Umlimator Atlas

2026-11-19 · `examples/umlimator-atlas.json`

12 views · 86 cards · 102 lines

Contents

Umlimator 2
What umlimator is made of, and who uses each part. Double-click the module, umlimator/all, the host page, the page, the MCP page or the diff Action to go in.

umlimator: system context 4

umlimator: containers 6

The module's packages, in build order 8

umlimator/all: the typed families 9

The diff Action on a pull request 10

umlimator's state machines 12
The machines a host, a reader or an agent drives, and the state they all report. Double-click one for its states.

Navigation 13
One move at a time, and the latest request wins. setView() resolves true once its view shows, else false.

Story player 14
player.state: entering a step, playing or paused on it, then ended or stopped. A story event comes on each change.

Tour and replay 15
One trip at a time. tour() and a replay's done resolve true at the end, false when cut short.

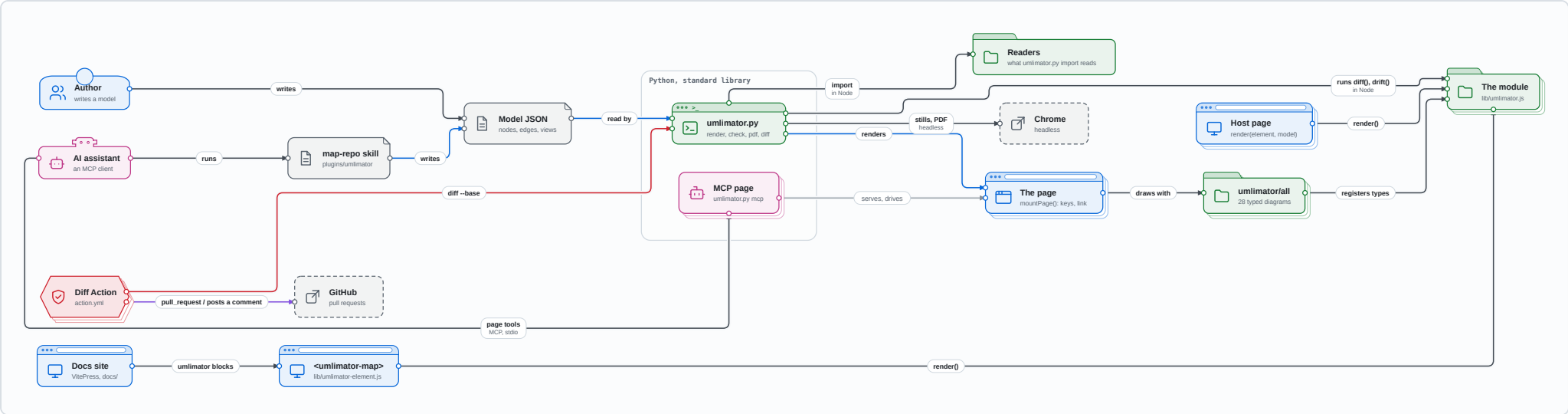
The page's keys 17
Esc and Backspace close the top open layer first, in this order. On the map, Esc ends a tour or replay, then clears the selection, then goes up.

The MCP page 18
A page tool, step by step: it answers "shown" only once a page has shown it.

Umlimator

16 cards · 18 lines · 1 box

What umlimator is made of, and who uses each part. Double-click the module, umlimator/all, the host page, the page, the MCP page or the diff Action to go in.



Legend **Hook / guard** fires on its own · **Client / UI** people and screens · **Service / API** runs the logic · **Agent / AI** an agent or model · **External** outside the system · **File / Doc** a document or config · **Person** someone who uses it · **Browser app** runs in a web browser · **Agent** an AI agent · **Command line** a CLI or a script · **Package** a module or a folder of code · ⤢ **Stacked** opens a view of its own · **calls** runs or invokes it · **feeds** hands work on · **triggers** suggests, offers, fires · **state** reads or writes a file, board, config · **guard** a hook steps in

Cards 16

Card	Kind	What it is
Author `writes a model`	Client / UI	Whoever draws a system: a person in an editor, or an assistant on their behalf.
AI assistant `an MCP client`	Agent / AI	An AI assistant that reads a model and shows its reader what it means.
map-repo skill `plugins/umlimator`	File / Doc	An agent skill that maps a repository: it reads the code and writes a model of it.
Model JSON `nodes, edges, views`	File / Doc	One file holds the whole map: cards, lines, the views that show them, stories, lenses and rules.
Readers `what umlimator.py import reads`	Service / API	The importers: a diagram written in another notation, read into a model.
umlimator.py `render, check, pdf, diff`	Service / API	The command line: one Python file, standard library only.

Cards 16 (continued)

Card	Kind	What it is
The module ⤢ `lib/umlimator.js`	Service / API	The core: `render()`, `layout()`, `validate()`, `check()`, `diff()` and the map's canvas, in one file with no dependency.
umlimator/all ⤢ `28 typed diagrams`	Service / API	The module with every typed diagram: UML, SysML, C4, ER, flowchart, data flow, BPMN, mindmap and three clouds.
The page ⤢ `mountPage(): keys, link`	Client / UI	The standalone page around the map, `mountPage()`: breadcrumbs, search, menus, the details panel and the story recorder.
MCP page ⤢ `umlimator.py mcp`	Agent / AI	The page `umlimator.py mcp` serves, which an AI assistant drives with its page tools: `show_view`, `highlight`, `tour`, `play_story` and `clear`.

