1

Why AI Matters for Domain-Driven Design Today

https://www.infoq.com/articles/architecture-trends-2024https://www.melconway.com/Home/Committees_Paper.htmlhttps://www.linkedin.com/posts/barry-o-reilly-b924657_model-drift-and-software-attractors-the-activity-7399023297276821504-h4Xm/https://teamtopologies.com/news-blogs-newsletters/2024/7/23/newsletter-july-mastering-team-effectiveness-the-power-of-managing-cognitive-loadhttps://github.com/ddd-crew/ddd-starter-modelling-process

Table 1.1 – The architect's week, before and with AI support

2

How LLMs Actually Think and Why Precision Unlocks Understanding

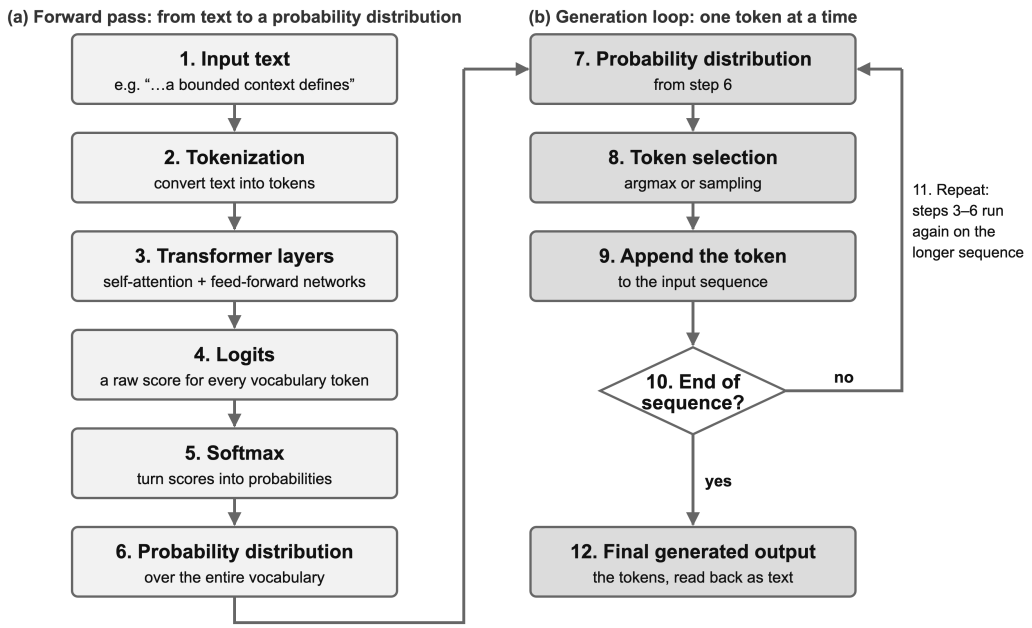


Figure 2.1 – Next-token prediction flow

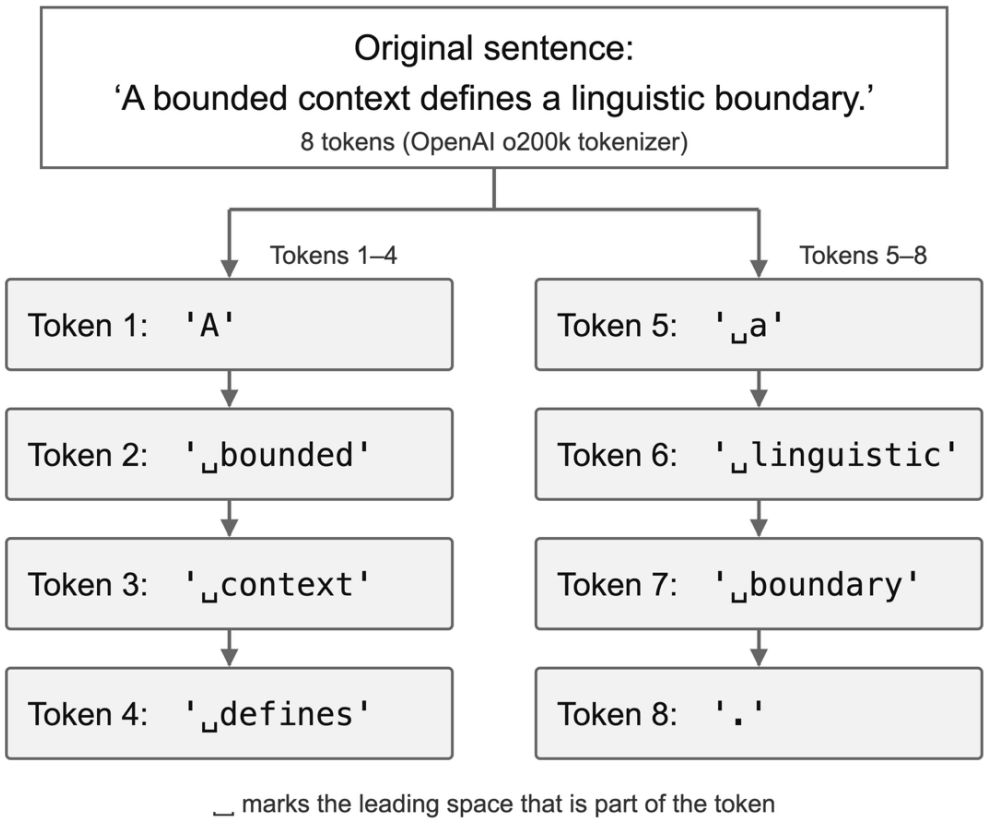


Figure 2.2 – Tokenization example

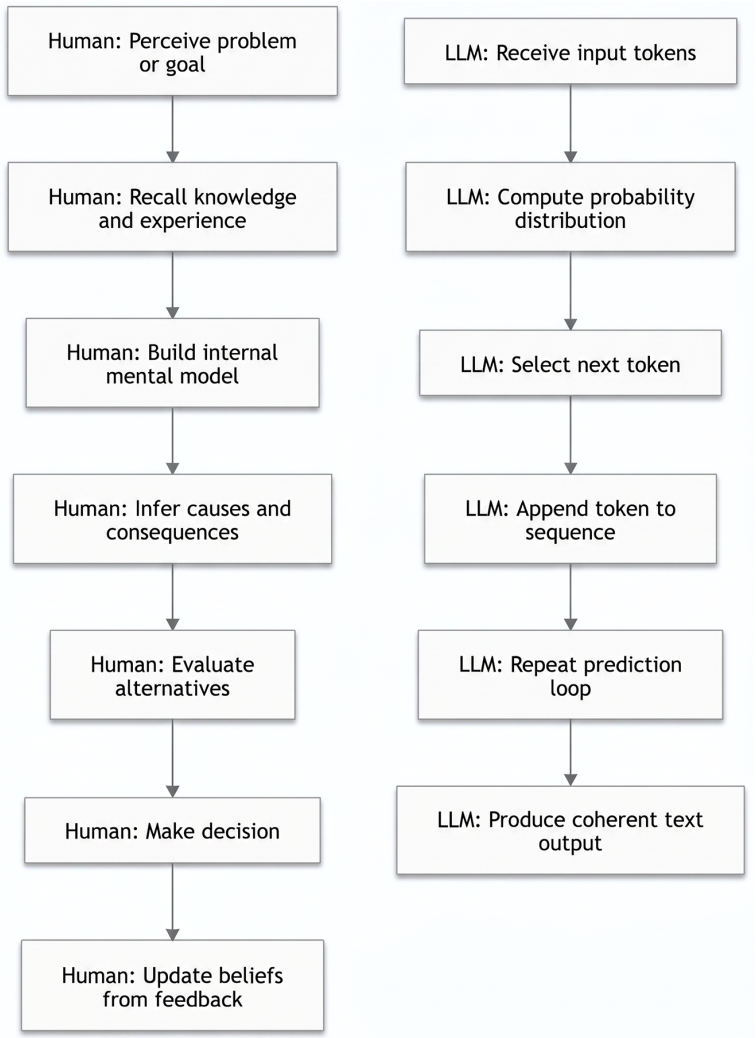


Figure 2.3 – Apparent reasoning versus statistical completion

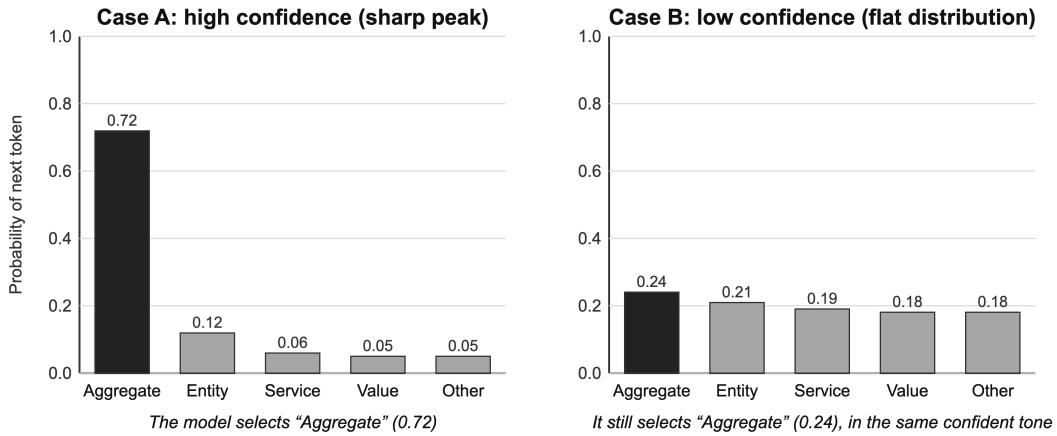


Figure 2.4 – Probability distribution under uncertainty

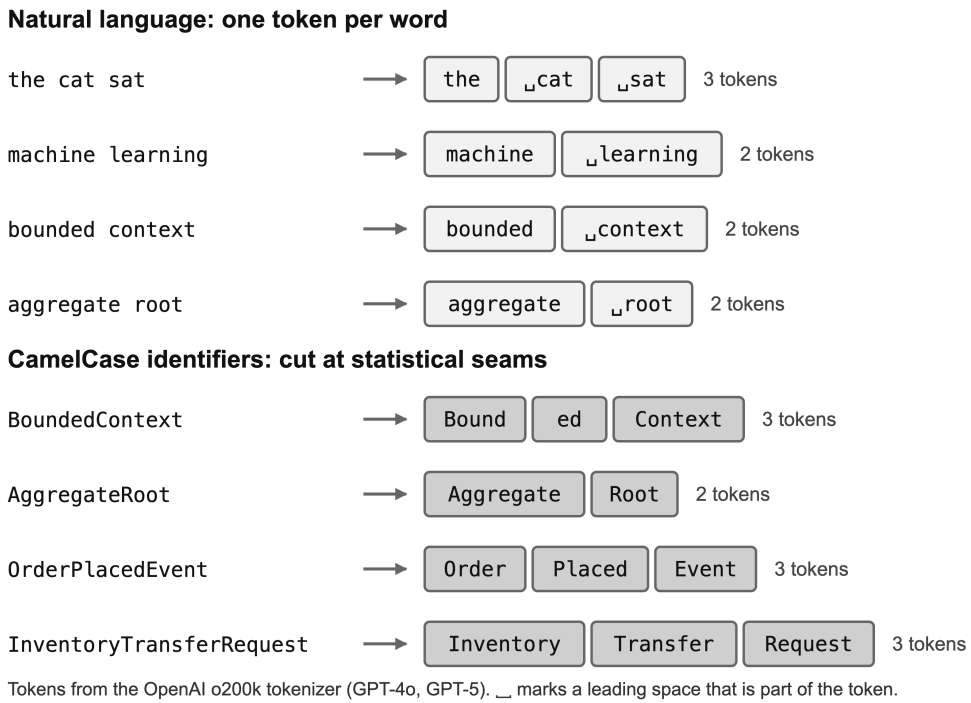


Figure 2.5 – BPE tokenization of DDD terms

`https://platform.openai.com/tokenizer`

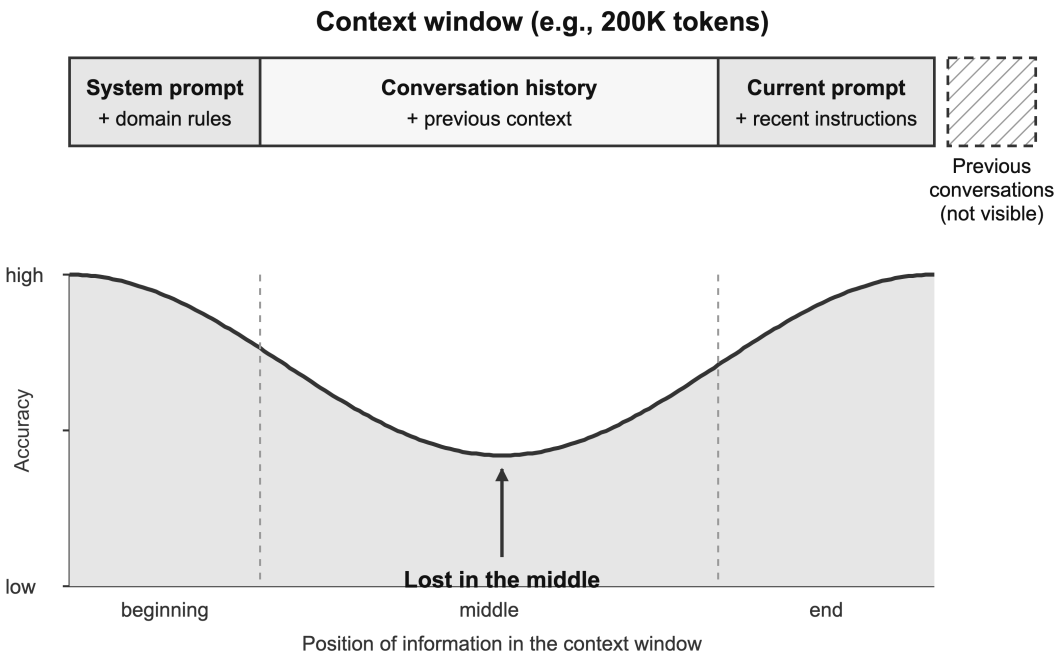
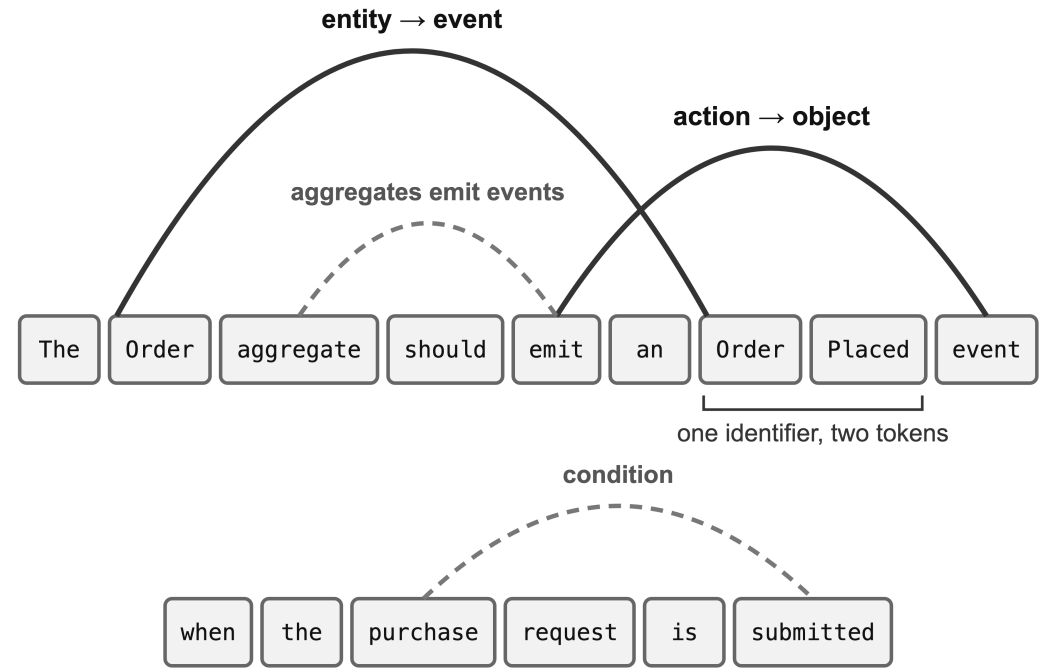


