Advanced Design Collaboration Tool

Intelligent Design Collaboration and Rationale Capture

ADCT Coordinates Design Efforts and Captures Design Expertise

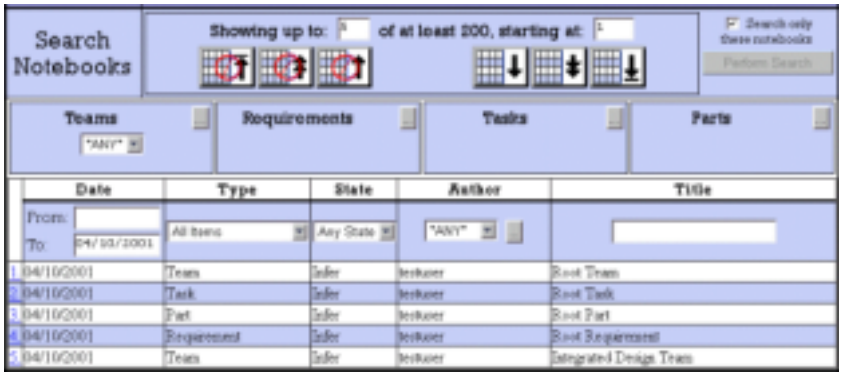
The Advanced Design Coordination Tool (ADCT) is an active, web-based repository of design decisions and underlying rationale. ADCT helps large, distributed design teams work more effectively by tracking possible effects of design changes and notifying affected designers automatically. ADCT also serves as a knowledge management system that preserves an organization's design expertise by providing easy access to the design rationale for previously-solved engineering problems.

ADCT supports large-scale, distributed, interdisciplinary engineering efforts by reducing coordination lapses that can cause project delays or even failure. Specifically, ADCT supports:

- Long-lived products or evolving product families that must be supported over several decades
- Extensive formal requirements and decisions that must be tracked, and managed in the face of inevitable changes generated externally or internally
- Concurrent engineering methods that reduce uncertainty via iterative design and refinement

ADCT provides these capabilities using innovative artificial intelligence techniques that are implemented using web server and relational database technologies. ADCT combines a flexible data model of the design *process*, adaptable to the needs of individual organizations, with a formal representation of design *products*, customizable to any particular design application. This approach enables active tracing of design change consequences, automated notification of affected team members, and structured assistance in resolving design conflicts.

Shared Design Notebooks



Design teams can use ADCT as a shared, searchable, secure, web-based library of engineering design *notebooks*. The notebooks organize team members' *notes* -- short texts describing either the evolving design and design process, or rationale for such product and process data. Each textual note can be supported by document files such as sketches, spreadsheets, datasets, and analysis results.

But notes in ADCT are far more than snippets of text with attached files. Each note is actually a database record that automatically logs attributes such as notebook, author, time, note-type, visibility, version, status, and other user-specified labels. By storing notes in a relational database, ADCT ensures that design histories and rationale are never lost, and remain easily available to teammates and managers—perhaps years after a project ends or after the designers leave the company. Storing attributes with the notes enables a range of capabilities including feature-based searches, privacy, version branching, and design revision.

For example, ADCT users can query and browse the database of notes by:

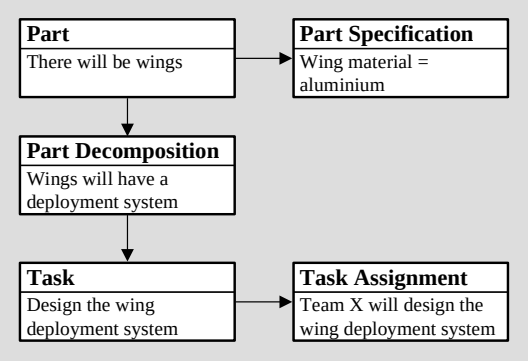
- Paging through a notebook chronologically
- Selecting notes using full text search
- Searching by attributes such as author, time, note-type, or status
- Selecting notes tagged with labels such as requirements, parts, and tasks, and teams
- Browsing the network of notes linked by structured decisions and logical dependencies

Notes that Tell "What" and "Why"

ADCT can be used to capture, organize, and safeguard unstructured, or free text, notes from a distributed design team. But with little effort, it can also be used in a more structured way to capture design history with startling clarity, and to enable new forms of automated assistance. To this end, ADCT recognizes two major classes of notes: product/process notes record *what* was considered or decided, while rationale notes record *why* certain decisions were reached.