Umlimator (continued)

Cards 16 (continued)

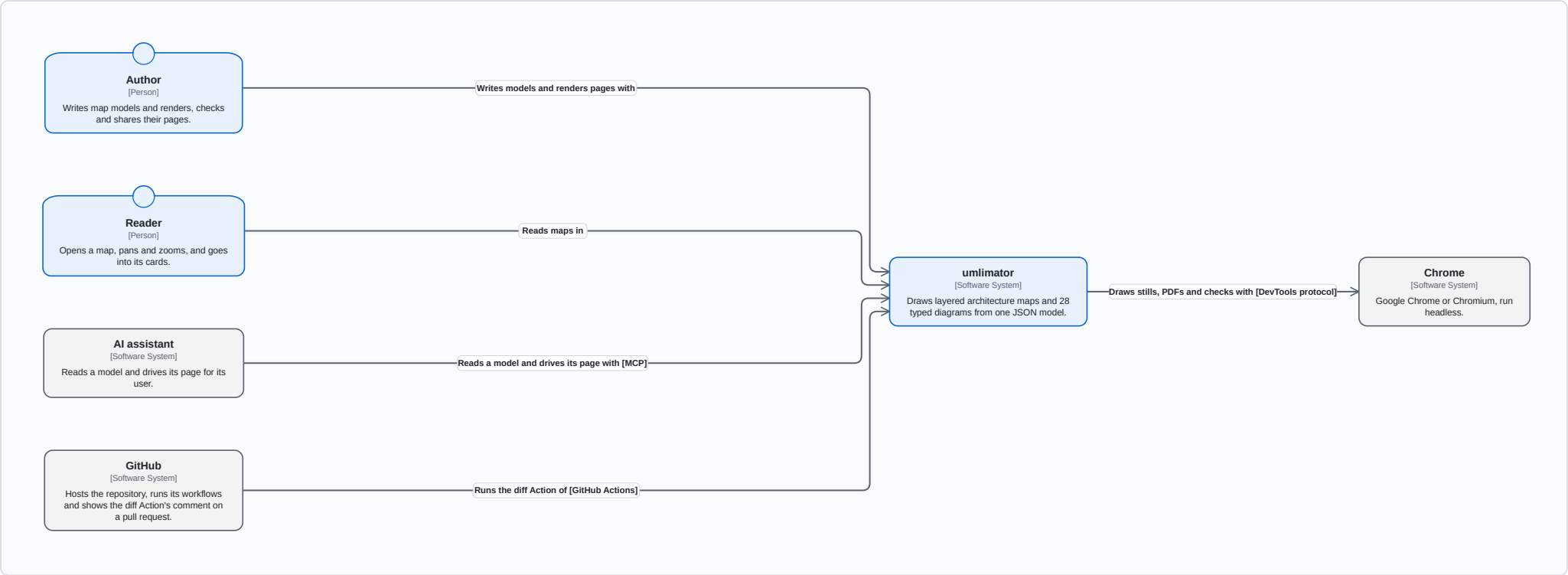
Card	Kind	What it is
Host page ⤢ `render(element, model)`	Client / UI	A web page that embeds a map with `render()` and drives it through its handle.
<umlimator-map> `lib/umlimator-element.js`	Client / UI	A custom element: a map from HTML alone, its still shown until the map is drawn.
Docs site `VitePress, docs/`	Client / UI	The guide, the API and the conformance notes, published with this gallery.
Diff Action ⤢ `action.yml`	Hook / guard	A GitHub Action: what a pull request changes in its map models, posted on the pull request.
GitHub `pull requests`	External	Where the code lives: a pull request starts the diff Action, and shows its comment.
Chrome `headless`	External	Google Chrome or Chromium, which the CLI drives to draw what a page shows.

Lines 18

From → to	Label	Technology
Author → Model JSON	writes	
AI assistant → map-repo skill	runs	
map-repo skill → Model JSON	writes	
Model JSON → umlimator.py	read by	
umlimator.py → Readers	import	in Node
umlimator.py → The module	runs diff(), drift()	in Node
umlimator.py → The page	renders	
umlimator.py → Chrome	stills, PDF	headless
The page → umlimator/all	draws with	
umlimator/all → The module	registers types	
Host page → The module	render()	
<umlimator-map> → The module	render()	
Docs site → <umlimator-map>	umlimator blocks	
AI assistant → MCP page	page tools	MCP, stdio
MCP page → The page	serves, drives	
GitHub → Diff Action	pull_request	
Diff Action → umlimator.py	diff --base	
Diff Action → GitHub	posts a comment	

Boxes 1

Box	Holds
Python, standard library	umlimator.py and MCP page



Legend **Person** a person **Software system** a software system **External software system** outside the system, drawn grey **Relationship** uses, with its technology in brackets

Elements 6

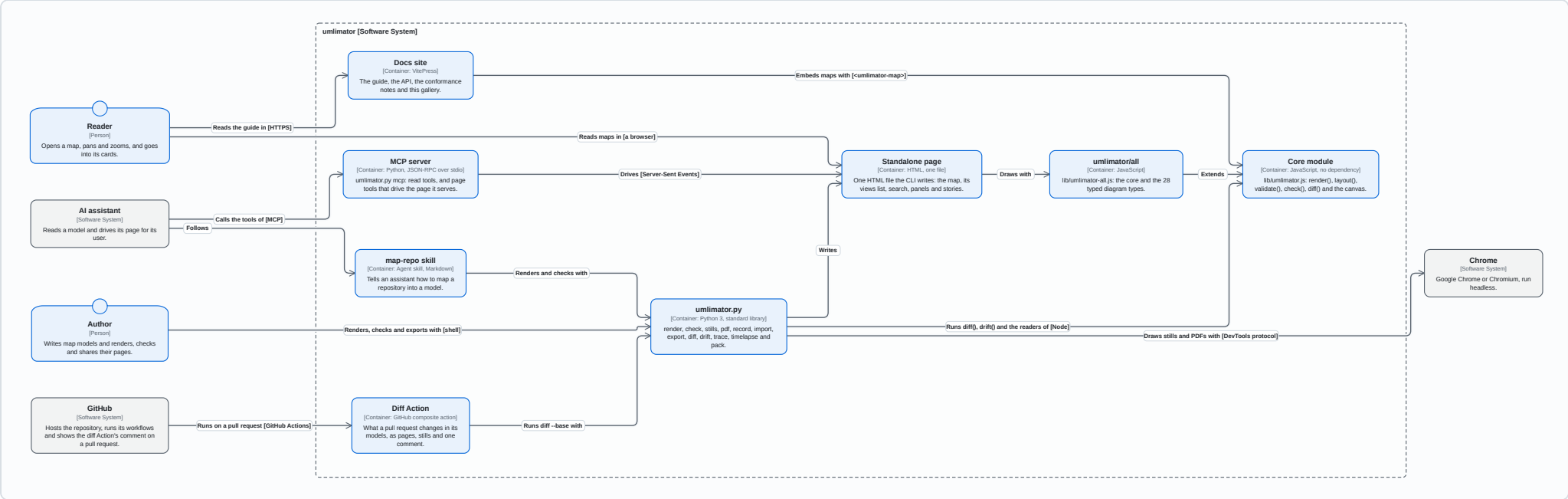
Card	Kind	Members
Author	Person	Writes map models and renders, checks and shares their pages.
Reader	Person	Opens a map, pans and zooms, and goes into its cards.
umlimator	Software system	Draws layered architecture maps and 28 typed diagrams from one JSON model.
AI assistant	Software system	Reads a model and drives its page for its user. external

