

# What someone wants, and what a product is held to

A use case with its stakeholders, scenario and extensions, and a requirement with its kind, modality and fit criterion, written as pages a check holds and drawn by the MCP server.

RESTS ON [concept Pack](#)

Robert Blust · Software Engineer & Architect

# The method you know

The pack writes the words of use cases and requirements down as pages, and its README names the sources they come from.

|                              |                                                            |
|------------------------------|------------------------------------------------------------|
| COCKBURN AND JACOBSON        | <a href="#">Use-Case Foundation</a>                        |
| COCKBURN                     | <a href="#">Basic use case template</a>                    |
| COCKBURN                     | <a href="#">Structuring use cases with goals</a>           |
| OMG                          | <a href="#">UML 2.5.1, clause 18</a>                       |
| JACOBSON, SPENCE AND BITTNER | <a href="#">Use-Case 2.0</a>                               |
| ISO/IEC/IEEE                 | <a href="#">29148:2018</a>                                 |
| IREB                         | <a href="#">CPRE glossary</a>                              |
| VOLERE                       | <a href="#">Requirement snow card</a>                      |
| IETF                         | <a href="#">RFC 2119, key words for requirement levels</a> |
| GITHUB                       | <a href="#">spec-kit, specification template</a>           |

RESTS ON concept Pack

# A use case

One goal for one actor, and every way of reaching it. Its level, its scope and its primary actor stand in the front matter, with the features it asks something of.

RESTS ON `use-case Ask the company a question`

`feature A visitor asks the model` `seat Visitor`

model/use-cases/ask-the-company-a-question.md

```
---
id: 01a1312c-4d5f-7923-b896-700a5ce8d15f
source: Local
level: user goal
scope: system
primary-actor: Visitor
features:
  - A visitor asks the model
  - A chat answer marks the claims its evidence does not carry
---
# Ask the company a question
> A visitor asks about the company in their own words and language, and gets an answer that
names the entity each claim rests on, says where the model does not say, and sends them
to the page the claim came from at the commit it was read.
...
```

# Stakeholders and guarantees

Who has a stake in the goal and what each wants. What holds whatever happens, and what holds when the goal is reached.

RESTS ON `use-case Ask the company a question`

model/use-cases/ask-the-company-a-question.md

...

## ### Stakeholders and interests

| Stakeholder | Interest |
|-------------|----------|
|-------------|----------|

|     |     |
|-----|-----|
| --- | --- |
|-----|-----|

|         |                                                                                                  |
|---------|--------------------------------------------------------------------------------------------------|
| Visitor | What the model says and nothing else, in their language, with no account and nothing left behind |
|---------|--------------------------------------------------------------------------------------------------|

|          |                                                                                 |
|----------|---------------------------------------------------------------------------------|
| Answerer | A host whose tools answer at one pinned commit, and a fence it can write inside |
|----------|---------------------------------------------------------------------------------|

|       |                                                                                                       |
|-------|-------------------------------------------------------------------------------------------------------|
| Owner | No claim standing that no tool returned, and nothing spent past the ceiling the deployment wrote down |
|-------|-------------------------------------------------------------------------------------------------------|

...

## ### Minimal guarantees

- No claim stands that no tool returned in that conversation.

...

- Nobody reads the answer before the visitor does.

## ### Success guarantees

- Every claim names the entity a tool returned it from, and the visitor can open that entity at the commit the answer names.

...

# The scenario

A trigger, then the steps on which everything goes right. A step names who acts, and a step that calls another use case names it, and the page lists it in `includes`.

RESTS ON `use-case Ask the company a question`

`use-case Read the model from an agent`

`use-case Walk a long list a page at a time`

model/use-cases/ask-the-company-a-question.md

...

## ### Trigger

The visitor opens the chat and presses send.

