

The Domain-Driven Architecture Mindset



Understand the business process.
Automate where it matters.
Anticipate when the architecture must change.

1. Model the stable process

Start with the business as it works manually—not with the software.

- What happens?
- Which decisions are made?
- Which domain events mark meaningful outcomes?
- Which exceptions and variations occur?
- Which business rules must always hold?

Output: A technology-independent domain model.

The business process is relatively stable. Its implementation is replaceable.

2. Find high automation value

Do not automate everything at once. Start where automation creates the greatest benefit for the least effort.

Evaluate each case by:

- frequency
- effort saved
- delay reduced
- errors prevented
- business value created
- implementation complexity

Automate the common, simple, valuable cases first. Keep rare or ambiguous cases manual until automation becomes worthwhile.

Optimize for business impact. Do not aim for automation coverage.

3. Find the thresholds

Every implementation works only within a certain operating range.

Identify the conditions under which the current solution stops being good enough:

- transaction volume
- response time
- number of users
- exception rate
- manual workload
- coordination effort
- cost per case
- regulatory or risk exposure

Define measurable thresholds:

This implementation works until **X** reaches **Y**.

A threshold is not merely a technical limit. It is the point at which the implementation begins to constrain the business.

4. Observe the constraint

Measure the part of the system currently limiting throughput, quality, or growth.

Ask:

- Where does work accumulate?
- Where do people wait?
- Where do failures and rework occur?
- Which resource determines overall throughput?
- How quickly are we approaching its threshold?

Do not optimize components that are not constraining the business.

The constraint determines where architectural change creates value.

5. Predict the next threshold

Use business forecasts and operational measurements to estimate:

- when the current constraint will reach its limit;
- which constraint will emerge after it is removed;
- how long the required architectural change will take.

The relevant moment is not when the threshold is crossed:

Start changing =
Predicted threshold date
– Lead time
– Safety margin

Architecture must respond to tomorrow's constraint, not yesterday's symptoms.

6. Change the implementation

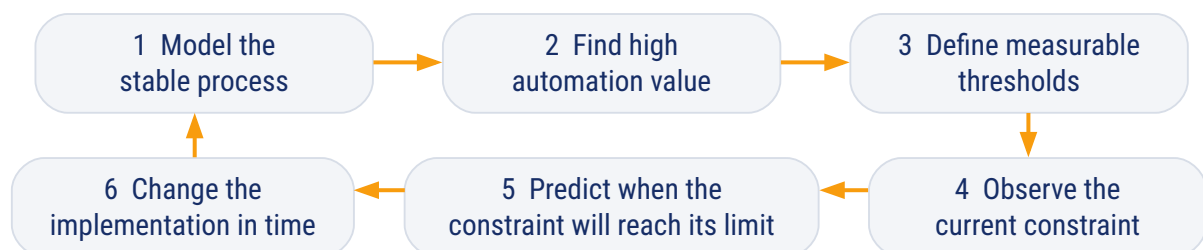
Change only what is necessary to move the threshold beyond the expected business demand.

Preserve the stable domain model whenever possible:

- keep domain language and events stable;
- replace algorithms, components, or infrastructure;
- introduce specialization only where justified;
- prefer small, reversible changes;
- measure whether the constraint actually moved.

Stable domain semantics enable replaceable implementations.

Repeat the loop



The core principle

Model what remains stable.
Measure what will become limiting.
Change the implementation before it constrains the business.