Diagram-Driven Reading Sequence

A Strategic Guide for AppSec Teams
Documenting and Improving Security Processes

Objective: Maximize business value per stakeholder through systematic reading and diagram production

Reading List:

Thinking in Systems	Donella Meadows
The E-Myth Revisited	Michael Gerber
Work the System	Sam Carpenter
Service Design Doing	Stickdorn et al.
Traction	Gino Wickman
Scaling Up	Verne Harnish
Modern Software Engineering	Dave Farley
The Mythical Man-Month	Fred Brooks
Practical Object-Oriented Design	Sandi Metz

Document Type:	Strategic Reading Guide
Target Audience:	AppSec Teams & Leadership
Focus Areas:	Intake, Threat Modeling, Vulnerability Management, Exceptions, CI/CD Gates, Reporting
Methodology:	Diagram-Driven Learning

Application Security Program

Version 1.0

Contents

1 Executive Summary

This document presents a strategically sequenced reading program designed to help Application Security (AppSec) teams systematically document and improve their processes. The sequence is optimized to maximize business value per stakeholder by pairing each book with specific diagram deliverables that address real organizational needs.

1.1 Core Philosophy

The reading sequence follows a deliberate progression:

Why → What → How → Scale → Sustain

Each book builds upon the previous, creating a coherent body of knowledge that translates directly into actionable architecture documentation.

1.2 Value Proposition

This approach delivers three key benefits:

1. **Purposeful Reading:** Every book connects to specific diagram outputs, ensuring knowledge translates to artifacts.
2. **Stakeholder Alignment:** Diagrams are mapped to stakeholder concerns, ensuring relevance and adoption.
3. **Progressive Complexity:** The sequence builds from foundational concepts to tactical implementation.

2 Reading Sequence Overview

The nine-book sequence is organized into three phases, each serving distinct organizational needs:

#	Book	Phase	Core Outcome
1	*Thinking in Systems*	Foundation	System behavior understanding
2	*The E-Myth Revisited*	Foundation	Standardization mindset
3	*Work the System*	Foundation	Process documentation
4	*Service Design Doing*	Design	Service optimization
5	*Traction*	Governance	Operating cadence
6	*Scaling Up*	Governance	Scalable execution
7	*Modern Software Engineering*	Implementation	DevSecOps integration
8	*The Mythical Man-Month*	Implementation	Coordination management
9	*Practical OO Design*	Implementation	Technical architecture

Reading Time Investment

- **Phase 1 (Foundation):** 4–6 weeks → Establishes "why" and "what"
- **Phase 2 (Design & Governance):** 4–6 weeks → Establishes "how" and "who"
- **Phase 3 (Implementation):** 4–6 weeks → Establishes technical execution

Total Program Duration: 12–18 weeks for full sequence with diagram production

3 Phase 1: Foundation

The foundation phase establishes the conceptual framework for understanding AppSec as a system, the importance of standardization, and the mechanics of process documentation.

3.1 Book 1: Thinking in Systems (Meadows)

"How the AppSec System Behaves"

Use this book first to avoid diagramming symptoms instead of causes. It provides the framing for feedback loops, delays, unintended consequences, and system resilience.

Key Concepts to Extract:

- Stocks, flows, and feedback loops
- System boundaries and interfaces
- Leverage points for intervention
- Delays and their effects on behavior
- Resilience vs. brittleness in systems

3.1.1 Diagrams to Produce

High Business Value — Produce Early

System Context + Boundary Diagram

Shows AppSec as a service within the broader Engineering ecosystem. Defines what is inside vs. outside the AppSec system boundary, identifying key interfaces with CI/CD, ticketing, CMDB, IAM, and GRC systems.

Causal Loop Diagrams

Visualize recurring pain points and their systemic causes. Example: "Scanner noise → developer frustration → ignored findings → increased risk → more scanners deployed → more noise." These diagrams reveal reinforcing and balancing loops that drive organizational behavior.

Capability Map

A high-level view of AppSec capabilities: Secure SDLC, Vulnerability Management, Security Reviews, Exception Management, Metrics & Reporting. This establishes the "what" before diving into the "how."