Figure 2.6 – The context window as working memory



Attention lets each token weigh itself and earlier tokens, however far apart they are in the sequence.

Figure 2.7 – Attention connecting distant tokens

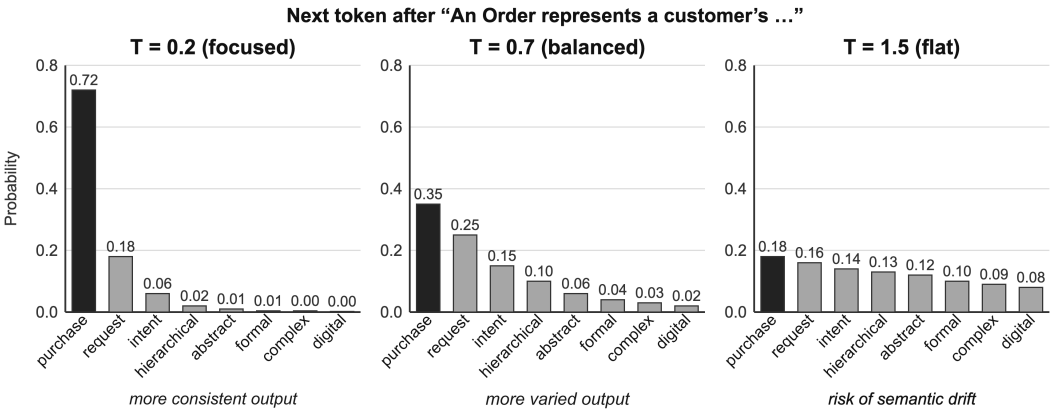


Figure 2.8 – Temperature and output variability (illustrative probabilities)

https://platform.claude.com/docs/en/models/overviewhttps://ai.google.dev/gemini-api/docs/long-contexthttps://arxiv.org/abs/1904.09751https://doi.org/10.1162/

tacl_a_00638https://github.com/openai/tiktokenhttps://developers.openai.com/api/docs/modelshttps://arxiv.org/abs/1706.03762

3

The Architect's View: Reasoning with AI Systems

Table 3.1 – The five components of an AI interaction system and the role each one plays

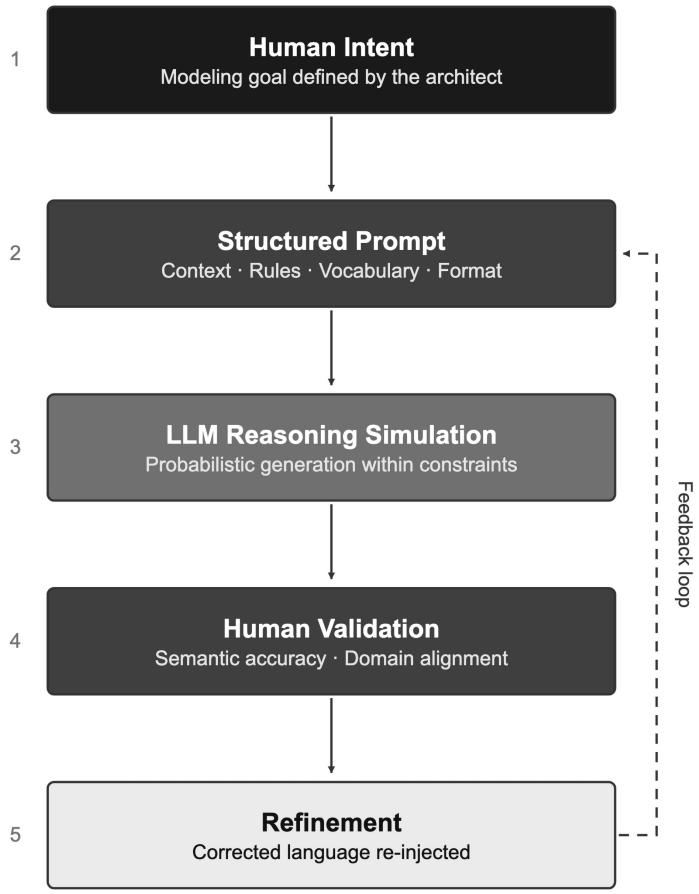


Figure 3.1 – AI reasoning loop

Table 3.2 – The DDD knowledge refinement cycle and its counterpart in the AI Reasoning Loop

Table 3.3 – The five information layers, their contents, and the architectural function of each