Elements 6 (continued)

Card	Kind	Members
GitHub	Software system	Hosts the repository, runs its workflows and shows the diff Action's comment on a pull request. external
Chrome	Software system	Google Chrome or Chromium, run headless. external

Lines 5

From → to	Label	Kind
Author → umlimator	Writes models and renders pages with	Relationship
Reader → umlimator	Reads maps in	Relationship
AI assistant → umlimator	Reads a model and drives its page with [MCP]	Relationship
GitHub → umlimator	Runs the diff Action of [GitHub Actions]	Relationship
umlimator → Chrome	Draws stills, PDFs and checks with [DevTools protocol]	Relationship



Legend **Person** a person **Container** a container **External software system** outside the system, drawn grey **Relationship** uses, with its technology in brackets

Elements 14

Card	Kind	Members
umlimator	Software system	Draws layered architecture maps and 28 typed diagrams from one JSON model.
Author	Person	Writes map models and renders, checks and shares their pages.
Reader	Person	Opens a map, pans and zooms, and goes into its cards.
umlimator.py	Container	Python 3, standard library render, check, stills, pdf, record, import, export, diff, drift, trace, timelapse and pack.
Core module	Container	JavaScript, no dependency lib/umlimator.js: render(), layout(), validate(), check(), diff() and the canvas.

Elements 14 (continued)

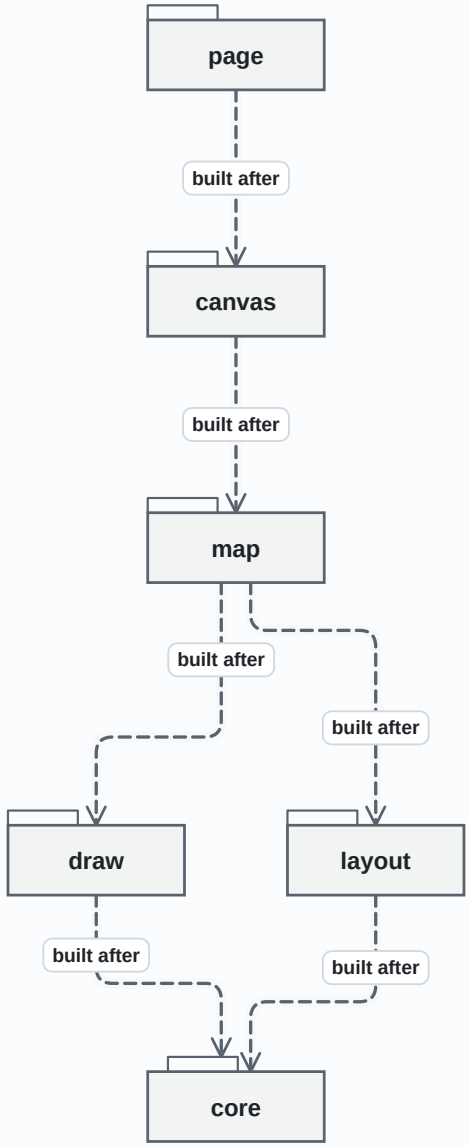
Card	Kind	Members
umlimator/all	Container	JavaScript lib/umlimator-all.js: the core and the 28 typed diagram types.
Standalone page	Container	HTML, one file One HTML file the CLI writes: the map, its views list, search, panels and stories.
MCP server	Container	Python, JSON-RPC over stdio umlimator.py mcp: read tools, and page tools that drive the page it serves.
Diff Action	Container	GitHub composite action What a pull request changes in its models, as pages, stills and one comment.

Elements 14 (continued)

Card	Kind	Members
map-repo skill	Container	Agent skill, Markdown Tells an assistant how to map a repository into a model.
Docs site	Container	VitePress The guide, the API, the conformance notes and this gallery.
AI assistant	Software system	Reads a model and drives its page for its user. external
GitHub	Software system	Hosts the repository, runs its workflows and shows the diff Action's comment on a pull request. external
Chrome	Software system	Google Chrome or Chromium, run headless. external

