Design Articles

The design topics in this section are intended for implementers who are designing and configuring an Agiloft knowledgebase. These discussions are aimed at helping you understand the differences between different implementation options so you can make an informed design choice.

There are an infinite number of ways to configure Agiloft to address the specific needs of an organization. There is no one right way to model the data, and some choices don't have a right or wrong answer.

The following tips will provide guidance when you are faced with difficult decisions.

1. Keep things as simple as possible from the user's perspective. A good way to judge how simple two alternative solutions would be is to ask how difficult each would be to describe them to a colleague. The solution that is easiest to describe is usually the best.
2. The most logical representation is usually the best. This is really another aspect of the first rule, and just means that you should represent information the way that you think about it.

   Example

   If, in your business, Suppliers are always individuals that you work with directly, you probably think of them as people, in which case it is reasonable to represents Supplier as a type of Contact. If however, your suppliers were Companies, you would create Supplier as a type of Company.

3. Represent information as explicitly as possible. For example, if you have a table for collecting bug reports, it is better to have a field named Reproducible, typically a Yes/No choice field, and another field Steps to Reproduce Issue than just to have a field "Steps to Reproduce Issue" which may be left blank if the problem is not reproducible.

The design articles in this section provide some practical advice based on real-life scenarios for implementers and administrative users when designing an Agiloft system. They contain a discussion of the pros, cons, and other implications of implementing a particular solution to a design dilemma. Each uses the criteria below when evaluating and recommending a solution.

Criteria for Evaluating Design

Rarely is there a single best solution when deciding how to implement a project specification. Choosing between two or more possibilities requires evaluating your design decision and thinking through the implications of each choice. When evaluating any implementation design, consider the following criteria:

Usability

Does the proposed design meet the business requirements? The users' needs?

- **User experience**. Is the system intuitive to users? Are action buttons clearly marked and intelligible to users? Are the intended process flows clear and as direct as possible?
- **Layout.** The system's look and feel, as well as form layouts, can have a big impact on how users adapt to using the system.
- **Error-proofing.** Are users prevented from making mistakes? Can they cancel out of an unwanted action? Can they recover from mistakes?
- **Transparency**. Is the system transparent to administrators? If something goes wrong, how hard is it to troubleshoot and fix it?

Elegance

Is the design as clean and robust as possible?

- **As Simple as Possible.** A simple, elegant design is often the best. How easy is it to explain the data structure?
- **Efficiency**. Have you addressed processing speed or efficiency concerns? Agiloft is an exceptionally scalable solution, but good rule and search design are still important.
    - For example, rules should be designed to use filters that limit the number of records they run on, where possible.
- **Maintainability**. How easy is the system to maintain?
    - How much time does it take to manage teams and group permissions?
    - How sensitive is the system to change? For example, a simple filter in a rule that uses the greater than or equals operator (>>=) on a choice list requires additional maintenance if the choice list values change in the future.

- **Time constraints**. Sometimes the quick-to-build solution must be used because of a customer's time or budget constraints.

Flexibility

Can the system be adapted to incorporate new business processes and features?

- **Adaptability**. If business processes or structures change, can the system be extended easily?
  - For example, if you set up a system based on five contract types, what happens when there are 50 or 100? Are the lookup mechanisms, choice lists, and system maintenance still intelligible?
- **Extensibility.** How easy is it to add new functions and new processes to the existing structure? Can new functions be integrated with the existing system?