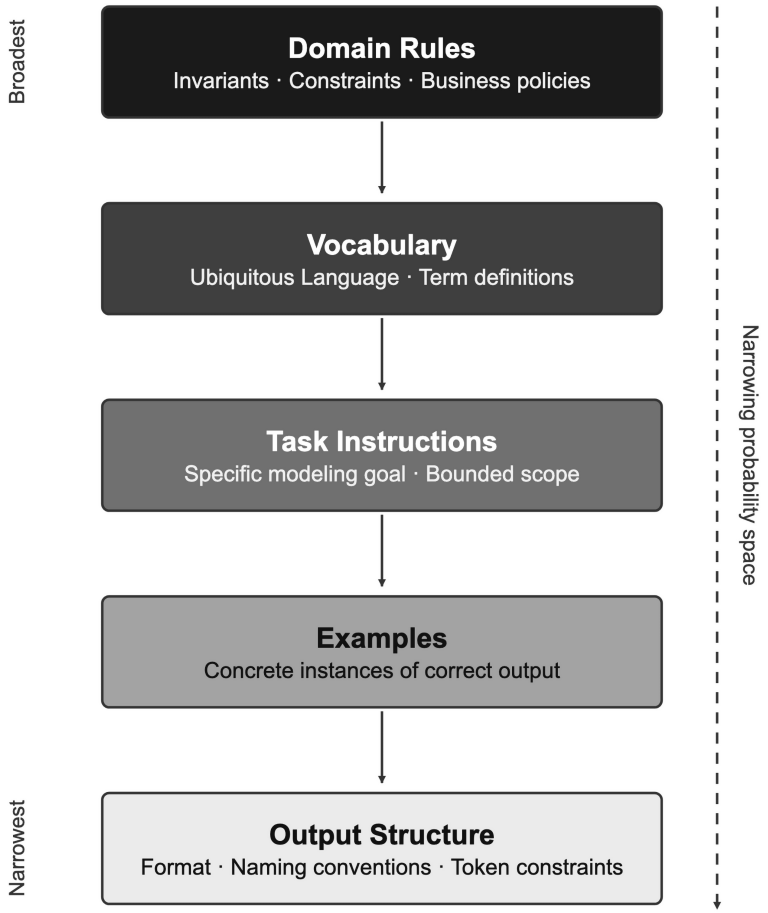


Figure 3.2 – Information layers in AI interaction

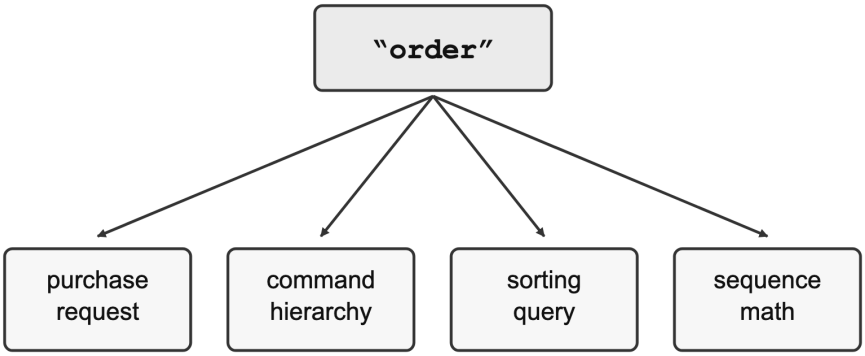


Figure 3.3 – The model statistically associates many meanings with the same word

Table 3.4 – A vocabulary section that fixes the meaning of three BrewUp terms for the model



Figure 3.4 – A disciplined vocabulary collapses multiple statistical interpretations into a single semantic anchor for the model

Table 3.5 – The four constraint types that bound the model's reasoning space

4

From Domain Exploration to AI Collaboration

https://github.com/PacktPublishing/AI-Augmented-Domain-Driven-Design/tree/main/docs/ch04

