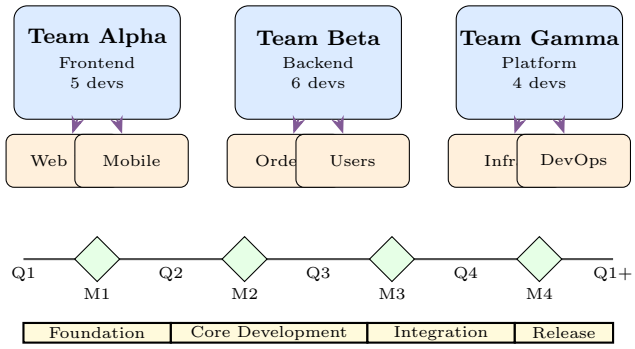
Planner's View

Architecture Viewpoint Specification

Work Assignment, Team Structure & Development Planning



Version:	2.0
Status:	Release
Classification:	ISO/IEC/IEEE 42010 Compliant
Last Updated:	December 12, 2025

Based on the Views and Beyond approach to software architecture documentation

Contents

1 Viewpoint Name

Viewpoint Identification

Name:	Planner's View
Synonyms:	Work Assignment View, Allocation View, Team Structure View, Project Planning View, Resource View, Development Organization View
Identifier:	VP-PLAN-001
Version:	2.0

1.1 Viewpoint Classification

The Planner's View is an allocation-style view in the Views and Beyond approach, specifically the Work Assignment style. It maps software elements to the organizational units (teams, developers) responsible for developing and maintaining them. This viewpoint bridges architecture and project management, enabling effective planning, resource allocation, and coordination.

Table 1: Viewpoint Classification Taxonomy

Attribute	Value
Style Family	Allocation Styles
Primary Focus	Work Assignment and Team Organization
Abstraction Level	Organizational / Planning
Temporal Perspective	Project Timeline and Phases
Related Styles	Work Assignment, Deployment (allocation)
IEEE 42010 Category	Development Viewpoint (organizational aspects)
Relationship	Bridges Architecture and Project Management

1.2 Viewpoint Scope

The Planner's View encompasses the following aspects:

- **Work Assignment:** Mapping of architectural elements to teams and individuals responsible for their development.
- **Team Structure:** Organization of development teams, their skills, and responsibilities.
- **Development Phases:** Timeline of project phases aligned with architectural milestones.
- **Resource Allocation:** Assignment of personnel, time, and budget to work items.
- **Dependencies and Sequencing:** Order in which work must be performed based on architectural dependencies.

- **Risk Management:** Technical and organizational risks affecting development.
- **Effort Estimation:** Estimates for developing architectural elements.
- **Communication Paths:** Required coordination between teams based on element interactions.

2 Overview

The Planner's View provides essential information for project planning and management by connecting architectural structure to development organization. It ensures that the technical architecture and organizational structure are aligned for effective delivery.

2.1 Purpose and Scope

The primary purpose of this viewpoint is to enable project managers, architects, and team leads to plan and coordinate development activities based on the system's architectural structure. It answers questions about who builds what, when, and how work is coordinated across teams.

Viewpoint Definition

The Planner's View maps software architecture elements to organizational units (teams, roles, individuals) and project timeline elements (phases, milestones, iterations). It documents work assignments, team structures, development sequences, resource allocations, and coordination requirements, providing the foundation for effective project planning and execution.

2.2 Key Characteristics

The Planner's View exhibits several distinctive characteristics:

Allocation Focus: Maps architectural elements to organizational and temporal elements, bridging technical and management domains.

Conway's Law Awareness: Recognizes that system structure and team structure influence each other and should be aligned.

Dynamic Nature: Updates as project progresses, teams change, and architectural understanding evolves.

Multi-Stakeholder: Serves both technical (architects, developers) and management (PMs, resource managers) audiences.

Planning Foundation: Provides the structural basis for project schedules, resource plans, and coordination strategies.

2.3 Relationship to Other Viewpoints

The Planner's View connects to other architectural viewpoints:

Table 2: Relationships to Other Viewpoints

Viewpoint	Relationship
Development	Module structure determines work breakdown. Package ownership maps to team assignments.
Logical	Domain boundaries inform team boundaries (inverse Conway). Service ownership by teams.
Deployment	Deployment units may align with team responsibilities. Ops teams for infrastructure.
Component-and-Connector	Runtime components map to developing/maintaining teams.
Context	External integrations require coordination with external parties.
Process	Complex concurrency may require specialized team skills.

