Confluence Space Structure and Directory Rationale

AdjecTex Technology

December 9, 2025

Contents

1 Purpose and Scope

This document provides the rationale for the proposed Confluence space structure and directory (section page) model for AdjecTex Technology. It explains why spaces, section pages, and leaf pages are organized as described in the *Space Structure and Directories* guide and how this information architecture supports day-to-day work across product, platform, architecture, and application security functions.

The scope of this rationale covers:

- The general pattern of *Space* → *Home* → *Section (directory) pages* → *Leaf pages*.
- The specific roles of core spaces: ADJT-HOME, ADJT-APPSEC, ADJT-ARCH, ADJT-PLATFORM, ADJT-DELIVERY, and product spaces (e.g., SBM-APP).
- The key design drivers, alternatives, tradeoffs, and risks that influenced the chosen structure.
- Governance and evolution of the structure over time.

2 Background and Problem Statement

Confluence can either act as a clear and navigable knowledge base or devolve into a "junk drawer" where content is hard to locate and even harder to maintain. Historically, common failure modes include:

- Flat page hierarchies with no clear boundaries between domains (e.g., architecture, application security, platform).
- Inconsistent use of spaces versus pages, leading to unclear ownership and permissions.
- Mixed content on the same page (strategy, reference, runbooks, and ephemeral notes), making content hard to consume and even harder to evolve.

AdjecTex is organizing architecture, AppSec, cloud platform, SDLC, and product documentation in a way that:

- Keeps knowledge discoverable for new and existing team members.
- Aligns with teams, responsibilities, and governance structures.
- Supports reuse of architectural and security assets across multiple products and platforms.

The proposed space and directory model is intended to prevent the "junk drawer" outcome by enforcing simple, repeatable patterns.

3 Design Drivers

3.1 Information Architecture Drivers

The structure is driven by the following information architecture goals:

- **Clarity of Home Pages:** Each space has a *Home* page with a short overview and curated links, not a dumping ground of all content.

- **Predictable Hierarchy:** Within each space, section (directory) pages clearly act as *buckets* for related leaf content pages.
- **Separation of Concerns:** Strategy, standards, templates, reference architectures, product-specific details, and runbooks live in distinct places.
- **Cross-Space Reuse:** Architecture templates, reference architectures, and AppSec program materials are reused across multiple product spaces via links rather than duplication.

3.2 Organizational Drivers

The structure also reflects organizational realities:

- **Program vs. Product:** There are cross-cutting programs (e.g., AppSec, cloud platform, SDLC standards) and product-specific initiatives (e.g., Smart Building Manager, ValueVision).
- **Ownership and Accountability:** Each space has an implicit owning team and a clear responsibility for keeping it current.
- **Stakeholder Perspectives:** Different stakeholders (engineers, architects, security, product owners, management) need predictable locations for their questions and documents.

3.3 Tool and Permission Drivers

Confluence-specific constraints also influence the design:

- **Spaces as Permission Boundaries:** Spaces are used where differentiated access control or a distinct audience is needed (e.g., AppSec vs. product).
- **Pages as Content Units:** Pages are used to host actual artifacts (charters, views, runbooks, etc.), not as the only structuring element.
- **Section Pages as Directories:** Section pages group leaf pages and carry light explanatory text, but not large amounts of mixed content.

4 General Pattern: Home–Section–Leaf

Across all spaces, the following pattern is used:

- **Space Home:** A real page providing a short overview, contacts, and links to major sections.
- **Section (Directory) Pages:** Pages whose primary purpose is to group child pages, e.g., *"1. Strategy & Governance"*, *"2. Templates"*, *"3. Architecture"*. These pages contain brief descriptions and navigation, not dense reference content.
- **Leaf Pages:** Pages that represent actual artifacts, such as *"AppSec Program Charter"*, *"Smart Building Manager – Context View"*, or *"Threat Modeling Playbook"*.

Simple rule of thumb (implemented across the structure):

- If the title is a **category** or **bucket** (e.g., "Templates", "Playbooks & Runbooks", "Architecture"), it is a *directory* page.
- If the title is a **specific artifact** (e.g., "Threat Modeling Playbook", "ADR-0001 – Choose Microservices"), it is a *leaf* page.

5 Space Overview

Table 1 summarizes the key spaces and their primary rationale.