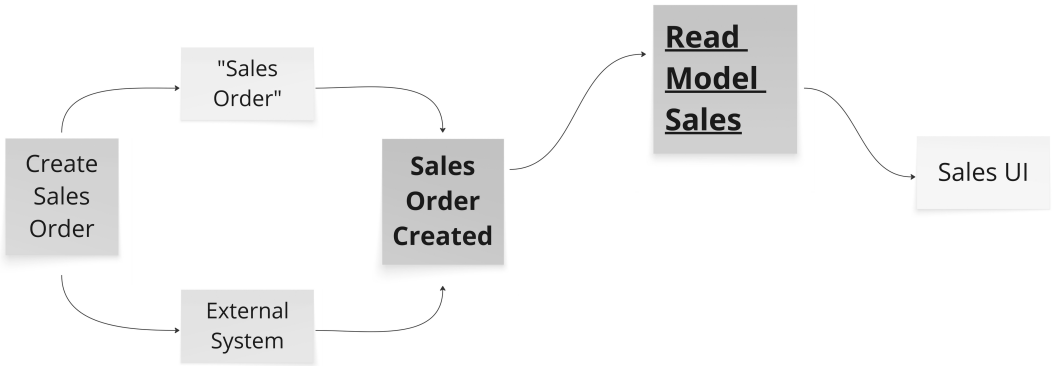


Figure 4.1 – An EventStorming example from the BrewUp ERP

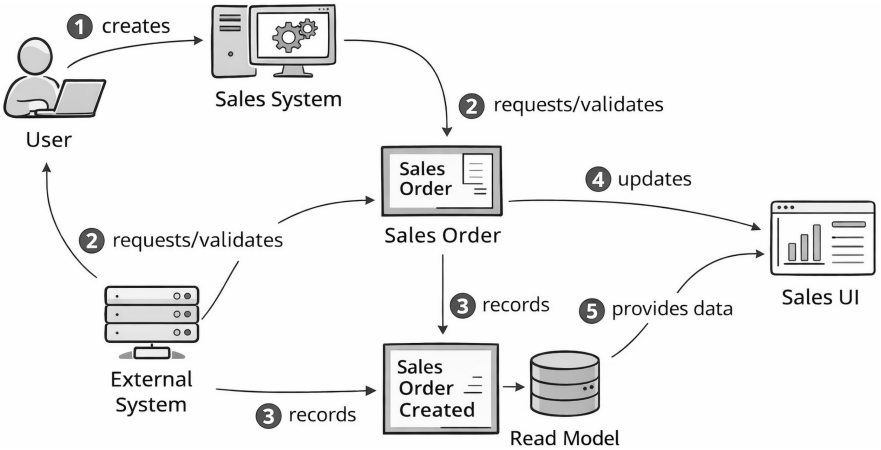


Figure 4.2 – The same process example using Domain Storytelling

`https://ghuntley.com/ralph/`

5

Defining Rules: Teaching the Model to Think in DDD

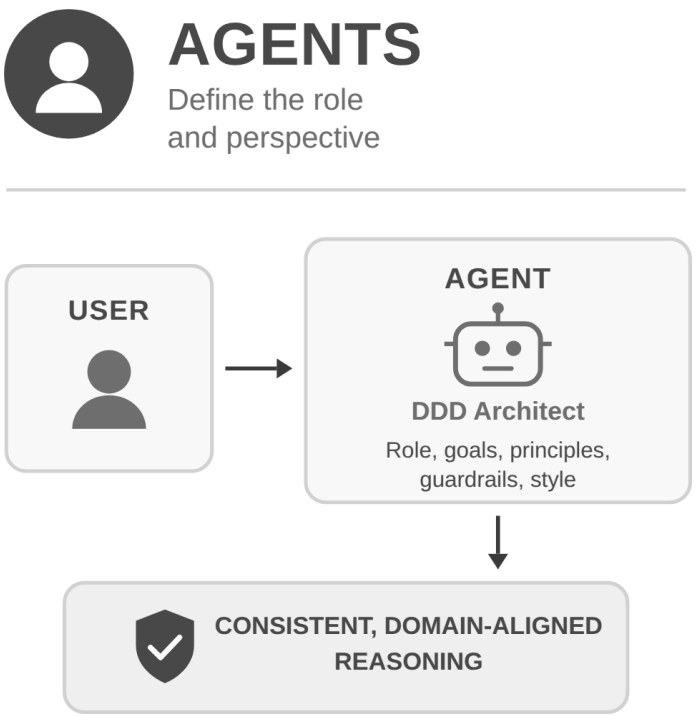


Figure 5.1 – An agent represents a DDD architect

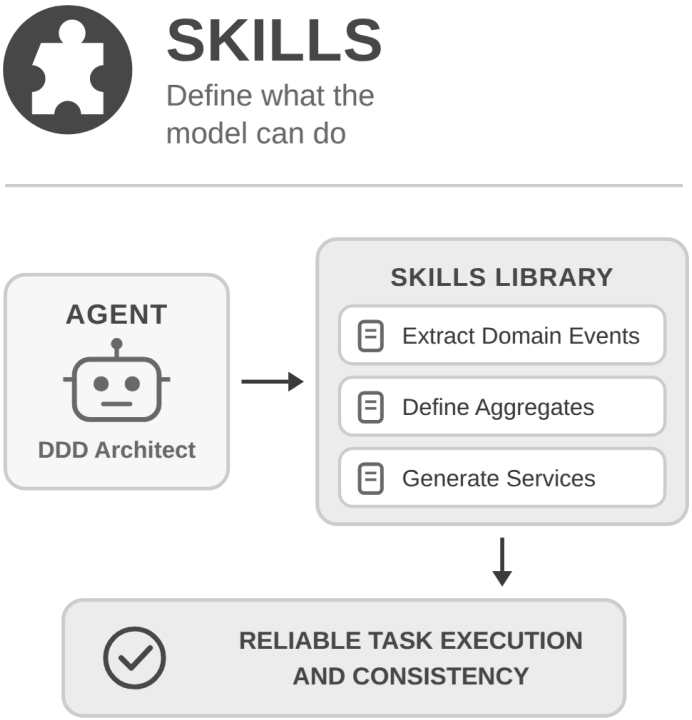


Figure 5.2 – A skill as a container for specific, reusable capabilities

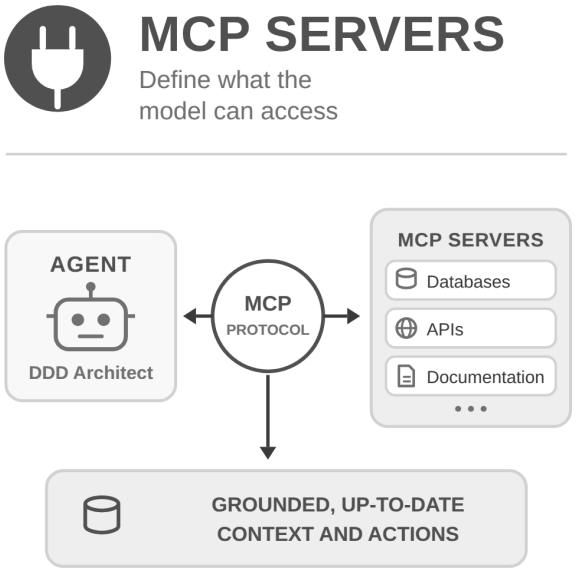


Figure 5.3 – MCP allows communication with external systems

```
https://modelcontextprotocol.io/
```

6

Iterative Refinement: Building the Bounded Context

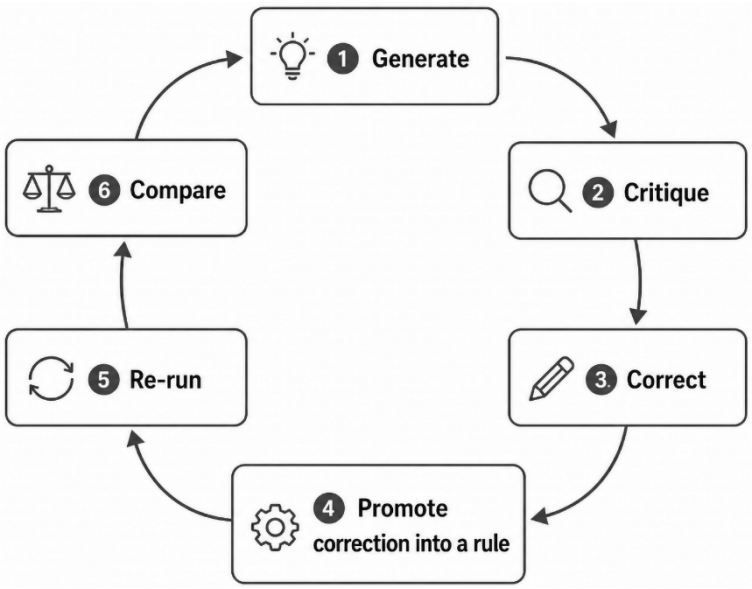


Figure 6.1 – Refinement loop

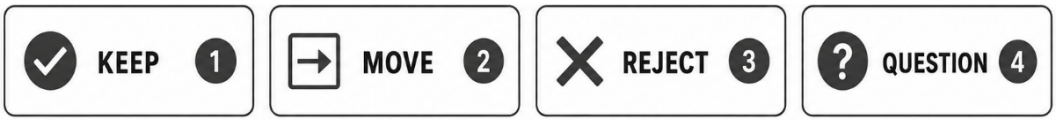


Figure 6.2 – The classification filter used during critique



https://packt.link/deep-engineering-newsletter

https://packt.link/deep-engineering-newsletter

7

Prompt as Domain Carrier

Table 7.1 – Roles of the domain carrier in different workflows

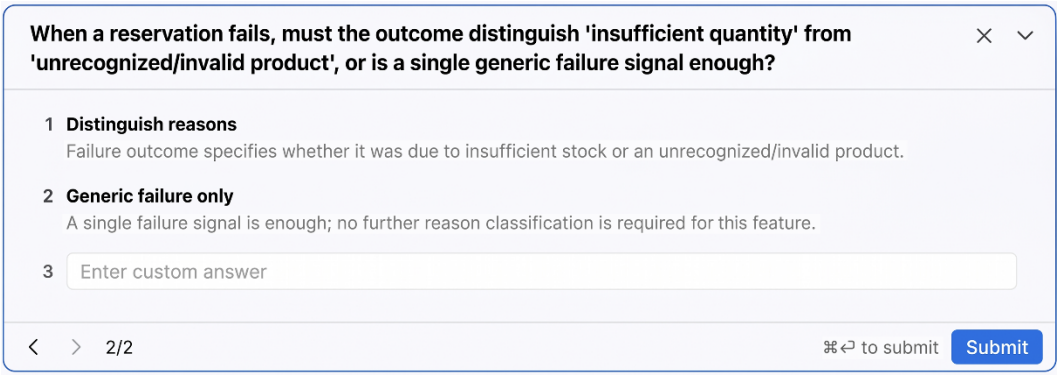


Figure 7.1 – An open question preserved by the domain carrier is surfaced to the team instead of being answered by the model

`https://github.com/bmad-code-org/BMAD-METHODhttps://github.com/github/spec-kithttps://github.com/Fission-AI/OpenSpec`

8

From Prompts to Agents: Structuring AI Collaboration

```
https://github.com/PacktPublishing/AI-Augmented-Domain-Driven-Design
```

Table 8.1 – Where each run is preserved in the companion repository

Table 8.2 – How the specification transforms the original request into explicit DDD constraints

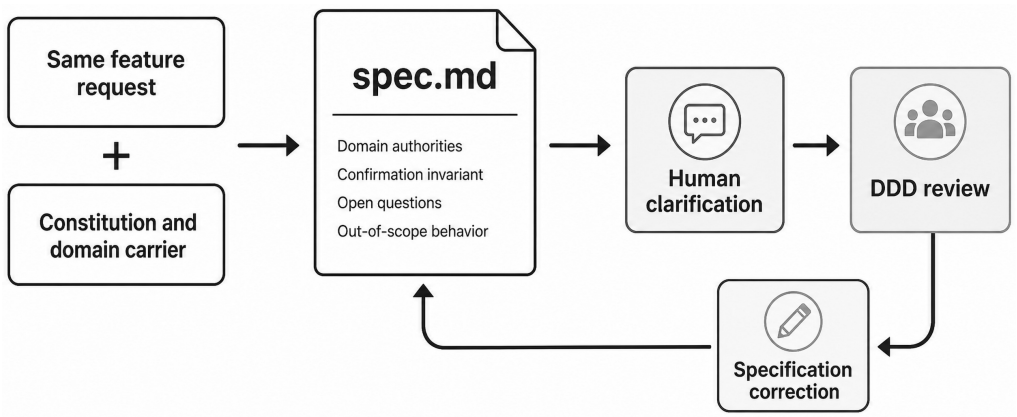


Figure 8.1 – From the same feature request to a reviewable and correctable specification