2.4 Planning Architecture Overview

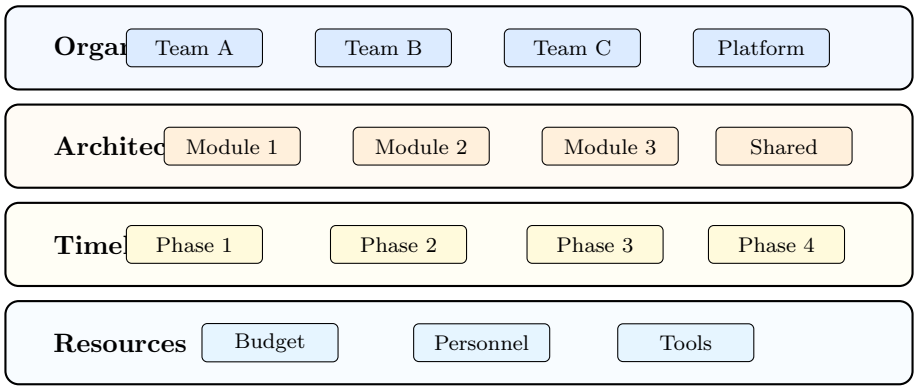


Figure 1: Planning Architecture Layers

3 Concerns

This section enumerates the architectural concerns that the Planner's View is designed to address.

3.1 Primary Concerns

C1: Work Assignment

- Which team is responsible for which architectural elements?
- What is the ownership model (single owner vs shared)?
- How are responsibilities distributed across teams?

- What happens when team composition changes?
- How are cross-cutting concerns assigned?

C2: Team Structure and Skills

- What teams exist and what are their compositions?
- What skills does each team possess?
- Are there skill gaps relative to assigned work?
- How are teams organized (feature, component, platform)?
- What is the team topology?

C3: Development Sequencing

- In what order must elements be developed?
- What are the architectural dependencies affecting sequence?
- What can be developed in parallel?
- What is the critical path?
- How do integration points affect sequencing?

C4: Effort Estimation

- How much effort is required for each element?
- What factors affect estimation (complexity, novelty, risk)?
- What estimation techniques are appropriate?
- How is uncertainty captured?
- How are estimates validated and refined?

C5: Resource Allocation

- How are personnel allocated to work items?
- What is the budget allocation across elements?
- Are resources sufficient for planned work?
- How are resource conflicts resolved?
- What is the resource utilization?

C6: Milestones and Phases

- What are the major project milestones?
- How do phases align with architectural increments?
- What deliverables are expected at each milestone?
- How is progress measured?
- What are the release criteria?

C7: Team Coordination

- Which teams need to coordinate?
- What interfaces exist between team responsibilities?
- How is cross-team communication managed?

- What synchronization points exist?
- How are integration activities coordinated?

C8: Risk Management

- What technical risks affect planning?
- What organizational risks exist?
- What dependencies create risk?
- How are risks mitigated in the plan?
- What contingency exists?

C9: Knowledge and Learning

- What knowledge is required for each element?
- Where does expertise reside?
- What training is needed?
- How is knowledge transferred?
- What documentation supports handoff?

C10: Evolution and Maintenance

- Who maintains elements post-delivery?
- How does ownership evolve over time?
- What is the long-term team structure?
- How are enhancements assigned?
- What is the support model?

3.2 Concern-Quality Attribute Mapping

Table 3: Concern to Project Attribute Mapping

Concern	Schedule	Budget	Quality	Velocity	Predict.	Agility	Risk
Work Assignment	○	○	●	●	○	○	○
Team Structure	○	●	●	●	○	●	○
Sequencing	●	○	○	○	●	○	●
Estimation	●	●	–	○	●	○	○
Resources	●	●	○	●	○	○	○
Milestones	●	○	○	○	●	○	○
Coordination	○	○	●	○	○	○	●
Risk Mgmt	●	●	○	○	○	○	●
Knowledge	○	○	●	○	○	●	●
Evolution	○	●	●	○	○	●	○

● = Primary impact, ○ = Secondary impact, – = Minimal impact

4 Anti-Concerns

Understanding what the Planner's View is *not* appropriate for helps stakeholders avoid misapplying this viewpoint.

4.1 Out of Scope Topics

AC1: Detailed Technical Design

- Algorithm implementations
- Data structure choices
- API specifications
- Code-level patterns
- Technology selections (except as affects planning)

AC2: Runtime Architecture

- Process and thread design
- Performance characteristics
- Scalability mechanisms
- Deployment topology
- Operational behavior

AC3: Individual Task Management

- Daily task assignments
- Sprint-level planning details
- Individual developer schedules
- Bug tracking
- Code review assignments

AC4: HR and Personnel Details

- Salary information
- Performance reviews
- Career development
- Hiring decisions (except capacity)
- Personal information

AC5: Business Requirements

- Functional specifications
- User stories content
- Acceptance criteria
- Business rules
- Feature prioritization rationale

Common Misapplications

Avoid using the Planner's View for:

- Detailed sprint planning (use Agile tools)
- Technical design decisions (use Development Viewpoint)
- Runtime behavior analysis (use Process/C&C Viewpoints)
- Requirements management (use Requirements documents)
- Day-to-day task tracking (use project management tools)

5 Typical Stakeholders

The Planner's View serves stakeholders involved in project planning and coordination.