Table 1: Confluence Spaces and Primary Rationale

Space Key	Name	Primary Purpose	Primary Owner(s)
ADJT-HOME	Adjectex Home	Landing space for "how to use Confluence at AdjecTex", high-level standards and space directory.	Org-wide / Architecture
ADJT-APPSEC	Application Security Program	Program-level AppSec strategy, governance, playbooks, tooling, and per-product engagements.	AppSec Team
ADJT-ARCH	Architecture & Engineering Standards	Architecture documentation meta, templates, reference architectures, and styles/patterns catalog.	Architecture Guild / Lead Architect
ADJT-PLATFORM	Cloud Platform & Infrastructure	Cloud landing zone, environments, platform services, and platform runbooks/SOPs.	Cloud Platform / SRE
ADJT-DELIVERY	SDLC, CI/CD & Dev Practices	SDLC model, Git workflows, CI/CD standards, testing strategy, and tooling guides.	Engineering Enablement
SBM-APP (example)	Smart Building Manager Application	Product-specific vision, requirements, architecture, security, operations, and knowledge base.	SBM Product Team
VV-APP (example)	ValueVision Application	Product-specific vision, requirements, and architecture for the ValueVision platform.	ValueVision Product Team

6 Space-by-Space Rationale

6.1 ADJT-HOME – AdjecTex Home

6.1.1 Intent

ADJT-HOME is the organizational landing zone. Its rationale is to:

- Provide a single place where new members learn how AdjecTex uses Confluence.
- Expose a concise *space directory* linking to the main program and product spaces.
- Host only a small number of high-level pages such as *"How to Use Confluence at AdjecTex"* and *"Standards & Policies (high-level)"*.

6.1.2 Why Not a Deep Hierarchy Here

ADJT-HOME is intentionally kept mostly flat:

- Deep hierarchies are delegated to program and product spaces.
- The home space should not become another "everything" location; it acts as a router, not a repository.

6.2 ADJT-APPSEC – Application Security Program

6.2.1 Intent

ADJT-APPSEC centralizes cross-cutting AppSec knowledge that serves every product:

- **Space Home:** Overview of the AppSec program, charter summary, and "How to engage AppSec".
- **Strategy & Governance:** Houses the *AppSec Program Charter*, *Policy & Standards Index*, and *Risk & Compliance Alignment*.
- **SDLC & CI/CD Integration:** Contains the *16-Gate CI/CD Security View*, gate groupings, and security tooling pages (CodeQL, secret scanning, dependency review, ticketing integration).
- **Playbooks & Runbooks:** Threat modeling, secure code review, vulnerability management, incident response, and secret scanning triage.
- **Patterns & Guidelines:** Secure coding guidelines and AppSec patterns for common concerns (auth, input validation, logging, etc.).
- **Engagements by Product:** Per-product AppSec engagement subtrees (e.g., Smart Building Manager, ValueVision, Learning Platform) with context, threat models, risk registers, and findings.
- **Templates:** Reusable templates for threat models, security reviews, security user stories, and acceptance criteria.

6.2.2 Rationale for Directories

Each section page under ADJT-APPSEC is a directory by design:

- *"1. Strategy & Governance"* is a bucket for program definition artifacts, not mixed with runbooks or tooling.
- *"2. SDLC & CI/CD Integration"* groups all materials about how security gates integrate into pipelines.
- *"3. Playbooks & Runbooks"* groups operational guides, which differ in audience and lifecycle from strategy documents.
- *"5. Engagements by Product"* is a directory with one child page per product engagement, which themselves act as directories for leaf artifacts like *Threat Model* or *Risk Register*.

6.3 ADJT-ARCH – Architecture & Engineering Standards

6.3.1 Intent

ADJT-ARCH is the central place for how AdjecTex documents and thinks about architecture:

- **Architecture Documentation Meta:** Explains views vs. beyond views, stakeholder needs, and documentation lifecycle.
- **Templates:** Provides canonical templates for architecture overview documents, context views, module views, C&C views, deployment views, data model views, behavior views, work assignment views, rationale/decisions, roadmaps, variability, interfaces, quality scenarios, and ADRs.
- **Reference Architectures:** Holds reusable architectures (web/microservices, event-driven messaging, data & analytics/lakehouse, cloud governance, security references).
- **Styles & Patterns Catalog:** Documents architectural styles and patterns used at AdjecTex.
- **System/Project Architecture Index:** Index pages that link from this standards space to the actual product-specific architecture spaces.

