Disclaimers

• The system used as an example in this architecture document is an adapted version of the Adventure Builder Reference application, which was developed in the context of the Java BluePrints program at Sun Microsystems. However, the architecture documented here does not reflect exactly the implementation provided by Sun.
  ◦ To make it a more interesting and realistic example of an SOA solution, we made several assumptions about the business context and requirements of the system.
  ◦ We have also added a few elements and relations that don't exist in Sun's implementation, and omitted others.

• Always remember that the purpose of this software architecture document IS to:
  ◦ Provide a good example of how to organize your architecture documentation.
  ◦ Show what kinds of information (multiple views, interfaces, behavior, rationale) should be recorded when documenting a software architecture.
  ◦ Show examples of different notations that can be used for different views.

• Remember that the purpose of this software architecture document IS NOT to:
  ◦ Provide an accurate design description of Sun's Adventure Builder system.
  ◦ Show the best design alternative for each design decision in scope.
  ◦ Provide guidelines for designing or implementing multi-tier, SOA, or Java EE systems.
  ◦ Show all possible architecture views.
  ◦ Show all possible alternative notations for a given piece of the document.
  ◦ Show the best possible organization for a software architecture document. (You should compare the template provided here with your needs and adapt and evolve accordingly.)
  ◦ Teach UML.
  ◦ Teach Java EE.

• This wiki is a companion to Documenting Software Architectures: Views and Beyond, 2nd Edition, Addison-Wesley, 2010. The book is sometimes referred to as the "DSA book" in the wiki.

• Places where the documentation is not complete are marked with "TODO" or "TODO: <description of what's missing>". There are three reasons we left these TODOs in the document:
  ◦ Real software architecture documents have TODOs. In practice, these documents are never complete. They should be updated as the system evolves. (The good reason.)
  ◦ Completing that section would be too time consuming or take too much space, and would not add much to the goals of this exemplar architecture document. (The bad reason.)
  ◦ The TODOs also work as places where visitors willing to learn more can practice architecture documentation. Because this is a wiki, you can sign in, complete that part of the documentation and remove the TODO. (The ugly reason.)

• Likewise, some places in the documentation have 'TBD' (to be defined) to indicate that a design decision still has to be made.

• If a section is marked as "N/A", it means the contents of that particular section are deemed not applicable or not necessary.

• Many text segments (e.g., the description of an architecture element) show up replicated in more than one place. It's not a good idea to copy and paste text, because it's hard to keep consistency later. We were able to have such replication (which is convenient to readers) using transclusion.

• In a couple of views we created two versions of the primary presentation diagram, one in UML and the other using informal notation. In a real architecture document, you would almost never spend time doing that. The goal here was to generate discussion of what notation works better for different stakeholders.

• Any diagram shows one (of many) possible depiction of the design.

• Each view has a context diagram subsection. The purpose of the context diagram is to show what is in scope and what is external to the system (or the part of the system shown in that view). Instead of a diagram, this subsection may contain:
  ◦ a pointer to a context diagram in another view in case two views share the same context
  ◦ "N/A" in case:
    ■ a context diagram is deemed unnecessary for that view (example)
    ■ the primary presentation already explicitly identifies the external elements via a color coding (example)
    ■ the primary presentation explicitly shows the boundary of the system (example)

• UML diagrams assume the reader is familiar with the UML notation.
  ◦ In a few UML diagrams we provided a more detailed key showing the UML symbols being used.
  ◦ Where we use a UML `<stereotype>`

    > that is not self-explanatory, there is a note in the key comment box.

• The language in the architecture documentation is more informal than in the main chapters of the DSA book. Like a real technical document produced by a software architect, this SAD has lots of technical terms, internal jargon, and acronyms. It may also have typos and grammatical errors.

• There are pros and cons in using wikis in software projects, so it may or may not be a good idea depending on skills and tools available, documentation requirements, team distribution, development process and other variables.