Lines 15

From → to	Label	Kind
Author → umlimator.py	Renders, checks and exports with [shell]	Relationship
Reader → Standalone page	Reads maps in [a browser]	Relationship
Reader → Docs site	Reads the guide in [HTTPS]	Relationship
umlimator.py → Standalone page	Writes	Relationship
umlimator.py → Core module	Runs diff(), drift() and the readers of [Node]	Relationship
Standalone page → umlimator/all	Draws with	Relationship
umlimator/all → Core module	Extends	Relationship
Docs site → Core module	Embeds maps with [<umlimator-map>]	Relationship
AI assistant → MCP server	Calls the tools of [MCP]	Relationship
AI assistant → map-repo skill	Follows	Relationship
map-repo skill → umlimator.py	Renders and checks with	Relationship
MCP server → Standalone page	Drives [Server-Sent Events]	Relationship
GitHub → Diff Action	Runs on a pull request [GitHub Actions]	Relationship
Diff Action → umlimator.py	Runs diff --base with	Relationship
umlimator.py → Chrome	Draws stills and PDFs with [DevTools protocol]	Relationship



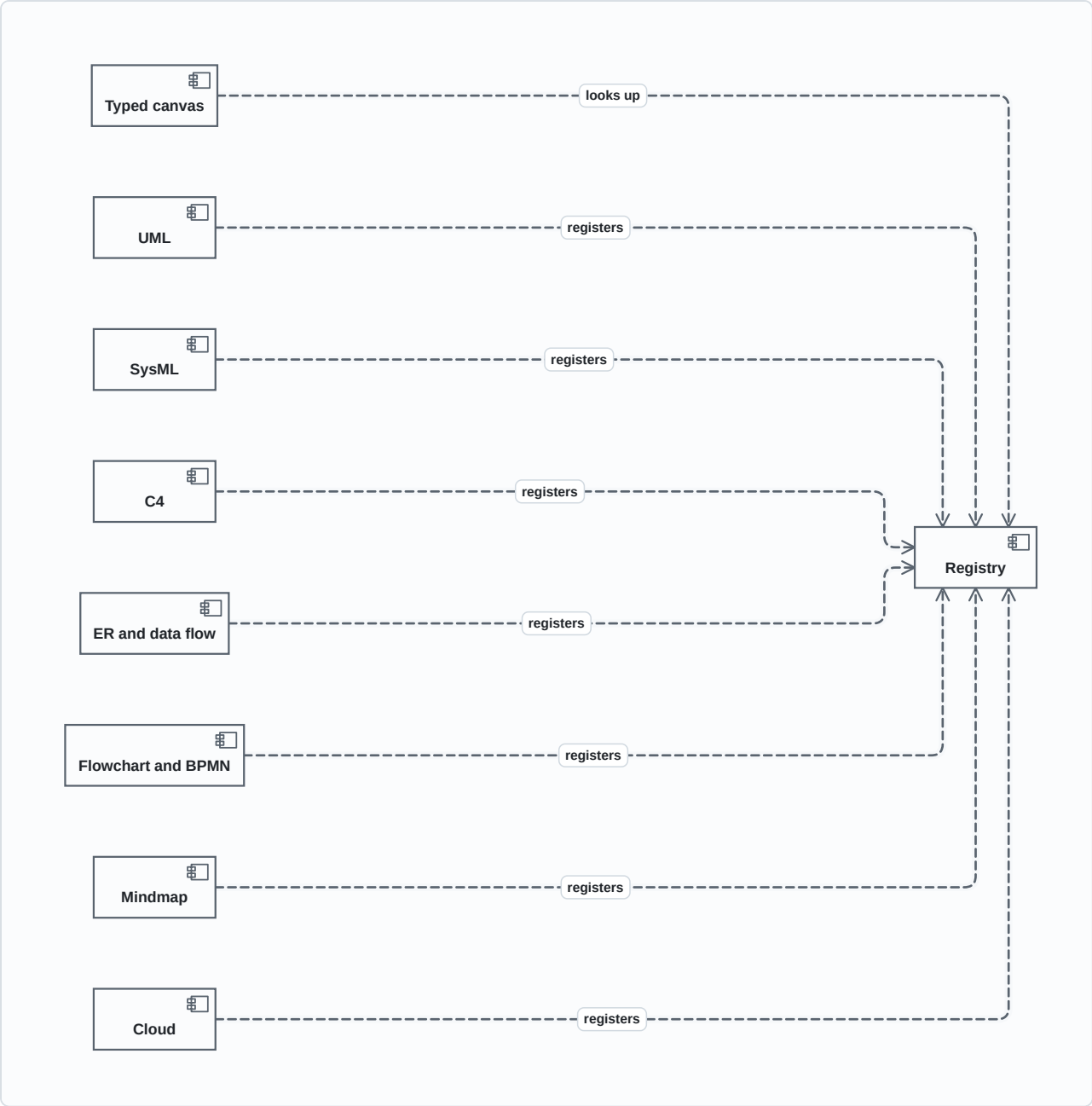
Legend --> **Dependency** uses the one it points to

Elements 6

Card	Kind	Members
page	Package	
canvas	Package	
map	Package	
draw	Package	
layout	Package	
core	Package	

Lines 6

From → to	Label	Kind
draw → core	built after	Dependency
layout → core	built after	Dependency
map → draw	built after	Dependency
map → layout	built after	Dependency
canvas → map	built after	Dependency
page → canvas	built after	Dependency