## ### Main success scenario

| Step | Actor | Action |
|------|-------|--------|
|------|-------|--------|

| --- | --- | --- |
|-----|-----|-----|
|-----|-----|-----|

|   |         |                                       |
|---|---------|---------------------------------------|
| 1 | Visitor | Asks, in their own words and language |
|---|---------|---------------------------------------|

|   |        |                                                                                                                                                      |
|---|--------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2 | System | Lets the message through the fence: under the length it allows, from a page the deployment names, inside the hour's, the day's and the month's share |
|---|--------|------------------------------------------------------------------------------------------------------------------------------------------------------|

|   |          |                                                                                                                           |
|---|----------|---------------------------------------------------------------------------------------------------------------------------|
| 3 | Answerer | Asks the host's tools, up to four rounds: a kind of thing by its type, a named thing by search and then the entity itself |
|---|----------|---------------------------------------------------------------------------------------------------------------------------|

|   |          |                                                                                                                                                                                          |
|---|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4 | Answerer | Writes the answer from what the tools said, in the language of the message's words, naming the entity each claim rests on, and where the tools said nothing, that the model does not say |
|---|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

...

# Extensions

Where the scenario branches: a condition at a step, and what happens then. A step may branch more than once.

RESTS ON `use-case Ask the company a question`

```
model/use-cases/ask-the-company-a-question.md

...
### Extensions
| Step | Condition | Handling |
| --- | --- | --- |
| 1a | The question is not about the model | It is answered with one sentence saying what the chat is for, and no tool is asked |
| 2a | The day's share is spent | The message is refused before the model is asked, the refusal names the next midnight UTC as the moment it lifts, and the question is kept with the refusal's code |
| 3a | The rounds the fence allows are spent while the model still asks for tools | The last request can call no tool, and the model writes the answer from what the tools returned |
| 3b | The host moved to a new commit while the chat runs | The message reads the model's types and questions again, and the answer names the new commit |
| 4a | The message is in English words with German word order, on the German page | It is answered in the language its words are in |
...

```

# A requirement

One statement a product is held to, with its kind, its modality, the fit criterion that shows it met, and what it applies to.

RESTS ON

requirement A question is answered in the language of its words

model/requirements/a-question-is-answered-in-the-language-of-its-words.md

```

---
id: 01a1312d-4501-7e47-abc4-8b179c53d140
source: Local
kind: functional
modality: must
---

# A question is answered in the language of its words
> An answer is written in the language the message's words are in, and in the page's
  language only where the words do not tell.

...

### Fit criterion
A message in English words with German word order, sent from the German page, is answered
  in English; and a German answer written after a tool round names each entity in Swiss
  Standard German first with its exact title in parentheses after.

### Applies to
| Type | Entity |
| --- | --- |
| use-case | Ask the company a question |

...

```

# Three kinds, three words

Functional, quality or constraint. Must, should or may, three of RFC 2119's words. A row each from our model.

RESTS ON

```
requirement A broken reference is marked when the line is left
```

```
requirement A German reader reads the vocabulary pages in German
```

```
requirement An instance may make its own ids under a declared pattern
```

```
requirement No call changes the model
```

quality·must

A broken reference is marked when the line is left

functional·should

A German reader reads the vocabulary pages in German

functional·may

An instance may make its own ids under a declared pattern

constraint·must

No call changes the model

# A rule carried in

A requirement names the company rule it carries into the product in `follows`. The trace runs from the obligation to the fit criterion.

RESTS ON

requirement A visitor is told before the first message that an agent answers

rule A visitor is told an agent answers

feature A visitor asks the model

model/requirements/a-visitor-is-told-before-the-first-message-that-an-agent-answers.md

```
---
id: 01a1312c-f784-7d5d-b36c-d380130e0b09
source: Local
kind: functional
modality: must
follows:
  - A visitor is told an agent answers
---

# A visitor is told before the first message that an agent answers
> The chat tells the visitor, before the first message is sent and in the language of the
  page, that an agent writes the answer from the model and which company's language model
  it goes through.

...

### Applies to
| Type | Entity |
| --- | --- |
| feature | A visitor asks the model |

...
```