The rationale is to avoid each product reinventing templates and styles, and to centralize the "how we do architecture" narrative.

6.4 ADJT-PLATFORM – Cloud Platform & Infrastructure

6.4.1 Intent

ADJT-PLATFORM documents the shared platform on which products run:

- **Cloud Landing Zone & Governance:** High-level cloud governance, landing zone concepts, policies, and reference diagrams.
- **Environments:** Patterns for development, test/staging, and production environments.
- **Networking & Identity:** Network topology, identity structure, and related diagrams.
- **Platform Services:** Shared services such as logging & observability, monitoring & alerting, secrets management, and CI runners.
- **Platform Runbooks & SOPs:** Operational runbooks, incident response, backup/restore procedures.
- **Infrastructure Architecture & Diagrams:** Platform architecture views that product teams can reference.

Rationale: platform concerns are cross-cutting and must not be buried inside individual product spaces. This space provides a single source of truth.

6.5 ADJT-DELIVERY – SDLC, CI/CD & Dev Practices

6.5.1 Intent

ADJT-DELIVERY is the home for engineering process and delivery practices:

- **SDLC Model & Policies:** Overall SDLC and policies (feature development, hotfixes, emergency changes).
- **Git Workflow & Branch Strategy:** GitFlow and trunk-based guidelines, branch naming conventions.
- **CI/CD Standards:** CI and CD standards, with cross-links to the AppSec 16-gate view.
- **Testing Strategy:** Unit, integration, performance, and security testing strategies.
- **Definition of Ready / Done:** Shared criteria to harmonize work intake and completion.
- **Tooling Guides:** GitHub, Jira, and code review best practices.

Having a dedicated delivery space separates process guidance from product-specific implementation details and from platform infrastructure specifics.

6.6 Product Spaces (e.g., SBM-APP – Smart Building Manager)

6.6.1 Intent

Each product space (e.g., SBM-APP for Smart Building Manager) is the canonical home for that product's documentation:

- **Vision & Strategy:** Product vision, roadmap, and goals/OKRs.
- **Stakeholders & Requirements:** Personas, high-level requirements, epics, and backlog links.
- **Architecture:** Context, module, C&C, deployment, data model, behavior views, quality scenarios, rationale, roadmap, and ADRs.
- **Security & Compliance:** AppSec engagement link, product threat model, findings & remediation summaries.
- **Operations & SRE:** SLIs/SLOs, runbooks, monitoring, and alerting configuration.
- **Implementation Notes & Testing:** Design notes, technical decisions, test strategies, and results.
- **Knowledge Base:** How-to guides, FAQs, and known issues.

The rationale is to:

- Give each product team a coherent, end-to-end space that mirrors the lifecycle of their system.
- Avoid mixing product-specific details into program spaces like ADJT-ARCH, ADJT-APPSEC, or ADJT-PLATFORM.
- Reinforce the pattern that cross-cutting standards and templates live centrally, while concrete instantiations live within product spaces.

7 Alternatives Considered

Several alternative structures were considered and rejected:

7.1 Single Monolithic "Engineering" Space

Description. All content (architecture, AppSec, platform, SDLC, and products) would reside in a single space with a deep page tree.

Reasons Rejected.

- Poor permission boundaries; hard to grant/limit access per program or product.
- Difficult to maintain a clear mental model as the page tree grows.
- Increases the "junk drawer" risk and makes onboarding harder.

7.2 Spaces Only Per Team, No Program-Level Spaces

Description. Each team (e.g., AppSec, Architecture, Platform) would own one or more spaces without a clear program-oriented split (e.g., no distinct architecture standards or AppSec program spaces).

Reasons Rejected.

- Blurs the distinction between cross-cutting programs and project work.
- Makes it harder to have a single canonical location for standards and templates.
- Encourages duplication of patterns across multiple team spaces.

7.3 Spaces Only Per Product, No Shared Program Spaces

Description. Only product spaces would exist, with standards and patterns copied into each product space as needed.

Reasons Rejected.

- Strong duplication of templates and standards across products.
- Increased risk of divergence and inconsistency (e.g., multiple versions of an architecture overview template).
- Harder for cross-cutting teams (AppSec, architecture, platform) to drive consistent practices.