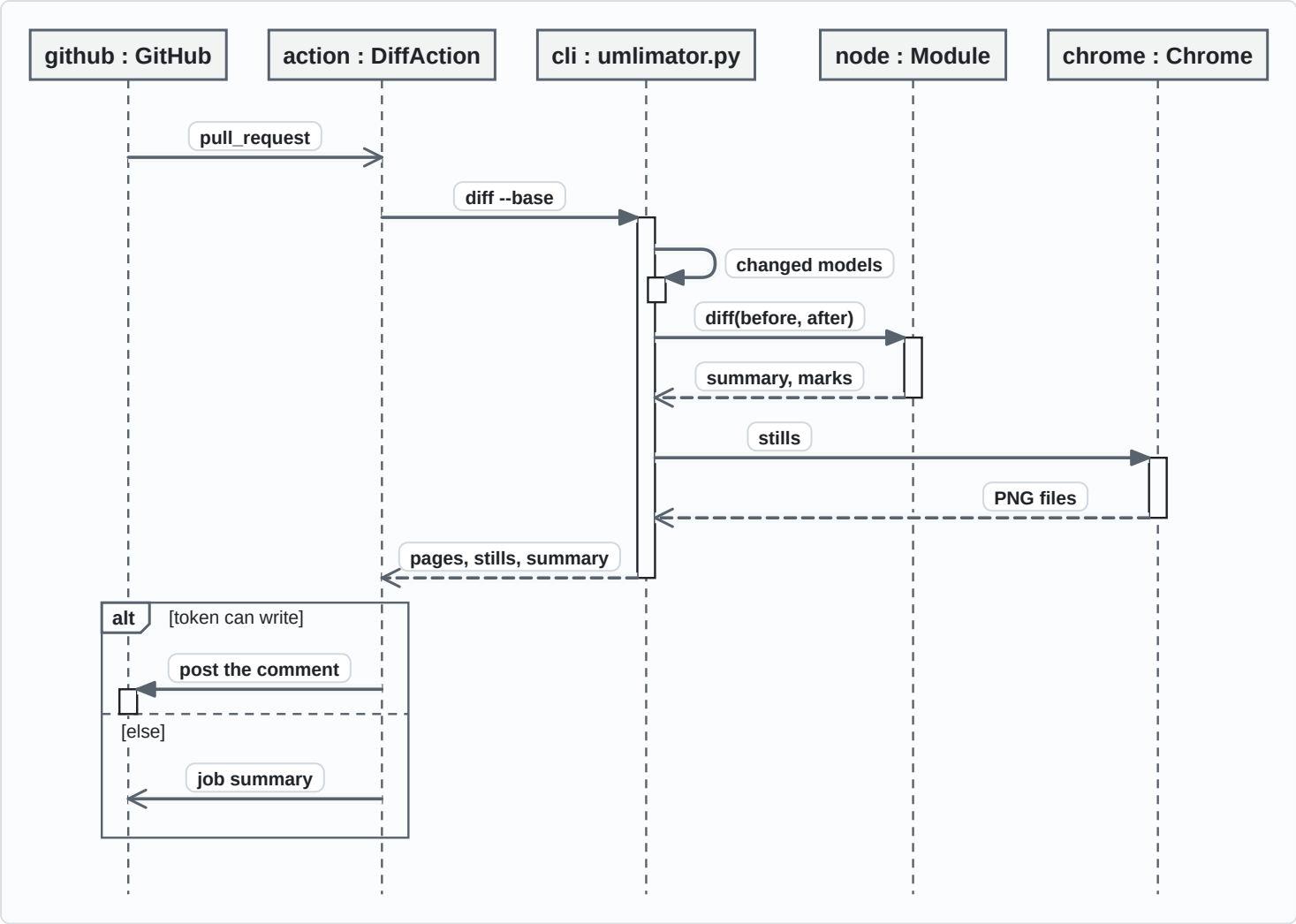
Legend --> **Dependency** uses the one it points to

Elements 9

Card	Kind	Members
Registry	Component	
Typed canvas	Component	
UML	Component	
SysML	Component	
C4	Component	
ER and data flow	Component	
Flowchart and BPMN	Component	
Mindmap	Component	
Cloud	Component	

Lines 8

From → to	Label	Kind
UML → Registry	registers	Dependency
SysML → Registry	registers	Dependency
C4 → Registry	registers	Dependency
ER and data flow → Registry	registers	Dependency
Flowchart and BPMN → Registry	registers	Dependency
Mindmap → Registry	registers	Dependency
Cloud → Registry	registers	Dependency
Typed canvas → Registry	looks up	Dependency



Legend → **Call** the sender waits for it → **Async** the sender goes on --> **Reply** the answer to a call

Elements 5

Card	Kind	Members
github : GitHub	Lifeline	
action : DiffAction	Lifeline	

Elements 5 (continued)

Card	Kind	Members
cli : umlimator.py	Lifeline	
node : Module	Lifeline	

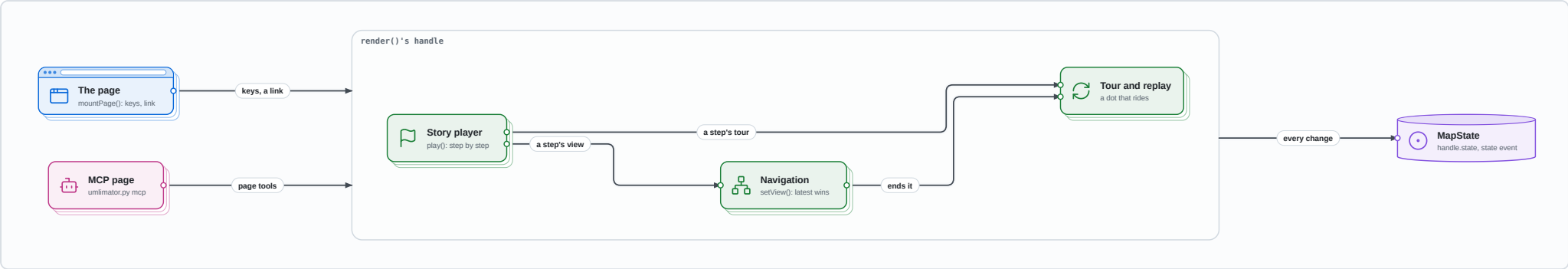
Elements 5 (continued)

Card	Kind	Members
chrome : Chrome	Lifeline	

Lines 10

From → to	Label	Kind
github : GitHub → action : DiffAction	pull_request	Message
action : DiffAction → cli : umlimator.py	diff --base	Message
cli : umlimator.py → cli : umlimator.py	changed models	Message
cli : umlimator.py → node : Module	diff(before, after)	Message
node : Module → cli : umlimator.py	summary, marks	Message
cli : umlimator.py → chrome : Chrome	stills	Message
chrome : Chrome → cli : umlimator.py	PNG files	Message
cli : umlimator.py → action : DiffAction	pages, stills, summary	Message
action : DiffAction → github : GitHub	post the comment	Message
action : DiffAction → github : GitHub	job summary	Message

The machines a host, a reader or an agent drives, and the state they all report. Double-click one for its states.



