Design Articles

The design topics in this section are intended for implementers, partners, and anyone who is designing and configuring an Agiloft knowledgebase. The discussions are aimed at helping you understand the differences between various implementation options so that you can make an informed design choice. In most cases, the discussions are based on real-life scenarios that have been encountered by administrative users when designing a knowledgebase.

Knowledgebase design in Agiloft is a large and complex topic, and systems can be configured in an infinite number of ways to address the specific needs of an organization. Oftentimes, there is no one right way to model the data, and some choices don't have a right or wrong answer. However, as you design your own system, these basic principles can help you always make a thoughtful, informed choice:

- **Keep things as simple as possible from the user's perspective.** A good way to judge two alternative solutions is to ask yourself how difficult each would be to describe to a colleague. The easiest solution to describe is usually the best.

  > **Example**
  >
  > Imagine that whenever you make a sale, you need to convert a Lead record into Person and Company records. Instead of having the user manually convert the record, you can create a rule that converts it automatically when the Lead's status changes. This lessens the number of steps a user needs to perform, which makes things simpler for them and reduces the likelihood of errors.

- **Represent information logically**. This often means simply representing information how you think about it.

  > **Example**
  >
  > If suppliers in your business are individuals that you work with directly, you might think of them as people, so it makes sense to represent a Supplier as a Contact Type. However, if your suppliers are companies that you don't have a direct relationship with, it may make more sense to represent a Supplier as a type of Company.

- **Represent information as explicitly as possible**. You're more likely to capture useful information if the system is designed to prompt users for the right data.

  > **Example**
  >
  > If you have a table for collecting bug reports, it's better to have two separate fields for describing the bug: one named Reproducible, typically a Yes/No Choice field, and another named Steps to Reproduce Issue. This is preferable to a single Steps to Reproduce Issue field, which might be left blank if the bug is not reproducible.

Criteria for Evaluating Design

In many cases, a design choice for a knowledgebase lacks a single best solution. You often need to choose between two or more competing possibilities, evaluating the benefits and drawbacks of each choice. This is an important part of the process, because no matter which decision you make, it can impact the rest of your design, so always consider the implications of your choices.

Sometimes time or budget constraints push you towards the quick-to-build solution, but whenever possible, opt for a design that adheres to high standards. In particular, evaluate your design from three key areas: usability, elegance, and flexibility.

Usability

Make sure that the proposed design meets both the business' requirements and the users' needs.

- **User Experience**
  - Make sure that the system is intuitive for users.
  - Always label action buttons and fields clearly and intelligibly.
  - Create clear and direct process flows.
- **Error-proofing**
  - Design a system that prevents user mistakes.
  - Allow users to cancel unwanted actions.
  - Make sure that users can recover from mistakes.
- **Transparency**
  - Make the system functionality transparent to administrators.
  - Make sure that it's as easy as possible to troubleshoot and fix errors.

Elegance

Make sure that the design is as clean and robust as possible.

- **Simplicity**
  - Create an overall simple and elegant design.
  - Set up a data structure that you can easily justify.
- **Layout**

    - Create record layouts that are uncluttered and easy to navigate.
    - Design a look and feel that is attractive and matches the business' image.
- **Efficiency**
    - Address any concerns with system performance, which might mean adjusting views, saved searches, global variables, and other aspects of the system that can impact performance.
    - Design rules with filters that limit the number of records that they run on, which increases processing speed and allows for scalability.

Flexibility

Make sure that the system can be adapted to incorporate new business processes and features.

- **Adaptability**
    - Design a system that isn't overly sensitive to change.
    - If you need to make changes to the system, consider whether the lookup mechanisms, choice lists, and other areas of the system will still be intelligible.
- **Extensibility**
    - Create a system that allows new functions and processes, such as additional record types or workflows, to be easily added to the existing structure.
- **Maintainability**
    - Simplify group and team permissions so that they can easily be managed and changed.
    - Create rules that require as little maintenance as possible. For example, if a filter uses the greater than or equal to operator, it can require additional maintenance if the choice list values change.