5.1 Primary Stakeholders

Table 4: Primary Stakeholder Analysis

Stakeholder	Role Description	Primary Interests
Project Managers	Plan and track delivery	Schedules, resources, milestones, risks, dependencies
Software Architects	Design system structure	Work breakdown alignment, team coordination points
Development Managers	Manage dev teams	Team assignments, capacity, skill requirements
Team Leads	Lead development teams	Team responsibilities, dependencies, coordination
Resource Managers	Allocate personnel	Staffing needs, skill matching, utilization
Program Managers	Coordinate programs	Cross-project dependencies, portfolio view

5.2 Secondary Stakeholders

Table 5: Secondary Stakeholder Analysis

Stakeholder	Role Description	Primary Interests
Senior Developers	Implement key elements	Understanding scope, dependencies, timeline
Product Managers	Define product direction	Feature delivery timing, release planning
Finance/PMO	Track budget and costs	Cost estimates, budget allocation, variance
Executive Sponsors	Fund and oversee project	High-level progress, risks, resource needs
QA Managers	Plan testing activities	Test planning, integration points, timelines
External Partners	Provide integrations	Integration timelines, coordination points

5.3 Stakeholder Concern Matrix

Table 6: Stakeholder-Concern Responsibility Matrix

	Assign.	Teams	Sequence	Estimate	Resource	Milest.	Coord.	Risk	Knowl.	Evolve
Project Mgr	A	C	R	A	R	R	A	R	C	C
Architect	R	C	R	C	C	C	R	R	A	C
Dev Manager	R	A	C	C	A	C	R	C	R	A
Team Lead	R	R	C	R	C	C	R	C	R	R
Resource Mgr	C	C	I	C	R	I	I	C	C	C
Sponsor	I	I	I	I	A	A	I	A	I	I

R = Responsible, A = Accountable, C = Consulted, I = Informed

6 Model Types

The Planner's View employs several complementary model types to capture different aspects of work assignment and planning.

6.1 Model Type Catalog

MT1: Work Assignment Matrix

- *Purpose:* Map architectural elements to responsible teams
- *Primary Elements:* Elements, teams, responsibility types
- *Key Relationships:* Owns, contributes-to, consumes

- *Typical Notation:* RACI matrices, ownership tables

MT2: Team Structure Diagram

- *Purpose:* Show team organization and relationships
- *Primary Elements:* Teams, roles, reporting lines
- *Key Relationships:* Reports-to, collaborates-with
- *Typical Notation:* Org charts, team topology diagrams

MT3: Development Timeline

- *Purpose:* Show project phases and milestones
- *Primary Elements:* Phases, milestones, deliverables
- *Key Relationships:* Precedes, gates, enables
- *Typical Notation:* Gantt charts, roadmaps

MT4: Dependency Graph

- *Purpose:* Show work item dependencies
- *Primary Elements:* Work items, dependencies
- *Key Relationships:* Depends-on, blocks, enables
- *Typical Notation:* Network diagrams, PERT charts

MT5: Resource Allocation Chart

- *Purpose:* Show resource distribution over time
- *Primary Elements:* Resources, allocations, time periods
- *Key Relationships:* Allocated-to, available-in
- *Typical Notation:* Resource histograms, allocation matrices

MT6: Team Interaction Matrix

- *Purpose:* Show required coordination between teams
- *Primary Elements:* Teams, interaction types
- *Key Relationships:* Coordinates-with, integrates-with
- *Typical Notation:* DSM (Design Structure Matrix), interaction maps

MT7: Risk Register

- *Purpose:* Document project risks and mitigations
- *Primary Elements:* Risks, impacts, mitigations
- *Key Relationships:* Affects, mitigates, triggers
- *Typical Notation:* Risk tables, risk matrices

MT8: Skill Matrix

- *Purpose:* Map required skills to team capabilities
- *Primary Elements:* Skills, teams, proficiency levels
- *Key Relationships:* Requires, possesses, lacks

- *Typical Notation:* Skill matrices, heat maps

MT9: Release Plan

- *Purpose:* Show release contents and timing
- *Primary Elements:* Releases, features, components
- *Key Relationships:* Includes, targets, depends-on
- *Typical Notation:* Release roadmaps, feature timelines

6.2 Model Type Relationships

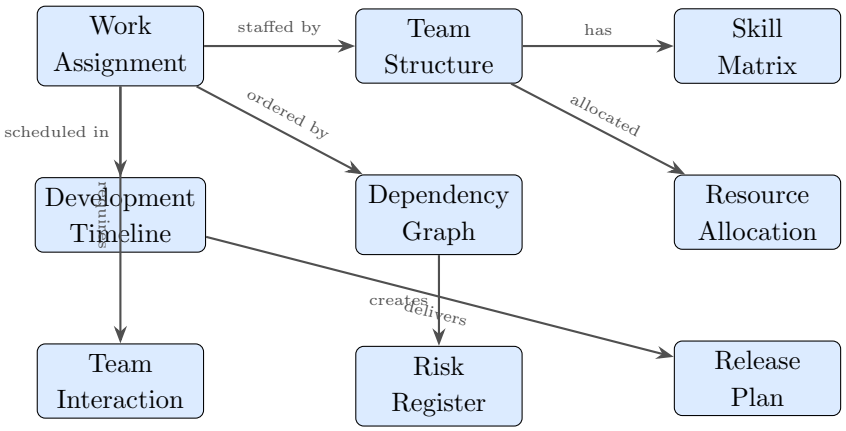


Figure 2: Model Type Dependency Relationships

7 Model Languages

For each model type, specific languages, notations, and techniques are prescribed.