3.1.2 Primary Stakeholders Served

Executives/BOD Understand why change is needed and where investment yields returns

Eng Leadership See systemic causes of friction, not just symptoms

GRC/Audit Appreciate control effectiveness within system dynamics

3.1.3 Application to AppSec Processes

AppSec Process	Systems Thinking Application
Intake	Model request queue as a stock; identify what increases/decreases it
Vulnerability Mgmt	Map feedback loops between finding volume and remediation capacity
Exceptions	Understand accumulation of risk debt over time
CI/CD Gates	Analyze delays between detection and developer feedback
Reporting	Connect leading indicators to lagging outcomes

3.2 Book 2: The E-Myth Revisited (Gerber)

"Why Standardization and Repeatable Systems Matter"

This book provides the foundation for treating AppSec work as a set of services with standardized operating procedures, not ad-hoc heroics. It emphasizes working *on* the system rather than *in* it.

Key Concepts to Extract:

- The franchise prototype model
- Working on the business vs. in the business
- Systemization of every repeatable task
- The importance of documented procedures
- Predictability through standardization

3.2.1 Diagrams to Produce

Core Deliverables

AppSec Service Catalog

A comprehensive inventory of AppSec services including: service name, description, entry criteria (what triggers the service), required inputs, outputs/deliverables, SLAs (response time, completion time), escalation paths, and service owner. This becomes the "menu" that stakeholders use to engage AppSec.

High-Level Operating Model

A visual representation of who does what, when, and why. Shows the relationship between AppSec roles (security engineers, architects, managers) and their responsibilities across different service types. Establishes clear ownership and accountability.

3.2.2 Primary Stakeholders Served

Executives See AppSec as a predictable, manageable function

AppSec Leadership Gain tools for capacity planning and resource allocation

Eng Managers Understand what to expect and how to engage AppSec

3.2.3 Service Catalog Structure

Service	Entry Criteria	Output	SLA
Security Review	Design doc ready	Findings report	5 business days
Threat Model	Architecture defined	Threat register	10 business days
Vuln Triage	Finding submitted	Priority assigned	24 hours
Exception Request	Justification provided	Approval/denial	48 hours
Pen Test	Scope defined	Test report	2–4 weeks

3.3 Book 3: Work the System (Carpenter)

"Document the Way Work Actually Happens"

With the mindset established, this book provides execution mechanics for capturing procedures and improving them. It emphasizes separating yourself from the work to see processes objectively.

Key Concepts to Extract:

- Strategic objective, operating principles, and working procedures
- The "outside and slightly elevated" perspective
- Procedure documentation as a management tool
- Continuous refinement of processes
- Measuring process effectiveness

3.3.1 Diagrams to Produce

Core AppSec Process Set — Immediate Friction Reduction

BPMN Swimlane: AppSec Intake
Request submission → triage → categorization → routing → assignment → closure. Shows handoffs between requesters, AppSec triage, and assigned engineers.

BPMN Swimlane: Secure Design Review / Threat Modeling
Engagement request → scoping → context gathering → threat enumeration → mitigation identification → documentation → sign-off → tracking.

BPMN Swimlane: Vulnerability Management
Discovery → deduplication → triage → assignment → remediation → verification → closure. Includes false positive handling and escalation paths.

BPMN Swimlane: Exception/Waiver Process
Request → justification review → compensating controls assessment → risk acceptance decision → documentation → expiry management.

RACI Overlays
For each BPMN diagram, add RACI (Responsible, Accountable, Consulted, Informed) designations to every step. This provides immediate clarity on ownership.

3.3.2 Primary Stakeholders Served

AppSec Team Clear procedures reduce ambiguity and enable onboarding
Developers Know exactly what to expect at each step
SRE/Ops Understand their role in vulnerability remediation
Eng Managers Can predict timelines and plan capacity

3.3.3 RACI Matrix Example: Vulnerability Management

Activity	**AppSec**	**Dev Team**	**Eng Mgr**	**SRE**	**GRC**
Discovery/Scanning	R/A	I	I	C	I
Triage & Prioritization	R/A	C	I	C	I
Assignment	R	A	R	I	I
Remediation	C	R/A	I	C	I
Verification	R/A	I	I	C	I
Closure	R/A	I	I	I	I
Exception Approval	R	C	C	I	A

4 Phase 2: Design & Governance

The second phase focuses on optimizing AppSec as a consumable internal service and establishing governance structures that scale.