Using ADCT, designers can create, edit, link, and visualize notes using customized displays for each note type. When a note has links to other notes, the titles of the referenced notes appear in the display, and are clickable, allowing easy browsing to connected notes. In this way, the structure of ADCT's network of notes is of direct value to designers trying to learn about the status and history of a project.

Product/Process Notes Record "What"



Product/process notes record possibilities about the design and the design process. Specialized note types capture requirements, parts breakdowns, part specifications, team structures, tasking assignments, and other information. ADCT currently supports 14 types of product/process notes, and future versions will support site customization. The figure at left shows links among notes.

In return for picking an appropriate note type, users see direct benefits. One advantage of putting types on notes is that it becomes easier to find any particular piece of information you might be looking for, as the system can restrict searches based on note type. Another advantage of typed notes is that they can carry specialized information. For instance, the kind of note that records the assignment of a task to a team can store references to the team and the task.

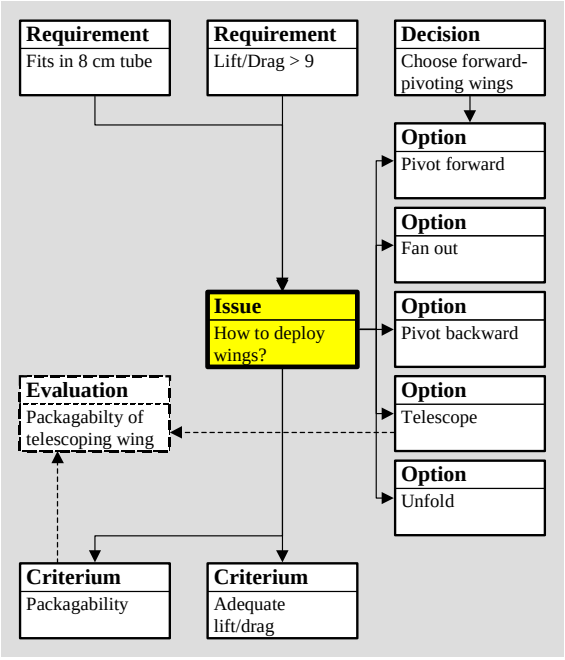
Those data references, in turn, serve several purposes. First, they allow the system to formally recognize the *meaning* of the note: if anyone ever tries to assign that task to that team again, the system will recognize the duplication; if anyone ever tries to assign that task to a different team, the system will recognize the conflict. Second, they provide another way for users to browse and access information: when looking at a note describing a team, it is easy to find the team's tasks. When looking at a note describing a task, it is easy to find the team to which a task has been assigned. Finally, since ADCT's data is stored in a standard relational database, it is easy to generate ad-hoc reports such as lists of tasks that have not yet been assigned to any team.

Rationale Notes Record "Why"

Rationale notes let designers record the reasoning behind the commitments captured in product/process notes. Several kinds of rationale notes work together to capture a structured decision process:

- **Issue Notes** describe decision point -- either major ones that might be the focus of entire trade studies, or minor ones that are resolved with a little thought by a single engineer.
- **Option Notes** describe alternate possible resolutions of Issues. A major Issue might have several well-analyzed Options. A minor Issue might start out with only a single recorded Option; nonetheless, the Issue/Option structure provides a place to attach new Options, should the Issue need to be revisited in the future.
- **Criteria Notes** describe how Options for an Issue are to be evaluated. Criteria generally relate back to Requirements. Again, the number of formal Criteria is likely to vary with the importance of the Issue.
- **Evaluation Notes** record a discussion and a rating of how a given Option fares with respect to a given Criterion.
- **Decision Notes** provide a place to summarize the reasons for choosing a particular Option for a particular Issue; the Decision marks the Option as active. If designers do not want to frame an entire Issue with Options, Criteria, and Evaluations, ADCT provides a view that focuses on Decisions and does not require managing the underlying rationale data structures.
- **Conflict Notes** offer a way to record problems caused by mutually incompatible assertions. When a Conflict is noted, Decisions that support the offending product/process notes must be revisited.