7.1 Work Assignment Notation

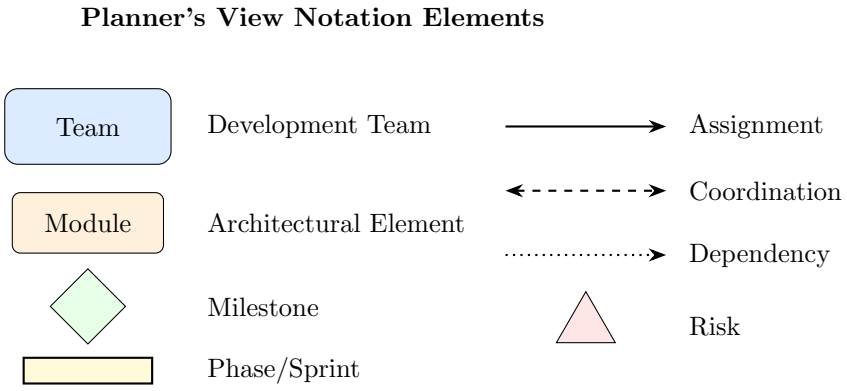


Figure 3: Planner's View Notation Legend

7.2 Team Topology Types

Table 7: Team Topology Classification (based on Team Topologies)

Team Type	Description	Characteristics
Stream-Aligned	Aligned to flow of work for a business domain	End-to-end ownership, fast flow, autonomous
Platform	Provides internal services to stream teams	Self-service, reduces cognitive load, enables
Enabling	Helps other teams overcome obstacles	Temporary, consultative, capability building
Complicated Subsystem	Handles complex specialized area	Deep expertise, shields complexity, specialist

7.3 Team Interaction Modes

Table 8: Team Interaction Mode Classification

Mode	Description	When to Use
Collaboration	Close work together	Discovery, innovation, solving complex problems
X-as-a-Service	One team provides, other consumes	Clear interface, platform capabilities
Facilitating	One team helps another	Capability building, obstacle removal

7.4 Responsibility Assignment (RACI)

Table 9: RACI Responsibility Types

Type	Description
Responsible (R)	Does the work. May be multiple people.
Accountable (A)	Ultimately answerable. Only one person.
Consulted (C)	Provides input before decisions. Two-way communication.
Informed (I)	Kept up-to-date. One-way communication.

7.5 Tabular Specifications

7.5.1 Work Assignment Table

Table 10: Example Work Assignment Matrix

Element	Team A	Team B	Team C	Platform	QA
Web Frontend	R/A	–	C	C	C
Mobile App	R/A	–	–	C	C
Order Service	I	R/A	C	C	C
User Service	C	R/A	–	C	C
API Gateway	C	C	R/A	R	C
Infrastructure	–	–	C	R/A	I

7.5.2 Team Specification Table

Table 11: Example Team Specification

Team	Type	Size	Key Skills	Ownership
Alpha	Stream-Aligned	6	React, TypeScript, UX	Frontend, Mobile
Beta	Stream-Aligned	7	Java, Spring, SQL	Orders, Users, Inventory
Gamma	Complicated Subsystem	4	ML, Python, Data	Recommendations, Analytics
Platform	Platform	5	K8s, AWS, Terraform	Infrastructure, CI/CD

7.5.3 Milestone Definition Table

Table 12: Example Milestone Definition

ID	Milestone	Target	Deliverables	Criteria
M1	Foundation	2024-Q1	Infrastructure, CI/CD, Auth	Env deployed, pipeline working
M2	Core MVP	2024-Q2	Orders, Users, Basic UI	E2E flow functional
M3	Beta Release	2024-Q3	Full features, Mobile	100 beta users
M4	GA Release	2024-Q4	Production ready	Performance, security certified

8 Viewpoint Metamodels

This section defines the conceptual metamodel underlying the Planner's View.

