responsible for data, what their responsibility entails, and when it might change hands.

What data elements are the primary responsibility of the system being developed? What data elements are externally referenced from the system? Try to establish as much detail as possible over ownership of the data. If more than one system can modify the data, which version is authoritative? Is there a process for conflict resolution?

Who creates the data? Who modifies it? Who disposes of it? This takes analysis of data ownership to greater granularity. Who are the key actors in the life cycle of the data? These questions are also closely related to full-blown use cases.

How should access to the data be controlled? It may be sufficient that users are organized into groups with differentiated access for creating, updating, reading, and deleting records. You may also require more granular control over actions on the data.

How sensitive are the various classes of information? What are the consequences of exposure or corruption? Determine the risks associated with the data, whether from leaks of data as it flows through the system, or from changes to the data or to the data processing systems themselves. The answers sometimes depend on policies and regulations, but you might also have to press the stakeholders to commit to business analysis of risks associated with information.

What data elements are considered invariants? This means that they can be changed only after extensive business process and review. Again, policy or regulations might mean that certain data elements are specially protected. This might indicate the need for additional testing to ensure integrity of such data.

**When is the planned obsolescence of the system for processing the data?
When is the planned obsolescence of the current format and storage of the data itself and for the essential information represented in the data?**

Remember to plan for obsolescence. Data usually outlives the systems that process it, but if you have a sense of the useful lifetime of the system, you can ensure smooth transitions in the future. Also, try to separate the essential content of the data from a particular form in which it's stored, remembering that there may be no foreseeable time at which the essential content is obsolete.

Refine the model

After gathering the base information from the stakeholders, the data architect has to refine the models and rules to prepare for lower-level design. This is generally the step in which you translate the conceptual model into the logical model.

Is the data mostly in the form of highly structured values or of documents and flowing text? These two classes of data are different in many ways, and it's a good idea to respect the differences. Clearly, you'll often have both kinds of data in the same system, but you still want to consider processing each in accordance with its nature.

To what extent are data elements dependent on or derived from other data elements? To what extent are data elements dependent on or derived from data externally referenced from the system? Dependencies between data are an important subtlety that requires technical skill to determine, but whether you're normalizing a database or creating properly extensible markup design, analyzing dependencies is key to getting relationships right.

Do you need to maintain versions and history of information? You can get the general idea for versioning needs through business analysis, but it takes experience to translate into implementation strategy. Version control is an ongoing challenge, so you should bite off only as much of it as you need to.

What rules govern which data must be updated and maintained together? At a high level this determines the key relationships in the data model. At a more granular level it outlines needs for transactions and concurrency.

Write it all down

That's a lot of questions to consider, and you might be wondering how best to record the answers. There are no definitive answers, of course, and the best advice is to follow the recommendations of your chosen software development methodology. Traditional methodologies provide very detailed specifications for meta-information, such as the class, activity, use case and structure diagrams of UML. Some agile

methodologies go no further than recommending that you jot down user stories on index cards. I'm not very comfortable with any less than writing down the most important factors of the data architecture. It's enough to do so on a wiki, as long as you do record the notes somewhere accessible and maintainable. Two advantages of a wiki are that it's easy to share and easy to annotate. Sharing helps ensure that all those involved in the project understand its parameters. Annotation helps ensure that the design documentation does not die after it's completed. Change, surprises, and further refinement are inevitable over the lifetime of a project, and the data architect should have every incentive to continue to track the answers to these questions.

Wrap it up

The questions presented in this article are not a complete, ultimate set. Data architecture is as much art as science, and each professional will have a palette of favored techniques, which extends to these driving questions. Also, different organizations and even individual projects have specialized needs that affect what data parameters the architect should record. But having an established set of questions for probing during analysis and design helps provide direction, give participants confidence, and avoid oversight. Give your project a better chance of success by asking the right questions to nail down the shape and dynamics of data as early as possible in the project, and be sure to share the results, make them easy to reference, and easy to update or annotate.

Resources

Learn

- In "[The professional architect, Part 2: Overcoming professional challenges in data architecture](#)", Uche Ogbuji shows you how to break down barriers to good data practices.
- Check out the latest on UML in "[Unified Modeling Language version 2.0](#)", by Bran Selic.
- Visit the [developerWorks Architecture zone](#) for more architecture resources.
- Stay current with [developerWorks technical events and webcasts](#).
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.
- In the [developerWorks IT architecture forum](#), you can share your experiences and confer with other information architects.

About the author

Uche Ogbuji

Uche Ogbuji is partner at [Zepheira, LLC](#), a solutions firm specializing in the next generation of Web technologies. Mr. Ogbuji is lead developer of [4Suite](#), an open source platform for XML, RDF and knowledge-management applications, the [Jacquard](#) agile methodology for team Web development, and the [Versa](#) RDF query language. He is a computer engineer and writer born in Nigeria, living and working in Boulder, Colorado, USA. You can find more about Mr. Ogbuji at his blog [Copia](#).

Trademarks

IBM, the IBM logo, ibm.com, and developerWorks are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.