4.1 Book 4: Service Design Doing (Stickdorn et al.)

"Design AppSec as a Consumable Internal Service"

Once you can document current state, use service design to improve it end-to-end. This book is explicitly process/design oriented and diagram-heavy, focusing on handoffs, touchpoints, and wait states.

Key Concepts to Extract:

- Service blueprinting methodology
- Customer journey mapping
- Touchpoint analysis
- Frontstage vs. backstage activities
- Moments of truth identification

4.1.1 Diagrams to Produce

Service Optimization Artifacts

Service Blueprints for AppSec Services

Multi-layer diagrams showing: (1) customer actions (developer touchpoints), (2) frontstage interactions (visible AppSec activities), (3) backstage activities (invisible AppSec work), (4) support processes (tooling, systems), and (5) physical evidence (artifacts produced). Create one blueprint per major service.

Customer Journey Maps

Visualize the developer experience when engaging AppSec from initial need through resolution. Identify pain points, emotional states, and opportunities for improvement. Focus on: awareness, engagement, waiting, resolution, and follow-up phases.

Value Stream Maps

Map the flow of work from vulnerability discovery to remediation completion. Identify: process time (actual work), wait time (queues, delays), and cycle time (total elapsed time). Highlight where time is lost and improvement opportunities.

4.1.2 Primary Stakeholders Served

Developers Experience less friction, clearer expectations
Eng Productivity Reduced toil and faster throughput
AppSec Leadership Data-driven service improvement

4.1.3 Service Blueprint Layers

Layer	Description
Customer Actions	What the developer does (submit request, provide info, implement fix)
Frontstage	Visible AppSec activities (respond to request, provide guidance)
Line of Visibility	—
Backstage	Invisible AppSec work (triage, research, tooling configuration)
Support Processes	Systems and tools (scanners, ticketing, reporting dashboards)

4.2 Book 5: Traction (Wickman)

"Turn Diagrams into an Operating Cadence"

With services and processes defined, Traction helps institutionalize accountability and metrics. It provides a practical operating system for running the AppSec function.

Key Concepts to Extract:

- The Entrepreneurial Operating System (EOS)
- Accountability charts (not org charts)
- Rocks (quarterly priorities)
- Scorecards with leading/lagging indicators
- Level 10 meeting structure
- Issues, Discuss, Solve (IDS) methodology

4.2.1 Diagrams to Produce

Governance That Doesn't Feel Bureaucratic

Accountability Chart

A function-based view of security ownership. Unlike org charts (which show reporting relationships), accountability charts show who owns what outcomes. Map: AppSec leadership, vulnerability management, security architecture, tooling/automation, metrics/reporting, and exception governance.

Scorecard Diagram

Visual representation of key metrics: leading indicators (scan coverage, review requests, training completion), lagging indicators (MTTR, open critical vulns, exception count), targets, and owners. Design for weekly review cadence.

Meeting/Decision Cadence Map

Document the rhythm of governance: daily standups, weekly tactical meetings, monthly leadership reviews, quarterly planning sessions. Map decision rights: who can approve exceptions, who escalates blockers, who reviews metrics.

4.2.2 Primary Stakeholders Served

Executives	Clear accountability and measurable progress
Eng Leadership	Predictable cadence for engagement
AppSec Manager	Structured operating rhythm

4.2.3 Sample Scorecard Structure

Metric	**Target**	**This Week**	**Trend**	**Owner**
Critical vulns open >30d	<5	3	↓	Vuln Mgmt Lead
MTTR (Critical)	<7d	5.2d	↓	Vuln Mgmt Lead
Security reviews completed	>10/wk	12	→	Security Architect
Exception requests pending	<3	2	↓	AppSec Manager
Scan coverage (%)	>95%	97%	↑	Tooling Lead

4.3 Book 6: Scaling Up (Harnish)

"Make It Scale Without Breaking"

Read after Traction to expand the operating system into scalable execution and cross-team alignment. Focuses on the four decisions: People, Strategy, Execution, and Cash.