8.1 Core Metamodel

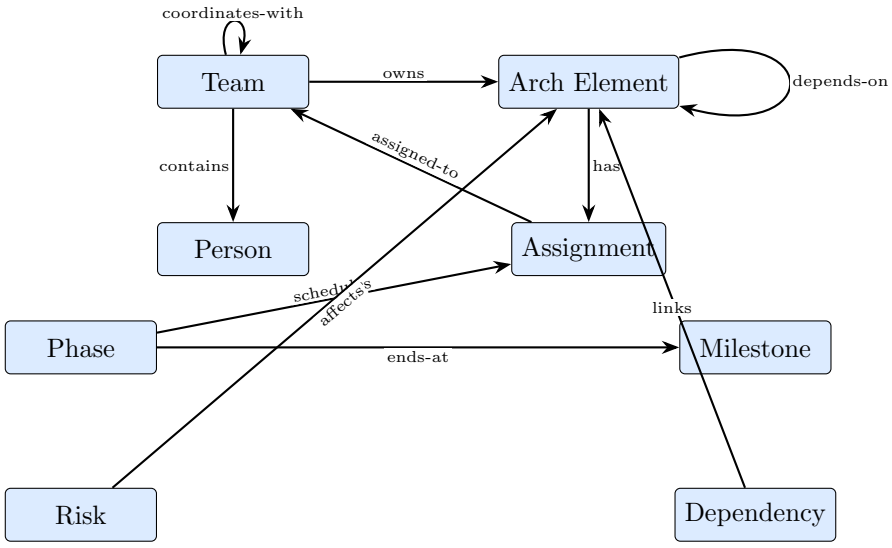


Figure 4: Planner's View Core Metamodel

8.2 Entity Definitions

Entity: Team

Definition: An organizational unit consisting of people who work together to develop and maintain assigned architectural elements.

Attributes:

- `teamId`: Unique identifier
- `name`: Team name
- `description`: Team purpose and mission
- `type`: Team type (stream-aligned, platform, enabling, complicated subsystem)
- `size`: Number of team members
- `lead`: Team lead/manager
- `members`: List of team members
- `skills`: Collective skills and capabilities
- `capacity`: Available capacity (story points, hours)
- `location`: Physical/virtual location
- `timezone`: Primary timezone

Constraints:

- Team size should be 4-9 members (two-pizza rule)
- Team should have clear ownership of elements
- Skills should match assigned work
- Team should minimize external dependencies

Entity: Architectural Element

Definition: A component of the architecture that requires development effort and has assigned ownership.

Attributes:

- `elementId`: Unique identifier
- `name`: Element name
- `description`: Element purpose
- `type`: Element type (module, service, component, subsystem)
- `complexity`: Complexity rating
- `size`: Estimated size/effort
- `priority`: Development priority
- `status`: Current status
- `dependencies`: Elements this depends on
- `interfaces`: Interfaces with other elements
- `requiredSkills`: Skills needed to develop

Constraints:

- Element should have single owning team
- Dependencies should be explicit
- Size estimates should be provided
- Priority should be assigned

Entity: Assignment

Definition: The allocation of responsibility for an architectural element to a team, including the type and extent of responsibility.

Attributes:

- `assignmentId`: Unique identifier
- `element`: Assigned architectural element
- `team`: Responsible team
- `responsibilityType`: Type (R/A/C/I or owner/contributor)
- `startDate`: When assignment begins
- `endDate`: When assignment ends (if temporary)
- `effort`: Estimated effort required
- `allocation`: Percentage of team capacity
- `notes`: Additional assignment details

Constraints:

- Each element should have exactly one Accountable team
- Allocation should not exceed team capacity
- Temporary assignments should have end dates

Entity: Person

Definition: An individual contributor assigned to a team with specific skills and capacity.

Attributes:

- `personId`: Unique identifier
- `name`: Person's name
- `role`: Job role/title
- `team`: Primary team assignment
- `skills`: Individual skills and proficiency
- `availability`: Available hours/percentage
- `startDate`: When person joins project
- `endDate`: When person leaves (if known)
- `location`: Work location
- `timezone`: Personal timezone

Constraints:

- Person should have primary team assignment
- Skills should be documented
- Availability should be realistic

Entity: Phase

Definition: A distinct period in the project timeline with specific goals, activities, and deliverables.

Attributes:

- `phaseId`: Unique identifier
- `name`: Phase name
- `description`: Phase purpose and goals
- `startDate`: Phase start date
- `endDate`: Phase end date
- `milestone`: Ending milestone
- `deliverables`: Expected deliverables
- `activities`: Key activities in phase
- `exitCriteria`: Criteria to complete phase
- `risks`: Phase-specific risks

Constraints:

- Phases should not overlap
- Each phase should have clear exit criteria
- Deliverables should be measurable

Entity: Milestone

Definition: A significant point in the project timeline marking completion of a major deliverable or achievement.

Attributes:

- `milestoneId`: Unique identifier
- `name`: Milestone name
- `description`: Milestone significance
- `targetDate`: Planned date
- `actualDate`: Actual achievement date
- `deliverables`: What is delivered
- `criteria`: Success criteria
- `dependencies`: Predecessor milestones
- `status`: Status (planned, at-risk, achieved, missed)
- `owner`: Accountable person

Constraints:

- Milestones should have clear criteria
- Dependencies should be explicit
- Owner should be assigned

Entity: Dependency

Definition: A relationship where one work item requires another to be completed first.