Table 8.3 – Traceability and DDD review as the specification becomes executable work

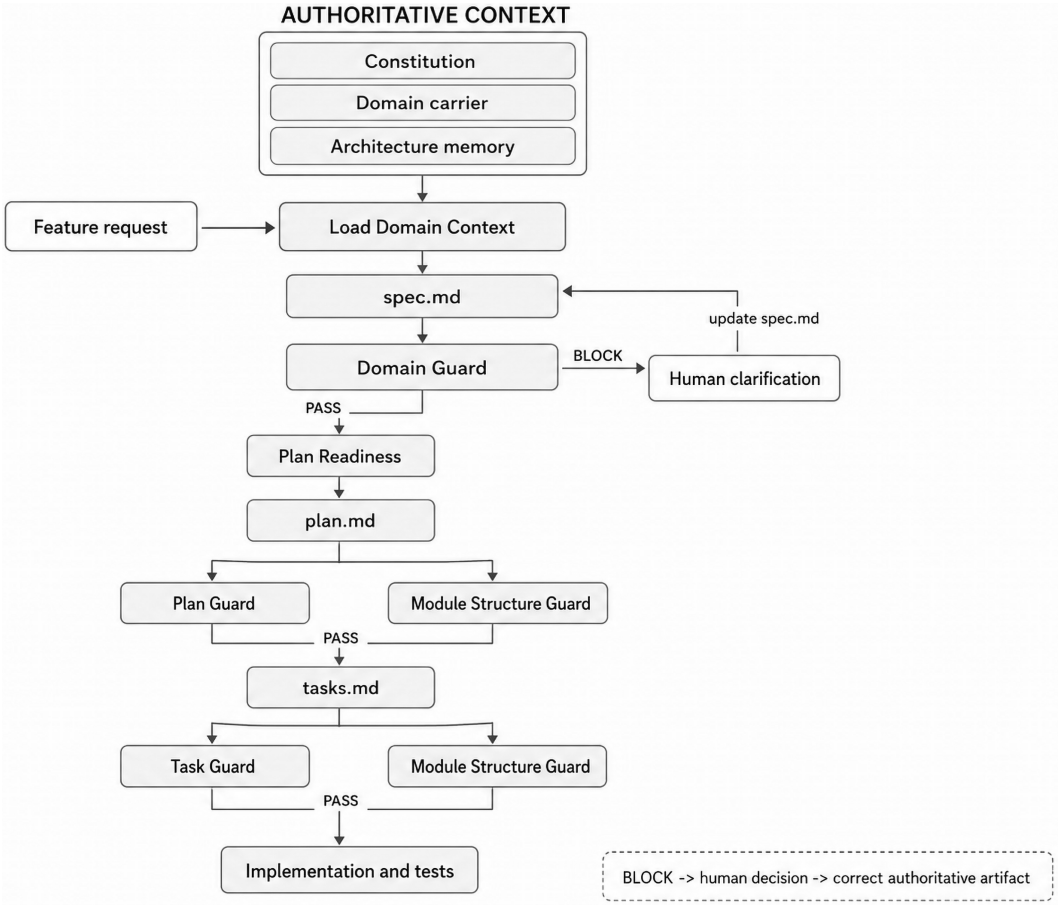


Figure 8.2 – Guard agents create cognitive boundaries between specification-driven artifacts

Table 8.4 – The harness agents and the transitions they inspect

Table 8.5 – The four runs, what each one added, and what each one left unproved

`https://github.com/github/spec-kithttps://github.com/bmad-code-org/BMAD-METHOD`

9

From Guardrails to Specialists: Designing Agents for DDD

Table 9.1 – Four bounded transformations: what each specialist may propose and what it may not decide

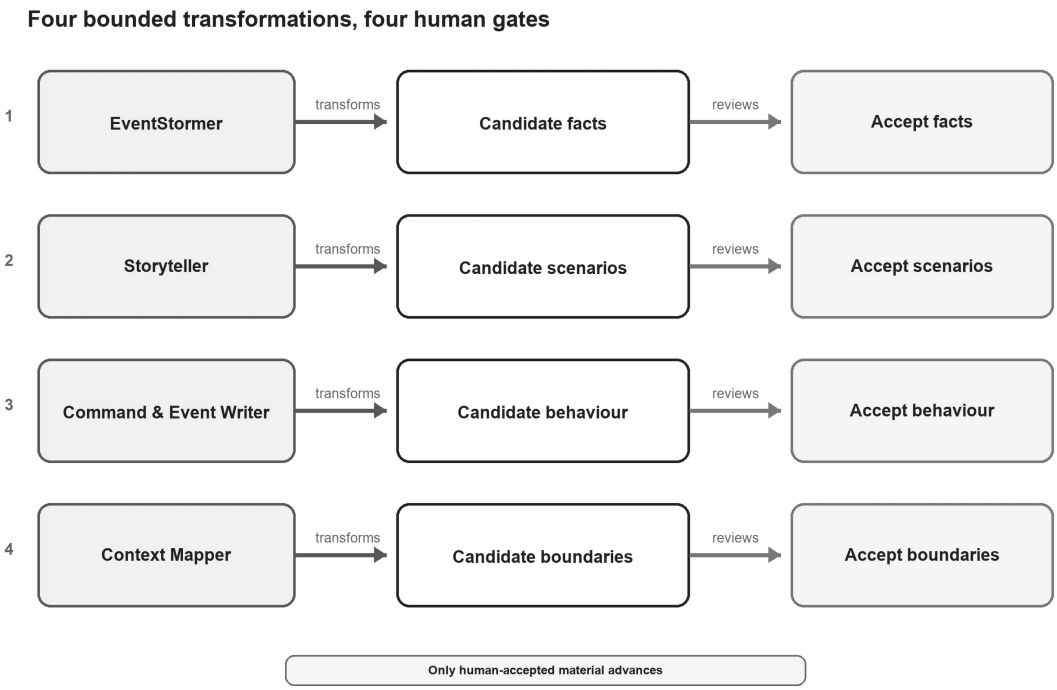


Figure 9.1 – Four bounded transformations separated by human gates

Table 9.2 – What a specialist proposes, and the DDD building block a reviewer may recognize in it

`https://github.com/PacktPublishing/AI-Augmented-Domain-Driven-Design`

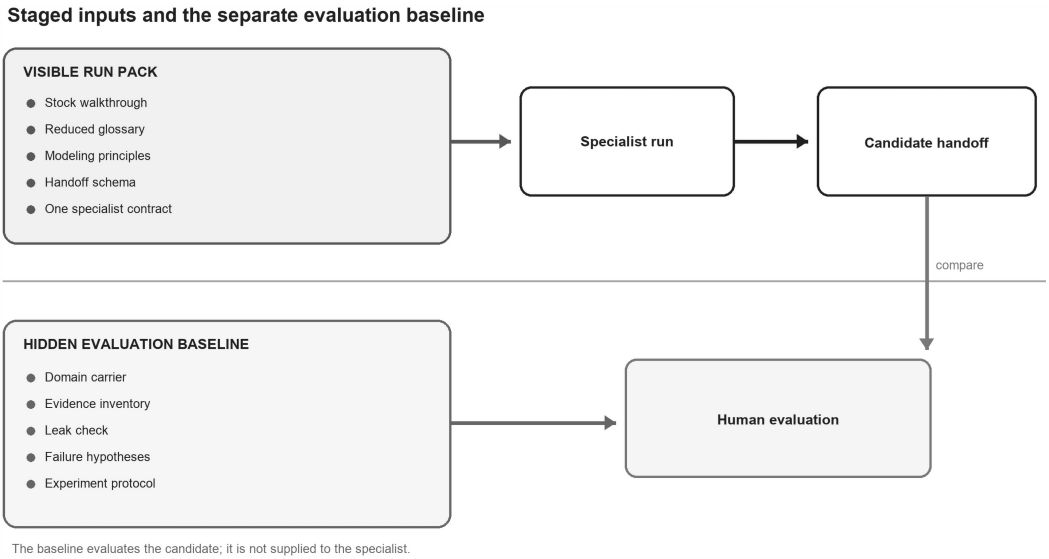


Figure 9.2 – Visible run pack and hidden evaluation baseline

Table 9.3 – The chapter's working vocabulary and where each artifact lives in the companion repository