Key Concepts to Extract:

- One-page strategic plan
- Meeting rhythm (daily, weekly, monthly, quarterly, annually)
- Rockefeller Habits checklist
- KPI cascading and alignment
- Cross-functional process ownership

4.3.1 Diagrams to Produce

Scalable Execution Artifacts

Operating Rhythm Map

Visual calendar showing the hierarchy of planning and review cycles: annual strategic planning → quarterly rock-setting → monthly metric reviews → weekly tactical meetings → daily standups. Shows how information flows up and decisions flow down.

KPI Tree

Hierarchical diagram connecting AppSec operational metrics to engineering outcomes to business results. Example: Scan coverage → Finding detection rate → Vulnerability exposure → Security incidents → Business risk posture.

Organizational Interface Map

Shows how AppSec connects with adjacent functions: Platform Engineering (tooling integration), Product Engineering (feature security), GRC (compliance alignment), IT/Ops (infrastructure security). Defines integration points and shared responsibilities.

4.3.2 Primary Stakeholders Served

Executives See how AppSec connects to business outcomes

Portfolio/Program Mgmt Understand cross-functional dependencies

Eng Leadership Clear integration points and expectations

4.3.3 Operating Rhythm Example

Cadence	Duration	Focus
Daily	15 min	Blockers, priorities, quick wins
Weekly	60–90 min	Scorecard review, IDS on issues, rock progress
Monthly	2–3 hours	Metric deep-dives, process improvements, staffing
Quarterly	Full day	Rock-setting, strategic alignment, retrospective
Annually	2 days	Strategy refresh, annual planning, team development

5 Phase 3: Implementation

The final phase translates business-process thinking into technical architecture and execution, ensuring AppSec integrates seamlessly with engineering practices.

5.1 Book 7: Modern Software Engineering (Farley)

"Align AppSec to Fast, Reliable Delivery"

This book translates business-process diagrams into SDLC/DevSecOps system diagrams that engineering teams respect. It emphasizes empiricism, feedback, and continuous improvement.

Key Concepts to Extract:

- Continuous delivery as an engineering discipline
- Feedback loops and cycle time optimization
- Deployment pipelines and quality gates
- Testability and observability
- Managing complexity through modularity

5.1.1 Diagrams to Produce

DevSecOps Integration Artifacts

Secure SDLC Value Stream
End-to-end visualization: Idea → Design → Code → Build → Test → Deploy → Operate. Overlay security activities at each stage: threat modeling (design), SAST/SCA (code/build), DAST (deploy), runtime protection (operate).

CI/CD Security Gates Diagram
Pipeline architecture showing: where security checks execute, what triggers them, pass/fail criteria, blocking vs. warning behaviors, SLAs for feedback, and override mechanisms. Include SAST, SCA, secrets scanning, IaC scanning, and container scanning.

Toolchain Data-Flow Diagram
Signal flow from source to action: Scanners → Findings aggregation → Deduplication → Enrichment → Ticket creation → Assignment → Remediation tracking → Verification → Reporting. Shows integration points between tools.

5.1.2 Primary Stakeholders Served

Engineering Org	See security as part of delivery, not a blocker
Platform/DevOps	Clear integration requirements
AppSec Engineers	Technical architecture for tooling work

5.1.3 CI/CD Gate Configuration Example

Check	**Stage**	**Block Criteria**	**Warn Criteria**	**SLA**
SAST	Build	Critical findings	High findings	<5 min
SCA	Build	Critical CVE	High CVE	<3 min
Secrets	Pre-commit	Any detection	—	<30 sec
IaC Scan	Build	Critical misconfig	High misconfig	<2 min
Container	Build	Critical CVE	High CVE	<5 min

5.2 Book 8: The Mythical Man-Month (Brooks)

"Avoid Coordination Failures in Security Initiatives"

This classic helps prevent predictable failure modes when rolling out process improvements across teams. Its insights on communication overhead and project management remain highly relevant.

Key Concepts to Extract:

- Brooks's Law: Adding people to a late project makes it later
- Communication overhead grows quadratically
- The surgical team model
- Conceptual integrity in design
- The second-system effect
- Plan to throw one away

5.2.1 Diagrams to Produce

Coordination and Rollout Artifacts

Communication and Handoff Map

Visualize where coordination costs appear in AppSec processes. Identify high-overhead touchpoints: cross-team handoffs, approval chains, information requests. Use this to simplify communication paths and reduce coordination burden.