Attributes:

- `dependencyId`: Unique identifier
- `predecessor`: Work item that must complete first
- `successor`: Work item that depends on predecessor
- `type`: Dependency type (finish-to-start, start-to-start, etc.)
- `lag`: Time lag between items
- `strength`: How critical (hard, soft, preference)
- `description`: Nature of dependency
- `risk`: Risk if dependency is not met

Constraints:

- Dependencies should be acyclic
- Critical dependencies should be highlighted
- Risk should be assessed for hard dependencies

Entity: Risk

Definition: A potential event or condition that could negatively impact project objectives.

Attributes:

- `riskId`: Unique identifier
- `name`: Risk name
- `description`: Risk description
- `category`: Risk category (technical, resource, schedule, external)
- `probability`: Likelihood (low, medium, high)
- `impact`: Potential impact (low, medium, high)
- `exposure`: Risk exposure (probability × impact)
- `affectedElements`: Elements affected
- `mitigation`: Mitigation strategy
- `contingency`: Contingency plan
- `owner`: Risk owner
- `status`: Current status

Constraints:

- High exposure risks must have mitigation plans
- Risks should have owners
- Status should be regularly updated

8.3 Relationship Definitions

Table 13: Metamodel Relationship Definitions

Relationship	Source	Target	Description
owns	Team	Element	Team is responsible for element
contains	Team	Person	Team includes this person
has	Element	Assignment	Element has this work assignment
assigned-to	Assignment	Team	Assignment is given to team
schedules	Phase	Assignment	Phase includes this work
ends-at	Phase	Milestone	Phase concludes at milestone
affects	Risk	Element	Risk impacts this element
links	Dependency	Element	Dependency connects elements
coordinates	Team	Team	Teams must coordinate
depends-on	Element	Element	Element requires another

9 Conforming Notations

Several existing notations and tools support Planner's View modeling.

9.1 Gantt Charts

Gantt charts are the most common notation for project timeline visualization.

Elements: Tasks, durations, dependencies, milestones, resources.

Tool Support: Microsoft Project, Smartsheet, GanttPro, Jira.

Conformance Level: High for timeline, medium for architecture mapping.

9.2 Network Diagrams (PERT/CPM)

Network diagrams show activity dependencies and critical path.

Elements: Activities, dependencies, durations, slack.

Analysis: Critical path, float time, earliest/latest dates.

Conformance Level: High for dependency analysis.

9.3 Team Topologies Notation

Team Topologies provides notation for team structure and interactions.

Elements: Team types, interaction modes, cognitive load.

Tool Support: Team Topologies diagrams, Miro templates.

Conformance Level: High for team structure modeling.

9.4 Design Structure Matrix (DSM)

DSM shows dependencies between elements in matrix form.

Elements: Elements, dependencies (marked cells).

Analysis: Clustering, partitioning, tearing.

Conformance Level: High for dependency and coordination analysis.

9.5 Notation Comparison

Table 14: Planning Notation Comparison

Feature	Gantt	PERT	DSM	RACI	Team Top.	Roadmap
Timeline	●	○	–	–	–	●
Dependencies	○	●	●	–	○	○
Resources	●	○	–	●	○	–
Milestones	●	●	–	–	–	●
Team structure	○	–	○	○	●	–
Coordination	–	–	●	○	●	–
Work assignment	○	–	○	●	○	–

● = Strong support, ○ = Limited support, – = Not applicable

10 Model Correspondence Rules

Model correspondence rules define how elements in planner's models relate to elements in other architectural views.

10.1 Development View Correspondence

Correspondence Rule CR-01: Module to Team Mapping

Rule: Every code module should have an owning team assignment.
Formal Expression:

$$\forall m \in Modules : \exists t \in Teams : owns(t, m)$$

Rationale: Ensures clear ownership and accountability for code.
Verification: Module ownership matrix review.

Correspondence Rule CR-02: Package Alignment

Rule: Package structure should align with team boundaries where possible.
Formal Expression:

$$packages(team_1) \cap packages(team_2) \approx \emptyset$$

Rationale: Reduces coordination overhead (Conway's Law).
Verification: Package-to-team mapping review.

10.2 Logical View Correspondence

Correspondence Rule CR-03: Domain to Team Mapping

Rule: Bounded contexts should map to team ownership.

Formal Expression:

$$\forall bc \in BoundedContexts : \exists t \in Teams : owns(t, bc)$$

Rationale: Aligns team structure with domain structure (inverse Conway).

Verification: Domain-to-team mapping review.

10.3 Component-and-Connector View Correspondence

Correspondence Rule CR-04: Interface to Coordination

Rule: Interfaces between components owned by different teams require coordination.

Formal Expression:

$$\forall i \in Interfaces : owner(source(i)) \neq owner(target(i)) \Rightarrow coordinates(owner(source(i)), owner(target(i)))$$

Rationale: Ensures necessary team coordination is planned.

Verification: Team interaction matrix completeness.

11 Operations on Views

This section defines methods for creating, interpreting, analyzing, and maintaining planner's views.