Example: Wing Design for a Deployable Glider



Consider the following aerospace design problem: how to design an unmanned, guided glider that stows in and deploys from a cylindrical canister. A major design issue is how to deploy the wings. The design must be highly reliable, provide a high lift to drag ratio, and enable the wing to fit in an 8cm diameter tube while in its stored position. In one exercise, designers came up with eight possibilities for the wing deployment, including wings that telescoped out, pivoted forward, pivoted backward, or fanned out. Criteria included packagability, cost, reliability, weight, and space.

The figure above shows a simplified version of the decision and rationale to have wings pivot forward. The matrix display below shows an evaluation of each option against each criteria.

Criteria vs. Options	Forward-Pivoting Wings	Backward-Pivoting Wings	Center-Pivoting Wings	Fanning-Out Wings	Telescoping Wings	Unfolding Wings	Unrolling Wings	Inflating Wings
Wing Deploy Adequate Lift/Drag	0.90	0.90	0.60	0.60	0.80	0.80	0.60	0.60
Wing Deploy Packagability	0.70	0.70	0.70	0.90	0.80	0.80	1	1
Wing Deploy Reliability	0.80	0.50	0.30	0.80	0.30	0.50	0.20	0.30

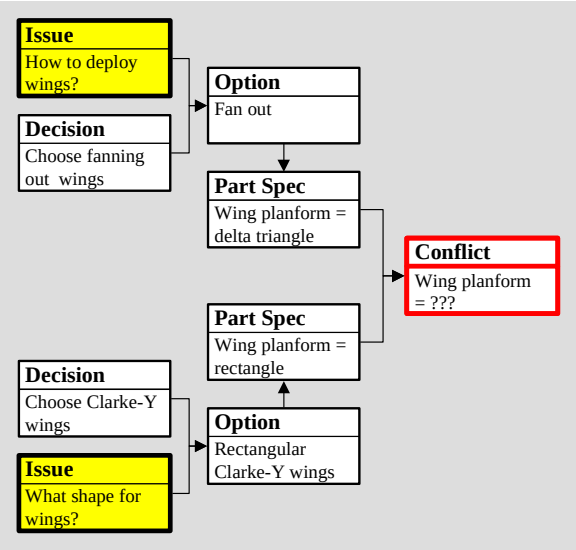
Linking "What" & "Why"

Product/process notes capture "what" may possibly be true about the design or design process, while rationale notes capture "why" certain decisions were reached. These two kinds of notes are linked together in a *dependency network.* The idea is that key rationale items—notably Issues and Conflicts—depend on prior product/process assertions, and such assertions, in turn, can depend on rationales—most especially Options.

Combinations of product/process notes can lead to Issues that must be resolved. In the glider example, Requirements for packaging and reliability combine to raise the Issue of wing deployment. Each Option has one or more product/process assertions that result if the Option is chosen. The "forward-pivoting wings Option" supports product notes that introduce pivots and position them with respect to the wings and body.

These dependency links can be used to maintain consistency as the contents of the repository change. Product/process and rationale notes can be either *active* or *inactive*. Instead of deleting notes, ADCT marks them as inactive to preserve a complete record of the design process. Now, if *any* of the product/process notes leading to an Issue become inactive, that Issue may no longer be relevant and might also need to become inactive. If an Issue becomes inactive, its Options should become inactive as well. If an Option becomes inactive, the product/process notes it leads to may deserve to be inactive as well. But since aspects of product and process design can be supported by more than one Option, the rule here is that *all* the supports for such a note must become inactive before it is reasonable to make the note itself inactive.

Design Conflicts



A Conflict, like an Issue, exists because of some combination of facts about the project. In this case, the facts don't simply present a challenge to be solved. Instead, they represent an inconsistency to be resolved by removing some subset of the conflicting facts. Dependency links to Conflicts and the optional links to Options and from Issues play into the activity calculations as well. The figure at left shows how incompatible decisions for two issues leads to a conflict. Specifically, the Decision to deploy wings by fan out Option implies a delta-shaped wing which conflicts with a previous Decision to use a rectangular wing. A Conflict note identifies the two conflicting Part Specification notes.

Automatic Notification of Design Changes

Using dependency information, ADCT can route change notifications automatically to appropriate team members. Pending notifications trigger a flashing icon in the user's display. From their notifications page, users can review pending notifications, and from a notification they can follow a hyperlink to a relevant Note.

