

CompanyGraph — Domain- driven design, as pages you can check

A bounded context, its language, its aggregates and events, as pages a check holds to their boundary.

RESTS ON [concept Pack](#)

Robert Blust · Software Engineer & Architect

The method you know

The pack writes the words of domain-driven design down as pages, rather than inventing its own.

EVANS

[Domain-Driven Design Reference](#)

VERNON

[Domain-Driven Design Distilled](#)

DDD CREW

[Bounded Context Canvas](#)

DDD CREW

[Aggregate Design Canvas](#)

DDD CREW

[Context Mapping](#)

JACKSON

[The Essence of Software](#)

CUCUMBER

[Gherkin reference](#)

RESTS ON

concept Pack

Bounded context

The boundary within which one model and one language hold, as a page: what it is responsible for, how it is classified and the domain it realizes.

RESTS ON `bounded-context Resolution`

model/bounded-contexts/resolution/resolution.md

```
---
id: 01a0faaa-5d02-71db-bcdb-f672c4324bee
source: Local
classification: core
realizes:
  - Core
decisions:
  - A reference resolves by its declared type, never by name alone
---

# Resolution
> Turns the names a page writes into the edges of the graph, by the type each schema
  declares and the owner a row names. Whether a page has the shape its schema asks for is
  left to the checks.

## Responsibilities
- Find the entity a written name means, looking only among the type the schema declares for
  that field or column
- Look for an owned entity only inside the owner its row names, so two owners can each have
  a page of the same name
...
```

Its language

Concept designs: the terms of the context's language, each an entity or a value object, owned by its context.

RESTS ON

concept-design Entity

concept-design Edge

concept-design Scope

concept-design Declaration

concept-design Graph

model/bounded-contexts/resolution/concept-designs/

declaration.md	kind: value object
edge.md	kind: value object
entity.md	kind: value object
graph.md	kind: entity
scope.md	kind: value object

Aggregate and event

An aggregate keeps a cluster of concept designs consistent as one unit. A domain event is a type, because other contexts name it; a command is a row, because nothing outside its aggregate does.

RESTS ON aggregate Graph domain-event Graph drawn

```
model/bounded-contexts/resolution/aggregates/graph.md

---
...
root: Graph
members:
  - Edge
  - Scope
...
# Graph
...
### Invariants
| Label | Invariant |
| --- | --- |
| INV-R1 | Every edge runs between two entities of the graph; a name that resolves to
nothing stops the read, and no edge points nowhere. |
...
### Handled commands
| Command | Emits | When | Description |
| --- | --- | --- | --- |
| Read an instance | | | Reads every page against the schemas the instance vendors and
draws the graph, or refuses at the first name that resolves to nothing |
...
```

Context map

A relationship between contexts is a row on the context that depends; a symmetric one is written once, on either side.

RESTS ON

bounded-context [Checking](#)

model/bounded-contexts/checking/checking.md

```
...  
### Relationships  
| Context | Pattern |  
| --- | --- |  
| Resolution | conformist |  
| Vendoring | shared kernel |  
...
```

Feature design

How a feature is built across the contexts it touches: an operational principle, then scenarios in Gherkin.

RESTS ON

```
feature-design A name that names nothing is reported where it is
written
```

model/feature-designs/a-name-that-names-nothing-is-reported-where-it-is-written.md

```
---
id: 01a10031-6925-7b3c-8f04-87b19b4f6103
source: Local
refines: Checks an instance runs
contexts:
  - Resolution
...
# A name that names nothing is reported where it is written
...
### Operational principle
When a page names an entity that does not exist under the type its field declares, the
  check fails on that page and says which type it searched, before the change can land.

### Scenarios
#### SC-C1: A name that exists nowhere
...
```

Where it departs

Where the pack does not do what its sources do, its README says so.

RESTS ON

`decision` A reference resolves by its declared type, never by name alone

Strategic classification

on the bounded context, as the Bounded Context Canvas puts it, not on Evans's subdomain, so core's `domain` stays untouched

A domain event and a command

an event a type, because other contexts name it; a command a row, because nothing outside its aggregate does

An architecture decision

core's `decision`, which already has an ADR's sections; the pack's types name it

A value's type

Term for a term of the context, a reference; Type for a plain type, which names nothing

How it joins core

The pack joins core's types: a bounded context realizes a domain, a concept design refines a concept and a feature design refines a feature. Every edge runs one way and is optional.

RESTS ON

domain Core

feature Checks an instance runs

LEVEL 1 · SOFTWARE

bounded context · concept design · feature design

↓ realizes · refines · refines ↓

LEVEL 0 **Core**

domain · concept · feature

What the check holds

A name is looked for only within its type and, for an owned type, its owner. One that names nothing fails where it is written.

RESTS ON `concept-design Entity · Publishing`

`concept-design Entity · Resolution`

`concept-design Entity · Serving`

`concept Check`

```
$ companygraph check
```

```
✗ 1 problem in model/ against meta/core, meta/software, meta/organization, meta/landscape/  
at core 0.65.0
```

```
model/bounded-contexts/resolution/concept-designs/edge.md: `Concept` in "### Relations" is  
declared `ref → concept-design` and says "Declarations", which names no entity in model/;  
a declared reference must resolve, and the concept-design it matches here is  
"Declaration" (R4)
```

```
not checked: every `### Writing rules` in every schema – that is the agent pass, R0
```

```
✓ 378 Markdown files in the one form
```

Left for later

What the pack does not have yet is named in its README, not hidden.