Legend **Client / UI** people and screens · **Service / API** runs the logic · **Agent / AI** an agent or model · **Data store** keeps data · ⤢ **Stacked** opens a view of its own · **calls** runs or invokes it

Cards 6

Card	Kind	What it is
The page ⤢ `mountPage(): keys, link`	Client / UI	The standalone page around the map, `mountPage()`: breadcrumbs, search, menus, the details panel and the story recorder.
MCP page ⤢ `umlimator.py mcp`	Agent / AI	The page `umlimator.py mcp` serves, which an AI assistant drives with its page tools: `show_view`, `highlight`, `tour`, `play_story` and `clear`.
Story player ⤢ `play(): step by step`	Service / API	A story told step by step: `play()`, the page's Stories menu, or a link's `story=` and `step=`.
Navigation ⤢ `setView(): latest wins`	Service / API	A move from one view to another: `setView()`, a drill-down, up, a story's step, or a link.
Tour and replay ⤢ `a dot that rides`	Service / API	One trip at a time: `tour()`, `replay()`, or a story step's tour.
MapState `handle.state, state event`	Data store	The map's state as one snapshot: fresh, frozen and plain JSON on every read. The `state` event says which of its keys changed.

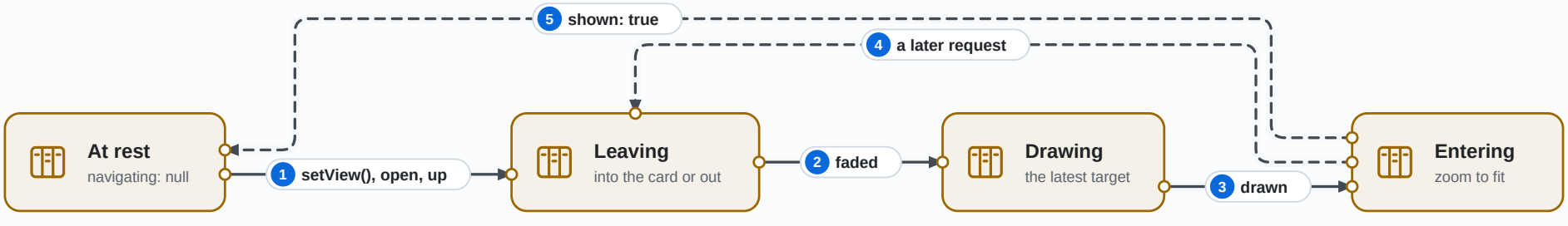
Lines 6

From → to	Label	Technology
The page → render()'s handle	keys, a link	
MCP page → render()'s handle	page tools	
Story player → Navigation	a step's view	
Story player → Tour and replay	a step's tour	
Navigation → Tour and replay	ends it	
render()'s handle → MapState	every change	

Boxes 1

Box	Holds
render()'s handle	Story player, Navigation and Tour and replay

One move at a time, and the latest request wins. setView() resolves true once its view shows, else false.



Legend State board, config, ledgers — calls runs or invokes it — loops back the one line that runs right to left

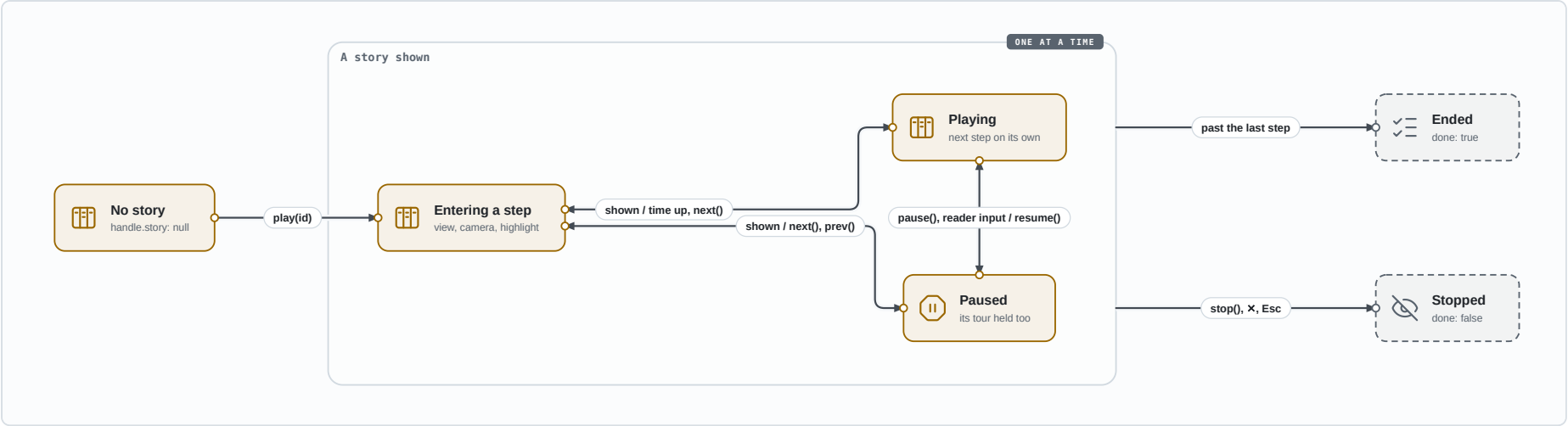
Cards 4

Card	Kind	What it is
At rest `navigating: null`	State	A view is shown and no move is under way.
Leaving `into the card or out`	State	The camera zooms into the card that opens the view, or out, and the view fades (130 ms). Skipped under reduced motion.
Drawing `the latest target`	State	The view asked for last is drawn, and the selection is cleared (a `select` event with `null`).
Entering `zoom to fit`	State	The camera zooms to fit the new view. A press on the map settles the glide, and the move goes on.