11.1 Creation Methods

11.1.1 View Development Process

Step 1: Identify Architectural Elements

1. Extract elements from development/logical views
2. Determine granularity for assignment
3. Identify element dependencies
4. Estimate element complexity and size
5. Prioritize elements for development

Step 2: Define Team Structure

1. Determine required team types
2. Size teams appropriately (5-9 members)
3. Identify required skills per team
4. Assign team leads
5. Define team missions and boundaries

Step 3: Create Work Assignments

1. Map elements to teams
2. Assign RACI responsibilities
3. Balance workload across teams
4. Identify skill gaps
5. Document assignment rationale

Step 4: Sequence Development

1. Analyze architectural dependencies
2. Identify critical path
3. Determine parallel work opportunities
4. Plan integration points
5. Create dependency graph

Step 5: Define Phases and Milestones

1. Identify major project phases
2. Define milestones and criteria
3. Allocate work to phases
4. Set target dates
5. Plan deliverables per phase

Step 6: Plan Team Coordination

1. Identify inter-team interfaces
2. Define coordination mechanisms
3. Plan integration activities
4. Schedule synchronization points
5. Document communication protocols

Step 7: Assess and Mitigate Risks

1. Identify technical and organizational risks
2. Assess probability and impact
3. Develop mitigation strategies
4. Create contingency plans
5. Assign risk owners

11.1.2 Team Organization Patterns

Pattern: Stream-Aligned Teams

Context: Need fast flow of features to production.

Solution: Organize teams around end-to-end value streams.

Characteristics:

- Full ownership from idea to production
- Cross-functional (dev, test, ops skills)
- Aligned with business domain
- Minimized handoffs

Use When: Optimizing for delivery speed and autonomy.

Pattern: Platform Teams

Context: Multiple teams need common capabilities.

Solution: Dedicated team provides self-service platform.

Characteristics:

- Provides infrastructure, tools, capabilities
- Self-service consumption model
- Reduces cognitive load on stream teams
- Enables rather than gates

Use When: Shared infrastructure or capabilities needed.

Pattern: Inverse Conway Maneuver

Context: Architecture and team structure misaligned.

Solution: Restructure teams to match desired architecture.

Characteristics:

- Architecture drives team structure
- Teams own bounded contexts
- Clear interfaces between teams
- Reduced coordination overhead

Use When: Designing new architecture, restructuring organization.

11.2 Analysis Methods

11.2.1 Critical Path Analysis

Critical Path Method (CPM)

Purpose: Identify the longest sequence of dependent activities.

Process:

1. List all activities and durations
2. Identify dependencies
3. Calculate forward pass (earliest start/finish)
4. Calculate backward pass (latest start/finish)
5. Identify zero-float activities (critical path)

Output: Critical path, activity float, project duration.

Application: Focus resources on critical path activities.

11.2.2 Team Dependency Analysis

Team Interaction Assessment

Purpose: Analyze coordination requirements between teams.

Factors:

- Number of shared interfaces
- Frequency of coordination needed
- Synchronization requirements
- Communication overhead

Output: Team interaction matrix with coordination intensity.

Application: Identify high-coordination pairs for attention.

12 Examples

12.1 Example 1: Work Assignment Diagram

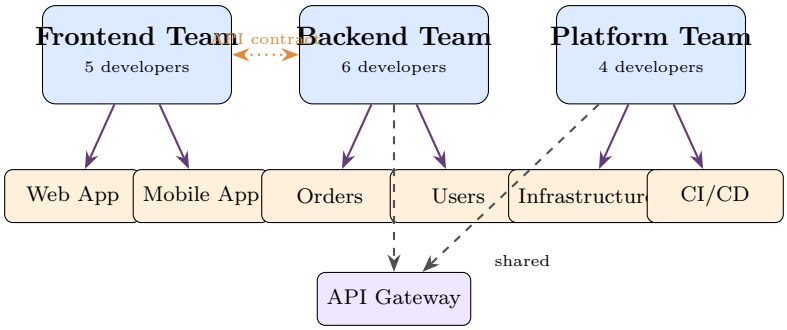


Figure 5: Work Assignment Diagram

Description: This diagram shows three teams with their assigned modules. Frontend Team owns Web and Mobile apps. Backend Team owns Orders and Users services. Platform Team owns Infrastructure and CI/CD. The API Gateway is shared between Backend and Platform teams. A coordination relationship exists between Frontend and Backend teams for API contracts.

12.2 Example 2: Development Timeline

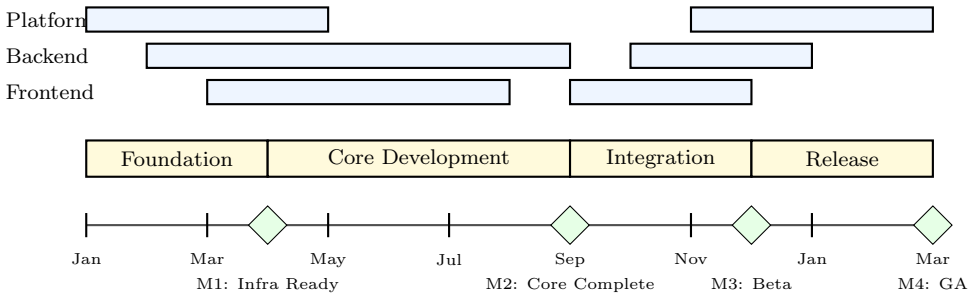


Figure 6: Development Timeline with Team Tracks

Description: This timeline shows the project phases (Foundation, Core Development, Integration, Release) with team activity tracks. Platform Team is active early for infrastructure and late for production readiness. Backend Team has the longest development phase. Frontend Team starts after API design and completes during integration. Milestones mark key completion points.