LATER	services, repositories, factories and modules
LATER	C4's system, container and component
LATER	a type for a relationship between contexts
LATER	commands as a type
LATER	a subdomain type
LATER	a level field
LATER	read models and policies as types, from Event Modeling

RESTS ON `concept Pack`

Drawn by the server

Nobody drew these. The MCP server's own code draws them from the pages, at the commit the site pins.

RESTS ON

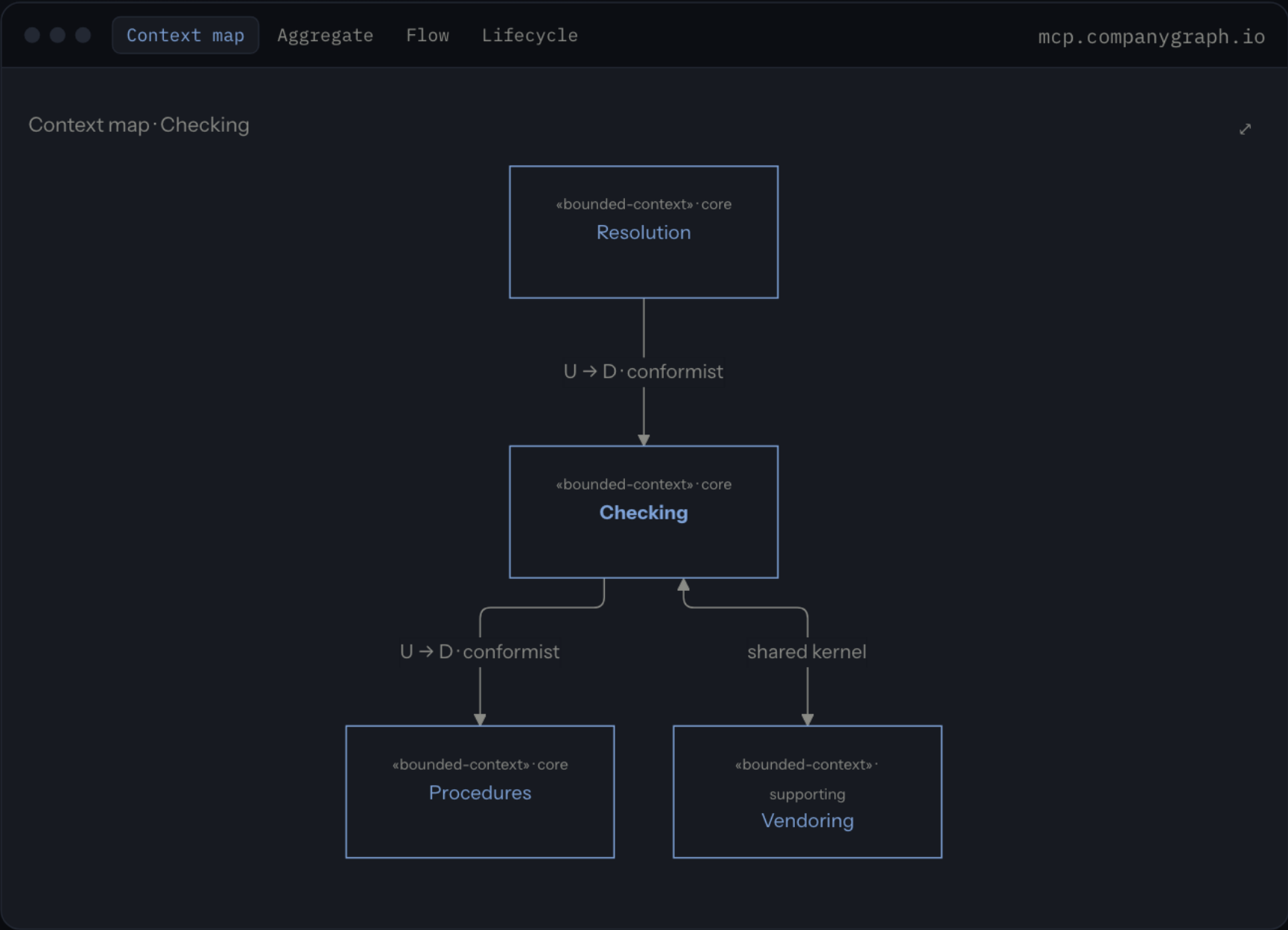
bounded-context Checking

bounded-context Resolution

bounded-context Procedures

bounded-context Vendoring

aggregate Check run



Your language, held to its boundary

Domain-driven design's own words, written as pages a check holds. Where the pack departs from its sources, it says why.

RESTS ON concept Pack

LEVEL 1 · SOFTWARE **bounded context · concept design · feature design**

↓ realizes · refines · refines ↓

LEVEL 0 **Core**
domain · concept · feature