Program Rollout Plan Diagram

Phased adoption plan for new processes or tools: pilot teams → early adopters → majority rollout → laggards. Include feedback loops at each phase, success criteria for progression, and rollback triggers. Map by repository criticality or team readiness.

5.2.2 Primary Stakeholders Served

Program/Project Leadership Realistic rollout planning
Eng Leadership Understanding of coordination costs
AppSec Lead Change management strategy

5.2.3 Rollout Phase Model

Phase	Coverage	Activities
Pilot	2–3 teams	Validate process, gather feedback, refine tooling
Early Adopters	10–15%	Expand to willing teams, document patterns
Majority	50–80%	Standardized rollout, training at scale
Laggards	Remaining	Targeted support, exception handling

5.3 Book 9: Practical Object-Oriented Design (Metz)

"Make Technical Diagrams Match Maintainable Systems"

This tactical book improves the quality of architecture and module diagrams underlying AppSec tooling and integrations. It ensures technical implementations remain maintainable.

Key Concepts to Extract:

- Single Responsibility Principle
- Dependency management and injection
- Interface design and duck typing
- Composition over inheritance
- Managing change through good design

5.3.1 Diagrams to Produce

Technical Architecture Artifacts

Module Decomposition for AppSec Services/Tooling

Break down AppSec technical components into well-defined modules with clear responsibilities: Scanner integration modules, Finding normalization, Deduplication engine, Ticket sync adapters, Reporting aggregators, API interfaces. Each module should have a single responsibility.

Integration/Adapter Diagrams

Document boundaries between AppSec systems and external tools: Scanner adapters (abstract different SAST/SCA tools), CI adapters (GitHub Actions, GitLab CI, Jenkins), Ticketing adapters (Jira, ServiceNow), Reporting adapters (dashboards, exports). Show how adapters isolate change.

5.3.2 Primary Stakeholders Served

AppSec Engineers Clear technical architecture for implementation

Platform Teams Integration patterns and expectations

5.3.3 Module Responsibility Matrix

Module	Responsibility	Dependencies
Scanner Adapter	Normalize findings from various tools	Finding Schema
Deduplication Engine	Identify and merge duplicate findings	Finding Store
Enrichment Service	Add context (asset, owner, criticality)	CMDB Adapter
Ticket Sync	Bidirectional sync with ticketing systems	Ticketing Adapter
Reporting Aggregator	Compile metrics and generate reports	Finding Store, Config

6 Stakeholder-to-Book Value Alignment

This section provides a quick reference for mapping stakeholder needs to the reading sequence.

6.1 Alignment Matrix

Stakeholder Group	Priority Reading Path
Executives / BOD	*Thinking in Systems → Traction → Scaling Up* *Focus: Risk narrative, governance structures, measurable outcomes*
Engineering Leadership	*Work the System → Modern Software Engineering → Service Design Doing* *Focus: Flow efficiency, bottleneck reduction, clear ownership*
Developers	*Service Design Doing → Work the System* *Focus: Clear intake process, predictable expectations, reduced toil*
GRC / Audit	*Thinking in Systems → Work the System* *Focus: Controls as process, evidence flows, defensible repeatability*
AppSec Team	Full sequence in order *Focus: Complete progression from "why" to "how" to "scale"*

6.2 Diagram-to-Stakeholder Mapping

Diagram Type	Exec	Eng Lead	Dev	GRC	AppSec
System Context	•	•		•	•
Causal Loop	•	•		•	•
Service Catalog	•	•	•	•	•
BPMN Workflows		•	•	•	•
Service Blueprints		•	•		•
Value Stream Maps		•		•	•
Accountability Chart	•	•		•	•
Scorecard	•	•		•	•
KPI Tree	•	•		•	•
CI/CD Gates		•	•		•
Toolchain Data-Flow		•			•
Module Decomposition					•