Table 9.4 – What the four specialist runs produced and what the gates changed

10

The Orchestrator: Building the Coordinating Agent

https://github.com/PacktPublishing/AI-Augmented-Domain-Driven-Design/tree/ch10/orchestrator

Table 10.1 – Decision rights across the orchestration boundary

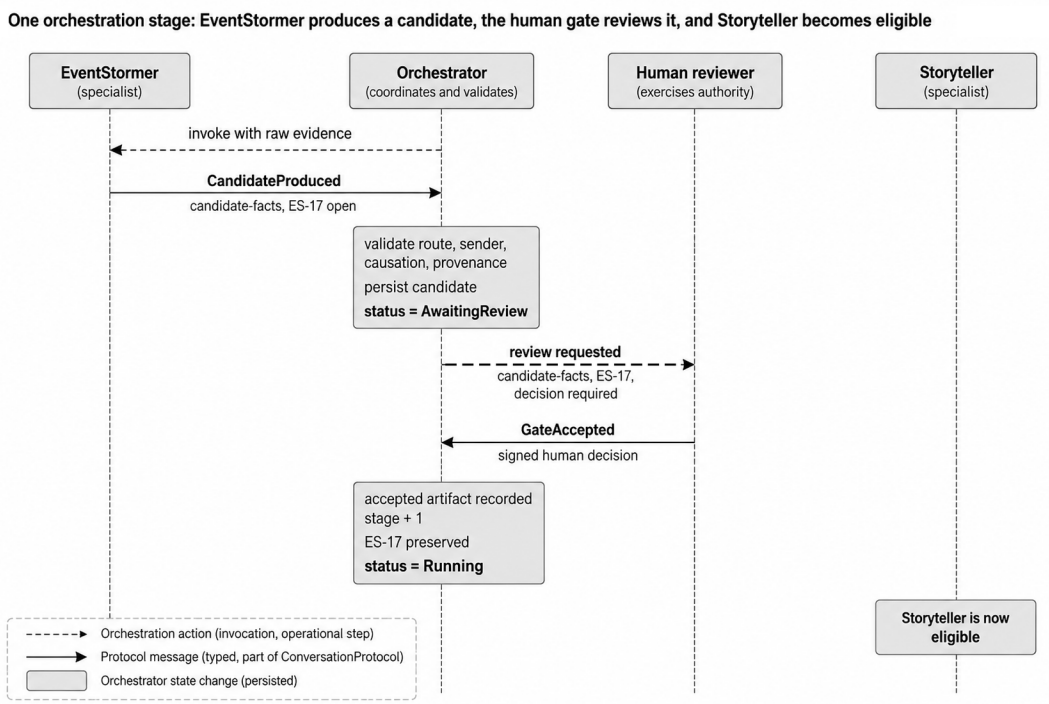


Figure 10.1 – One orchestration stage: candidate, human gate, and the next specialist made eligible

Table 10.2 – Legal messages in the conversation protocol

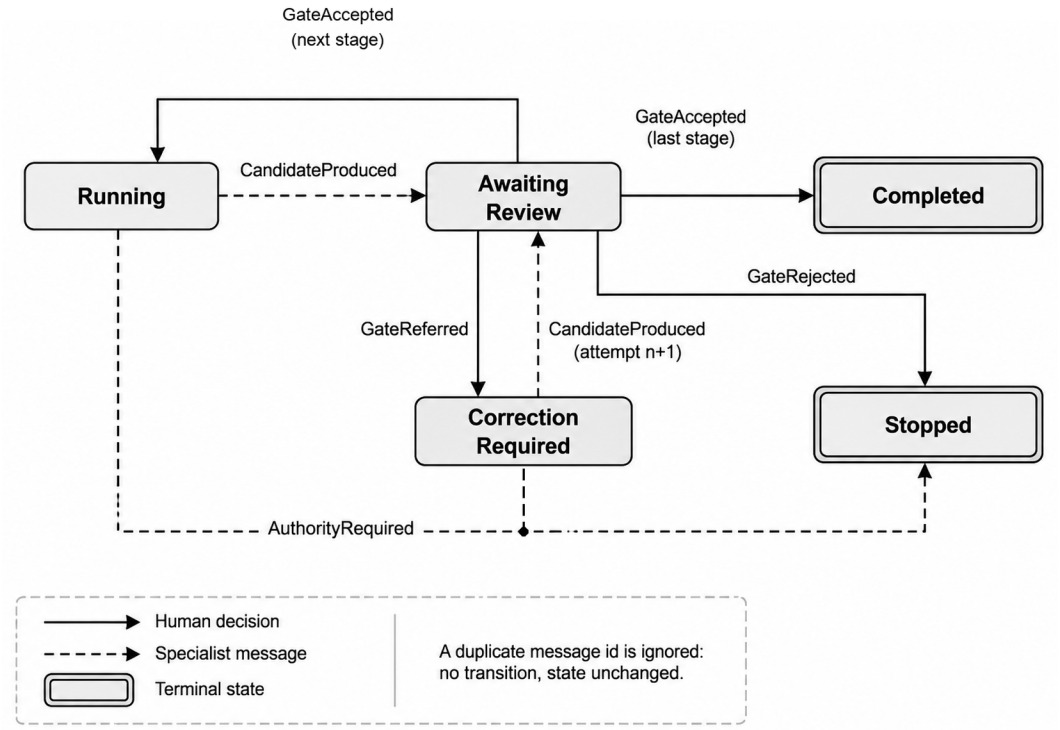


Figure 10.2 – Legal transitions of ProtocolStatus

Table 10.3 – A blocked message and a stopped workflow

11

Governed Autonomy: Balancing Autonomy and Alignment Across Agents

`https://github.com/PacktPublishing/AI-Augmented-Domain-Driven-Design/tree/ch11/governance`

Table 11.1 – The five mechanical alignment checks and the boundary each one protects