# Where it departs

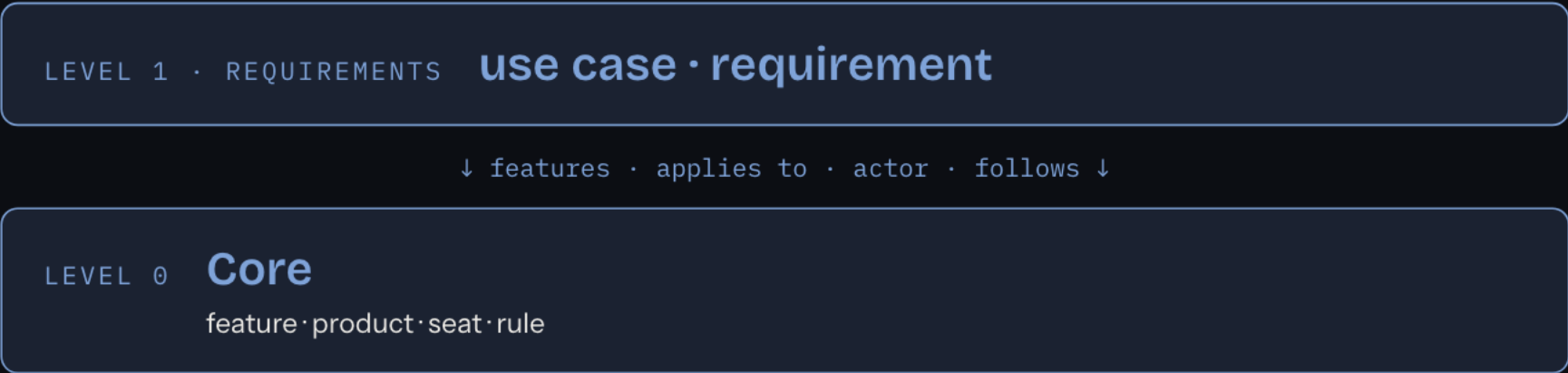
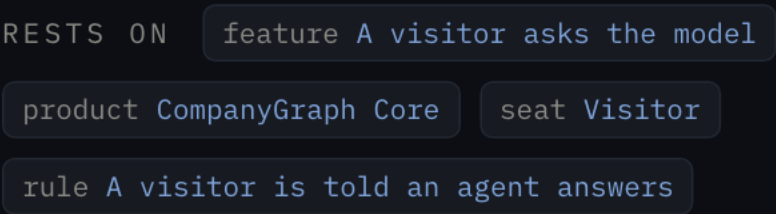
Where the pack does not do what its sources do, its README says so and says why.

RESTS ON concept Pack

|                                                                           |                                                                                                                                                                                                         |
|---------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| core's rule: <code>must</code> , <code>must not</code> , <code>may</code> | RFC 2119's <code>must</code> , <code>should</code> and <code>may</code> : a requirement has a <code>should</code> , and a <code>must not</code> is a negated <code>must</code> written in the statement |
| One table of guarantees with a <code>kind</code> column                   | two sections under Cockburn's own headings, <code>minimal</code> and <code>success</code>                                                                                                               |
| Black-box and white-box scope                                             | dropped: the steps show how much of the inside a scenario reaches                                                                                                                                       |
| Goal levels as icons, five of them                                        | Cockburn's words, and the lowest left out, since his advice for a goal that low is not to write it                                                                                                      |
| An underlined step calls a use case                                       | the step's actor resolves, and the use case it calls is drawn from <code>includes</code> and named in the action                                                                                        |
| Priority, satisfaction, conflicts, history                                | none: each moves or judges one requirement against another                                                                                                                                              |
| spec-kit's measurable success criterion                                   | a requirement, of kind <code>quality</code> where it measures a quality, its measure the fit criterion                                                                                                  |

# How the pack joins core

A use case asks something of a feature and its actors are seats. A requirement applies to a feature, a product or a use case and follows a rule. Every edge runs one way and is optional.