8 Tradeoffs and Implications

The chosen structure introduces several tradeoffs:

8.1 Cross-Space Navigation vs. Duplication

- **Benefit:** Standards, patterns, and templates are centralized (ADJT-ARCH, ADJT-APPSEC, ADJT-DELIVERY).
- **Cost:** Users occasionally navigate between multiple spaces (e.g., from a product space to AppSec engagement pages or architecture templates).

  This is mitigated by using cross-links from product spaces to relevant program pages.

8.2 More Spaces vs. Simpler Space List

- **Benefit:** Each space has a clear mission and owning team, making stewardship easier.
- **Cost:** The global space list is slightly longer, but this is mitigated by the *Space Directory* in ADJT-HOME.

8.3 Structured Directories vs. Ad-Hoc Trees

- **Benefit:** The pattern of Home → Section → Leaf pages allows users to predict where to place and find content.
- **Cost:** Some content reorganizations may be required when new categories emerge; however, the pattern itself remains stable.

9 Governance and Evolution

9.1 Ownership

Each space has a clear owner:

- ADJT-HOME: Architecture leadership or an agreed central documentation owner.
- ADJT-APPSEC: AppSec lead or AppSec program owner.
- ADJT-ARCH: Lead architect or architecture guild.
- ADJT-PLATFORM: Cloud platform lead / SRE lead.
- ADJT-DELIVERY: Engineering enablement or SDLC process owner.
- Product spaces (e.g., SBM-APP): Product owner and, operationally, the product team.

9.2 Change Process

- Structural changes at the *space* or *section* level should be discussed and agreed upon by the owning team and key stakeholders.
- Significant changes (e.g., new spaces, renaming spaces, collapsing/expanding major sections) should be documented and, if appropriate, captured in ADRs in ADJT-ARCH.
- Leaf pages can be added more freely, provided they follow the naming and placement conventions described in the *Space Structure and Directories* guide.

9.3 Name and Label Conventions

- Numbered section pages (e.g., *"1. Strategy & Governance"*) make ordering explicit and stable.
- Consistent suffixes (e.g., "– Context View", "– Module View") make it clear what kind of artifact a page represents.
- Labels can be used to tag pages by product, domain, technology, or concern (e.g., `appsec`, `architecture-view`, `runbook`).

10 Traceability and Usage Scenarios

10.1 Stakeholder–Question–Location Examples

- **Question:** "Where is the architecture overview for Smart Building Manager?"
  Answer: SBM-APP → *3. Architecture* section (context, module, C&C, deployment views, etc.).
- **Question:** "What is our standard architecture overview template?"
  Answer: ADJT-ARCH → *2. Templates* → *Architecture Overview Document Template*.
- **Question:** "Where do I find the AppSec threat modeling playbook?"
  Answer: ADJT-APPSEC → *3. Playbooks & Runbooks* → *Threat Modeling Playbook*.
- **Question:** "What are the CI/CD security gates and how do they integrate with GitHub?"
  Answer: ADJT-APPSEC → *2. SDLC & CI/CD Integration* → *16-Gate CI/CD Security View (Overview)*.
- **Question:** "Where are the cloud environment patterns?"
  Answer: ADJT-PLATFORM → *Environments* section.

This mapping ensures that the structure is not only theoretically sound but also practical in answering real questions.

11 Risks and Open Issues

11.1 Risks

- **Drift from Intended Structure:** Over time, users may start creating ad-hoc sections or mixing content on directory pages.
- **Stale Landing Pages:** Home pages and section pages that are not kept up to date can mislead users.
- **Over-Segmentation:** There is a risk of creating too many spaces or sections; this is mitigated by using ADJT-HOME as a curated directory.

11.2 Open Issues

Potential areas for future refinement include:

- Finalizing labeling standards across all spaces.
- Agreeing on a formal process for deprecating spaces, sections, or templates when they are superseded.
- Deciding whether some highly related spaces (e.g., ADJT-ARCH and ADJT-APPSEC) should share joint reference pages for security architecture.

12 Conclusion

The proposed Confluence space structure and directory model for AdjecTex is intentionally simple, repeatable, and aligned with both organizational responsibilities and architectural documentation principles. By clearly separating program-level standards and templates from product-specific implementations, and by using a consistent Home–Section–Leaf pattern, this structure aims to keep Confluence usable, navigable, and sustainable as the organization grows.