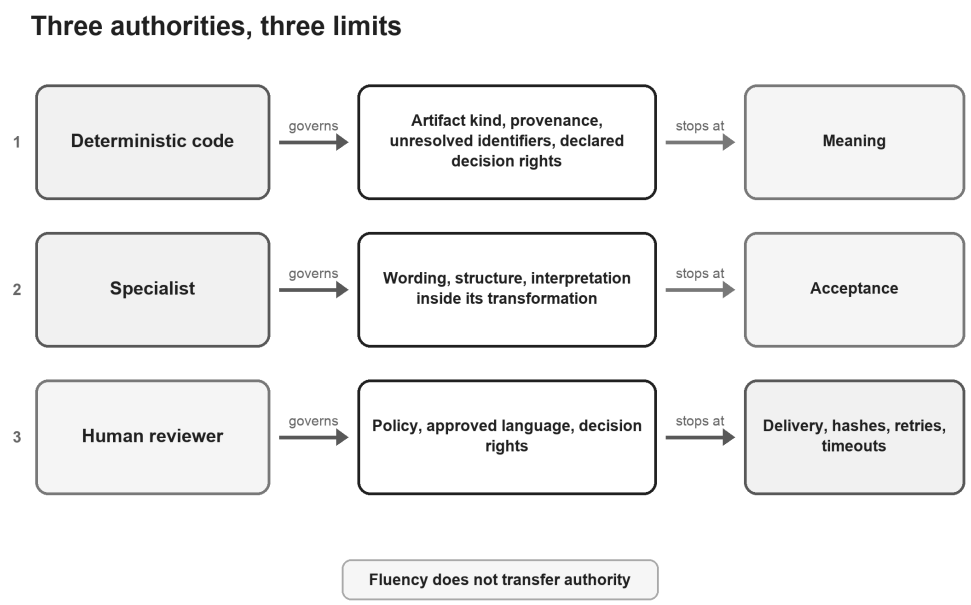


Figure 11.1 – Three authorities and the limit of each one

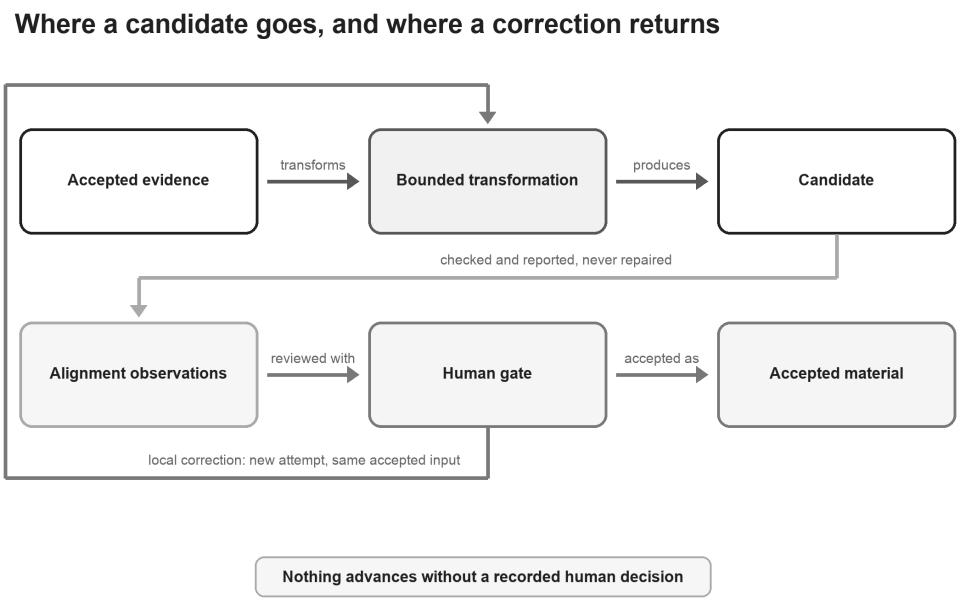
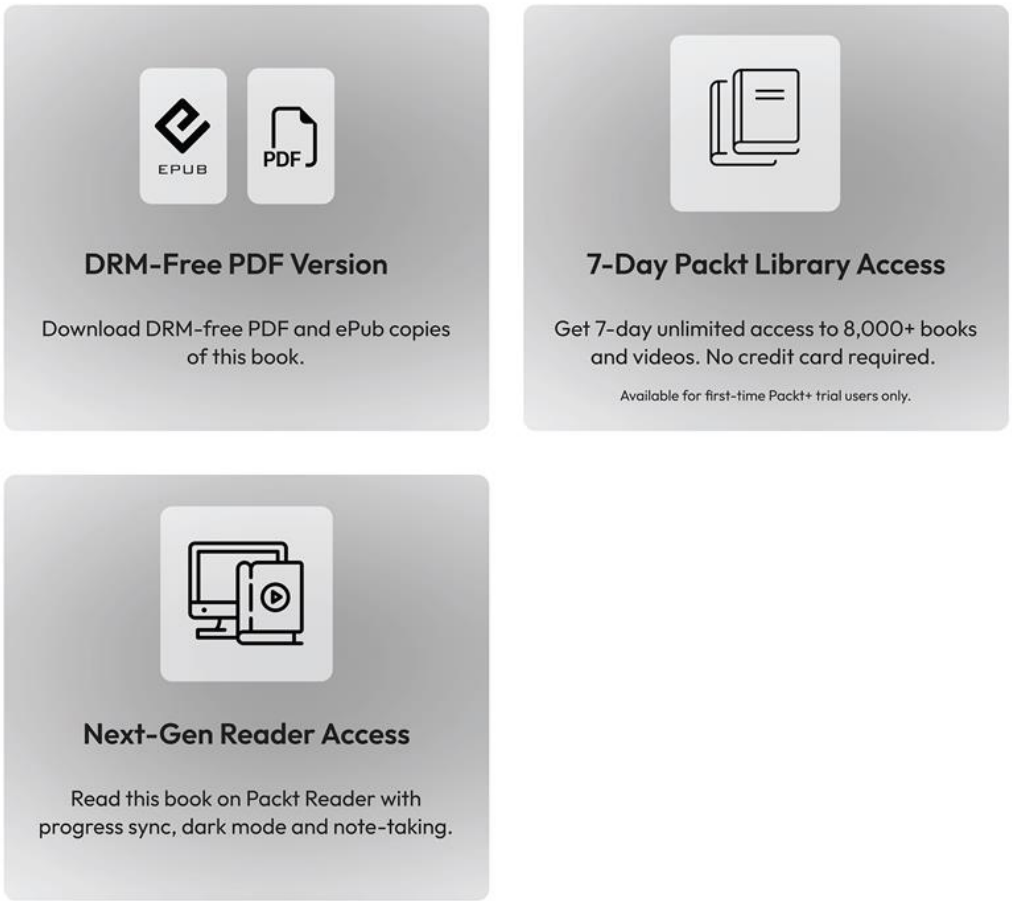


Figure 11.2 – Where a candidate goes, and where a correction returns

12

Unlock Your Exclusive Benefits



https://www.packtpub.com/en-us/unlock?step=1.packtpub.com/unlock

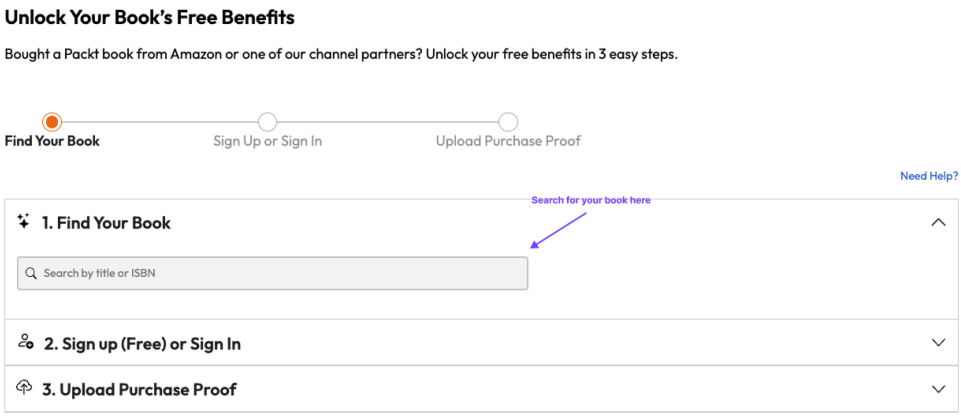


Figure 12.1: Packt unlock landing page on desktop

`https://www.packtpub.com/unlock-benefits/helpcustomercare@packt.com`