12.3 Example 3: Risk Register

Table 15: Sample Risk Register

ID	Risk	Prob.	Impact	Exp.	Mitigation	Owner
R1	Key developer leaves	Med	High	High	Cross-training, documentation	Dev Manager
R2	External API delayed	Med	Med	Med	Build mock, parallel dev	Architect
R3	Performance issues	Low	High	Med	Early perf testing, budgets	Tech Lead
R4	Scope creep	High	Med	High	Change control, PM rigor	PM

13 Notes

13.1 Conway's Law Considerations

Conway's Law

"Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations." – Melvin Conway, 1967

Implications for Planning:

- Architecture and team structure influence each other
- Align teams with desired architectural boundaries
- Reduce interfaces between teams to reduce component interfaces
- Use "Inverse Conway Maneuver" to drive architecture through team design
- Cross-team interfaces require coordination overhead

13.2 Estimation Guidelines

Estimation Best Practices

- Use relative sizing (story points) for comparison
- Include uncertainty ranges (optimistic/expected/pessimistic)
- Account for unknowns with contingency
- Re-estimate as understanding improves
- Track actuals vs estimates for calibration
- Involve the team doing the work
- Consider dependencies in estimates
- Factor in integration and testing time

13.3 Common Pitfalls

Common Mistakes to Avoid

1. **Ignoring Conway's Law:** Misaligned teams and architecture
2. **Over-allocation:** Assigning more work than capacity allows
3. **Hidden Dependencies:** Missing coordination requirements
4. **Single Points of Failure:** Critical knowledge in one person
5. **Optimistic Estimation:** Not accounting for uncertainty
6. **Waterfall in Disguise:** Big-bang integration at end
7. **Skill Mismatch:** Assigning work without needed skills
8. **Communication Overhead:** Too many cross-team interfaces

14 Sources

14.1 Primary References

1. Clements, P., et al. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley Professional.
2. Skelton, M., & Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press.
3. Conway, M. (1968). "How Do Committees Invent?" *Datamation*, 14(4), 28-31.
4. Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley Professional.
5. Brooks, F. (1995). *The Mythical Man-Month* (Anniversary ed.). Addison-Wesley Professional.

14.2 Supplementary References

6. McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*. Microsoft Press.
7. DeMarco, T., & Lister, T. (2013). *Peopleware* (3rd ed.). Addison-Wesley Professional.
8. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate*. IT Revolution Press.
9. Reinertsen, D. (2009). *The Principles of Product Development Flow*. Celeritas Publishing.
10. PMI. (2021). *A Guide to the Project Management Body of Knowledge* (7th ed.).

14.3 Online Resources

- Team Topologies: `https://teamtopologies.com/`
- Agile Estimation: `https://www.mountaingoatsoftware.com/`
- Conway's Law: `https://www.melconway.com/Home/Conways_Law.html`
- DORA Research: `https://dora.dev/`

A Planner's View Checklist

Item	Complete?
Work Assignment	
All architectural elements have owners	☐
RACI responsibilities defined	☐
Workload balanced across teams	☐
Assignment rationale documented	☐
Team Structure	
Teams appropriately sized	☐
Team types defined (stream, platform, etc.)	☐
Skills match assignments	☐
Team leads identified	☐
Timeline	
Phases defined with criteria	☐
Milestones identified	☐
Critical path analyzed	☐
Dependencies mapped	☐
Coordination	
Inter-team interfaces identified	☐
Coordination mechanisms defined	☐
Integration points planned	☐
Communication protocols documented	☐
Risk Management	
Risks identified and assessed	☐
Mitigations defined	☐
Risk owners assigned	☐
Contingency plans created	☐

B Glossary

Allocation Assignment of resources to work items.

Capacity Available work capacity of a team or person.

Conway's Law Principle that system design mirrors organization structure.

Critical Path Longest sequence of dependent activities.

Dependency Relationship where one item requires another.

Float/Slack Time an activity can be delayed without affecting schedule.

Milestone Significant point marking project progress.

Phase Distinct period with specific goals and activities.

Platform Team
: Team providing shared capabilities to other teams.

RACI
: Responsibility matrix (Responsible, Accountable, Consulted, Informed).

Stream-Aligned
: Team aligned to business value stream.

Team Topology
: Pattern for organizing teams.

Work Assignment
: Allocation of architectural element to team.

Work Breakdown
: Decomposition of work into assignable units.