Lines 5

#	From → to	Label	Technology
1	At rest → Leaving	setView(), open, up	
2	Leaving → Drawing	faded	
3	Drawing → Entering	drawn	
4	Entering → Leaving	a later request	
5	Entering → At rest	shown: true	

player.state: entering a step, playing or paused on it, then ended or stopped. A story event comes on each change.



Legend **State** board, config, ledgers **External** outside the system **calls** runs or invokes it

Cards 6

Card	Kind	What it is
No story `handle.story: null`	State	`handle.story` is `null`. `play()` of a story the model lacks returns `null` and leaves the story shown.
Entering a step `view, camera, highlight`	State	The step's view, camera, selection and highlight are on their way. The `story` event comes first.
Playing `next step on its own`	State	The next step comes once this one's time is up (4 s unless the step says) and its tour has ended.
Paused `its tour held too`	State	`pause()`, ❚❚ on the bar, or the reader taking the map: a press, a wheel that zooms, a key, a view or a selection the story didn't make.
Ended `done: true`	External	Past the last step: `next()`, ✓ on the bar, or the last step's time. `player.done` resolves `true`.
Stopped `done: false`	External	`stop()`, ×, Esc, `play(null)`, another `play()`, an `update()` whose model lacks the story, or `destroy()`. `player.done` resolves `false`.

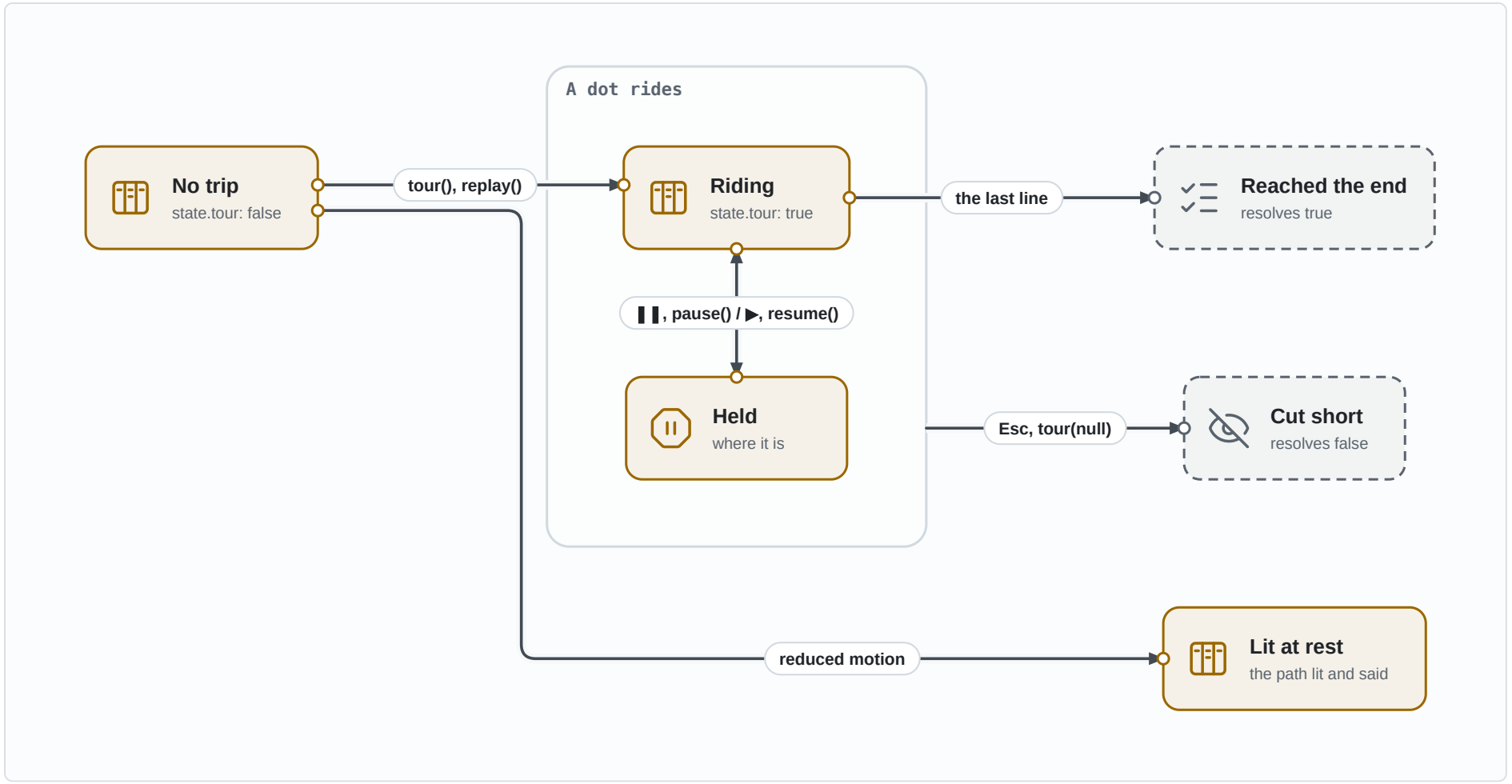
Lines 9

From → to	Label	Technology
No story → Entering a step	play(id)	
Entering a step → Playing	shown	
Playing → Entering a step	time up, next()	
Entering a step → Paused	shown	
Paused → Entering a step	next(), prev()	
Playing → Paused	pause(), reader input	
Paused → Playing	resume()	
A story shown → Ended	past the last step	
A story shown → Stopped	stop(), ×, Esc	

Boxes 1

Box	Holds
A story shown	Entering a step, Playing and Paused

One trip at a time. tour() and a replay's done resolve true at the end, false when cut short.