When any team member rescinds a Decision, the corresponding Option becomes *inactive*. This may mean that some of the Option's dependent product/process notes no longer have any active support. In that case, the authors of those notes are notified and encouraged to make them inactive. If a product/process note is removed from active status, any Issues or Options it supports may need to be revisited, so their authors can also be notified.

ADCT also lets users turn such active/inactive decisions over to the system on an item-by-item basis, so the system will not stop to notify an author that the status of an item might need to be changed. Instead, it will go ahead and change the status automatically and then notify the author that the change has been made. In this mode, the effects of changes can ripple forward through the dependency network.

When a Conflict is noted, the system identifies relevant parties by searching backward through the network. Each of the product/process notes implicated in the Conflict is traced to the active Options that are its supports, and the authors of those Options' Decisions are notified of the Conflict. Those decision-makers are invited to join an ad-hoc discussion group about the Conflict. Conflicts will normally be resolved when one of the decision-makers rescinds one of the Decisions.

By exploiting the dependency network, ADCT greatly improves project coordination. Its notification mechanism ensures that all the right people find out, in a timely manner, about changes and conflicts that impact their work. Its discussion group mechanism provides automated support for asynchronous and distributed resolution of conflicts. In combination with the versioning mechanisms discussed below, dependency processing enables "what-if" explorations of alternate designs when changes are necessary.

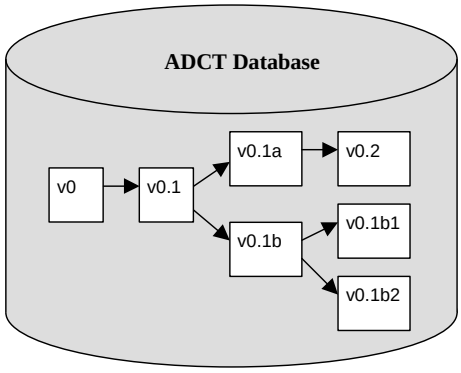
Discussion of Designs and Changes

Conflict discussion groups are just one example of ADCT's general facility for managing threaded discussion attached to Notes. If a Note is public, team members are free to view it and free to post comments in a public forum. ADCT safely stores the entire history of such discussions along with the basic Note. Conflict discussions are special in that they initiate automatically, subscribing a selected group of users who are then notified of new postings.

Conflict discussions also support the invitation of other users into the group: either authors of other Decisions further upstream or leaders of teams whose members are already engaged in the discussion. Escalation of a Conflict discussion is useful when the original notified group cannot settle on a resolution.

Versioning for Concurrent Design

The rigid, one-pass waterfall process model can lead to inefficient designs. Concurrent design enables design teams to explore alternatives and iteratively refine all aspects of the design, as additional information is generated.



To support concurrent design, ADCT supports a branching model for storing multiple versions of all designs. Any user can split a branch off a previous version. Users can designate any version as their current working version, and so long as that version is unlocked they can perform edits that register in that version. Each version only records its differences from prior versions.

Ready for the Web

ADCT uses a 3-tier, web-based architecture as shown in Figure 5. This approach eases support for distribution and replication, making the system more accessible, available, reliable, and scalable. Users can work from any machine that has a modern web browser and a network connection. The ADCT web application server generates standard HTML and JavaScript to create a custom user interface for viewing and editing the shared notebooks and their entries. ADCT stores all its data within a relational database to provide secure, reliable, scalable, concurrent queries and updates.

Ready for Your Data Center

ADCT's database schemas have been designed to ease exploitation of pre-existing project data. ADCT's versions of commonly available project structure and management data, such as requirement breakdowns, parts lists, tasks, teams, and team membership, can be keyed to match existing database schemas.

System Requirements

The ADCT server is implemented in Java for portability and runs on computers running Windows NT v4.0 and Windows 2000. The server runs comfortably on a Pentium III 600MHz CPU with 256 Mbytes of memory and large amounts of secondary storage. For light use, a single machine can support both the ADCT Web Application Server and the ADCT Database. ADCT v3.0 requires Oracle 8iR2, although future releases may support other databases. Either Microsoft Internet Explorer v5.x or Netscape Navigator v6.x can serve as the ADCT client.

About Stottler Henke

Stottler Henke develops intelligent software systems to solve problems that defy solution using traditional approaches. We specialize in artificial intelligence software solutions for education & training, knowledge management & discovery, planning & scheduling, and decision support.