# Left for later

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

|       |                                                                             |
|-------|-----------------------------------------------------------------------------|
| LATER | Cockburn's technology and data variations                                   |
| LATER | secondary actors as a field                                                 |
| LATER | extend as a relation between use cases                                      |
| LATER | user stories and use-case slices as types, which are backlog items and move |
| LATER | goals in KAOS's or iStar's sense                                            |
| LATER | a stakeholder type beyond the seat                                          |
| LATER | SysML's derive, satisfy and verify                                          |
| LATER | a requirement's originator                                                  |

RESTS ON [concept Pack](#)

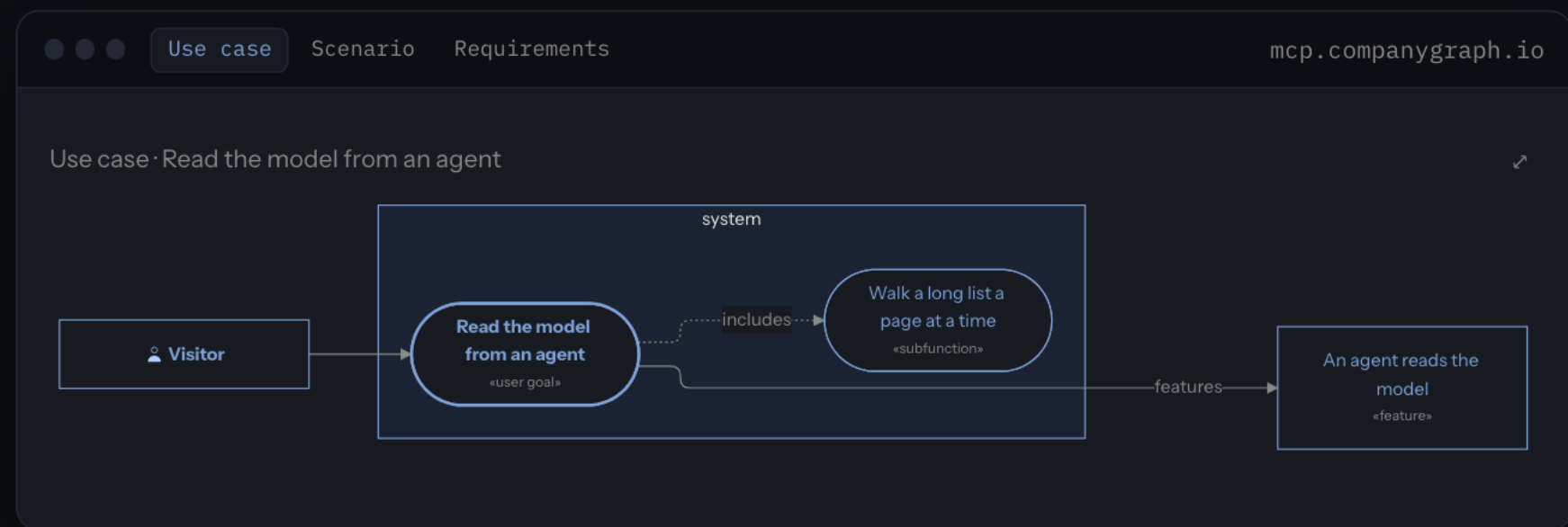
# A use case, drawn

Nobody drew these. The MCP server's own code draws them from our pages at the commit this site pins — the same code that answers an agent at [mcp.companygraph.io](https://mcp.companygraph.io).

RESTS ON `use-case Read the model from an agent`

`use-case Walk a long list a page at a time`

`feature An agent reads the model`



# What someone wants, as pages

Use cases and requirements as pages a check holds and the server draws. Where the pack departs, its README says why.

RESTS ON

concept Pack

LEVEL 1 · REQUIREMENTS use case · requirement

↓ features · applies to · actor · follows ↓

LEVEL 0 **Core**

feature · product · seat · rule