Legend **State** board, config, ledgers **External** outside the system **calls** runs or invokes it

Cards 6

Card	Kind	What it is
No trip `state.tour: false`	State	No dot rides. `tour()` and `replay()` start one; a new one ends the one before.
Riding `state.tour: true`	State	The dot rides the line between each pair of cards in turn, and each card it reaches flashes.
Held `where it is`	State	The page's ❙❙, a paused story (for its step's tour), or the replay's `pause()`.
Reached the end `resolves true`	External	`tour()` resolves `true`, and so does a replay's `done`; a replay's labels stay.
Cut short `resolves false`	External	Esc on the map, `tour(null)`, the next tour or replay, a new picture of the view, the replay's `stop()`, or `destroy()`.
Lit at rest `the path lit and said`	State	Under reduced motion no dot rides: the path is lit in the `tour` layer and said, and `tour()` resolves `true` at once. It stays until Esc, `tour(null)`, the next trip or a new picture.

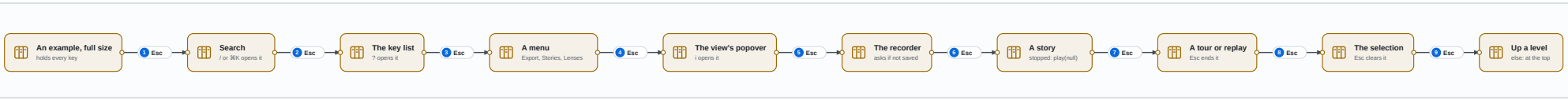
Lines 6

From → to	Label	Technology
No trip → Riding	tour(), replay()	
Riding → Held	❙❙, pause()	
Held → Riding	▶, resume()	
Riding → Reached the end	the last line	
A dot rides → Cut short	Esc, tour(null)	
No trip → Lit at rest	reduced motion	

Boxes 1

Box	Holds
A dot rides	Riding and Held

Esc and Backspace close the top open layer first, in this order. On the map, Esc ends a tour or replay, then clears the selection, then goes up.



Legend **State** board, config, ledgers **calls** runs or invokes it

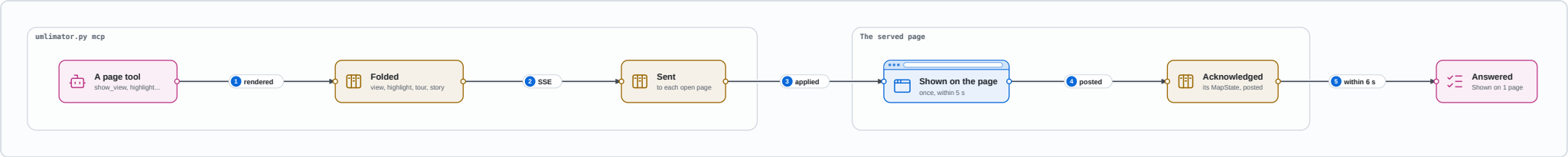
Cards 10

Card	Kind	What it is
An example, full size `holds every key`	State	An example shown full size, from a thumbnail in the details panel. It holds every key while open; Esc closes it and focus goes back to the thumbnail.
Search `/ or ⌘K opens it`	State	Esc closes it and gives focus back; ⌘K (Ctrl K) closes it too.
The key list `? opens it`	State	One of search, the key list and the menus is open at a time.
A menu `Export, Stories, Lenses`	State	Esc closes it and gives focus back to its button.
The view's popover `i opens it`	State	The popover of the view's pill.
The recorder `asks if not saved`	State	Export ▸ Record a story...: Esc closes it, asking first while it holds steps not saved.
A story `stopped: play(null)`	State	Esc or Backspace stops the story shown.
A tour or replay `Esc ends it`	State	Esc ends it and says so: "Tour stopped", "Replay stopped".
The selection `Esc clears it`	State	
Up a level `else: at the top`	State	Esc or Backspace with nothing selected goes up a level; at the top, the hint line says "Already at the top".

Lines 9

#	From → to	Label	Technology
1	An example, full size → Search	Esc	
2	Search → The key list	Esc	
3	The key list → A menu	Esc	
4	A menu → The view's popover	Esc	
5	The view's popover → The recorder	Esc	
6	The recorder → A story	Esc	
7	A story → A tour or replay	Esc	
8	A tour or replay → The selection	Esc	
9	The selection → Up a level	Esc	

A page tool, step by step: it answers "shown" only once a page has shown it.



Legend **State** board, config, ledgers · **Client / UI** people and screens · **Agent / AI** an agent or model · **calls** runs or invokes it

Cards 6

Card	Kind	What it is
A page tool `show_view, highlight...`	Agent / AI	`show_view`, `highlight`, `tour`, `play_story` or `clear`. A view, card, line or story the model lacks is an error, and nothing is sent.
Folded `view, highlight, tour, story`	State	The latest of each, so a page that opens later shows what the assistant built, not its last command.
Sent `to each open page`	State	One event (Server-Sent Events) to each open page.
Shown on the page `once, within 5 s`	Client / UI	The page shows each command once, one at a time; one that takes more than 5 s is answered as not shown.
Acknowledged `its MapState, posted`	State	The page posts its `MapState` with the command's id, or the error that stopped it.
Answered `Shown on 1 page`	Agent / AI	"Shown on 1 page at ..., on view ...". Otherwise the tool says the page could not show it and why, that the open pages did not show it within 6 s, or that no page is connected and the URL to open.

Lines 5

#	From → to	Label	Technology
1	A page tool → Folded	rendered	
2	Folded → Sent	SSE	
3	Sent → Shown on the page	applied	
4	Shown on the page → Acknowledged	posted	
5	Acknowledged → Answered	within 6 s	

Boxes 2

Box	Holds
umlimator.py mcp	A page tool, Folded and Sent
The served page	Shown on the page and Acknowledged