7 Implementation Guidance

7.1 Execution Timeline

Recommended Pacing

Weeks 1–2 Read *Thinking in Systems*; produce system context and causal loop diagrams
Weeks 3–4 Read *The E-Myth Revisited*; produce service catalog draft
Weeks 5–6 Read *Work the System*; produce core BPMN workflows with RACI
Weeks 7–8 Read *Service Design Doing*; produce service blueprints and journey maps
Weeks 9–10 Read *Traction*; produce accountability chart and scorecard
Weeks 11–12 Read *Scaling Up*; produce KPI tree and operating rhythm
Weeks 13–14 Read *Modern Software Engineering*; produce CI/CD gate diagrams
Weeks 15–16 Read *The Mythical Man-Month*; produce rollout plan
Weeks 17–18 Read *Practical OO Design*; produce module decomposition

7.2 Success Criteria

Each phase should produce tangible, reviewed artifacts:

Phase	Exit Criteria
Foundation	System context approved by leadership; service catalog published; core BPMN workflows documented
Design & Governance	Service blueprints validated with developers; scorecard in weekly use; operating rhythm established
Implementation	CI/CD gates deployed to pilot; rollout plan approved; technical architecture documented

7.3 Common Pitfalls to Avoid

- **Diagram without purpose:** Every diagram should serve a specific stakeholder concern
- **Perfect before publish:** Start with draft diagrams and iterate based on feedback
- **Tools before process:** Document the process before automating it
- **Big bang rollout:** Phase implementations to gather feedback and adjust
- **Ignoring tribal knowledge:** Interview practitioners to capture actual workflows

Appendix A: Complete Diagram Inventory

#	Diagram	Source Book	Type
1	System Context + Boundary	Thinking in Systems	C4 L1
2	Causal Loop Diagrams	Thinking in Systems	Systems
3	Capability Map	Thinking in Systems	Capability
4	AppSec Service Catalog	E-Myth Revisited	Catalog
5	High-Level Operating Model	E-Myth Revisited	Operating Model
6	Intake BPMN	Work the System	BPMN
7	Threat Modeling BPMN	Work the System	BPMN
8	Vulnerability Mgmt BPMN	Work the System	BPMN
9	Exception Process BPMN	Work the System	BPMN
10	RACI Overlays	Work the System	RACI
11	Service Blueprints	Service Design Doing	Blueprint
12	Customer Journey Maps	Service Design Doing	Journey Map
13	Value Stream Maps	Service Design Doing	VSM
14	Accountability Chart	Traction	Org
15	Scorecard Diagram	Traction	Metrics
16	Meeting/Decision Cadence	Traction	Cadence
17	Operating Rhythm Map	Scaling Up	Cadence
18	KPI Tree	Scaling Up	Metrics
19	Org Interface Map	Scaling Up	Integration
20	Secure SDLC Value Stream	Modern Software Eng	VSM
21	CI/CD Security Gates	Modern Software Eng	Pipeline
22	Toolchain Data-Flow	Modern Software Eng	DFD
23	Communication/Handoff Map	Mythical Man-Month	Communication
24	Program Rollout Plan	Mythical Man-Month	Rollout
25	Module Decomposition	Practical OO Design	Architecture
26	Integration/Adapter Diagrams	Practical OO Design	Architecture

Appendix B: Book Reference Details

Title	Author	Key Application
Thinking in Systems: A Primer	Donella H. Meadows	System dynamics, feedback loops
The E-Myth Revisited	Michael E. Gerber	Standardization, service mindset
Work the System	Sam Carpenter	Process documentation
Service Design Doing	Stickdorn et al.	Service optimization
Traction	Gino Wickman	Operating cadence, EOS
Scaling Up	Verne Harnish	Scalable execution
Modern Software Engineering	Dave Farley	DevSecOps integration
The Mythical Man-Month	Frederick P. Brooks Jr.	Coordination management
Practical Object-Oriented Design	Sandi Metz	Technical architecture

Next Steps

This reading sequence and diagram backlog can be converted into a **Views-and-Beyond style architecture documentation package** with:

- Epics organized by process area (intake, threat modeling, vulnerability management, exceptions, CI/CD gates, reporting)
- Diagram deliverables as user stories with acceptance criteria
- Stakeholder concerns mapped to each artifact
- Dependencies and execution order

See the companion document: *AppSec Architecture Documentation Package — Views-and-Beyond Style Diagram Backlog* for the complete ticketable backlog.