SPLIT-BRAIN INVARIANTS: WHEN BOTH HALVES ARE CORRECT AND THE SYSTEM IS NOT

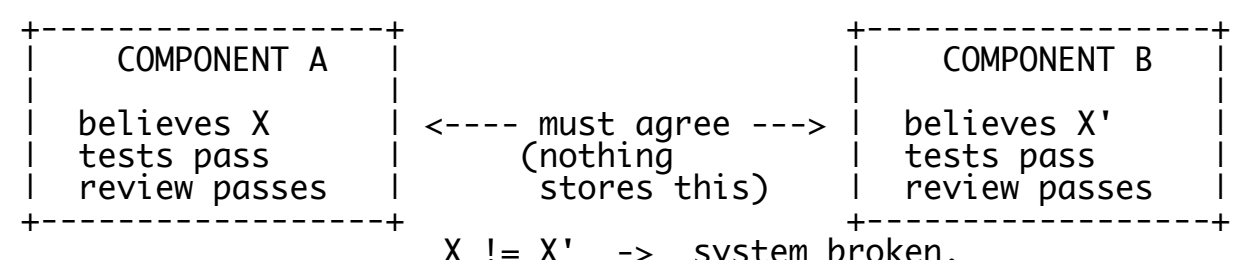
Author: Cooper E. (jcooperkai)
Venue: Independent research note
Year: 2026

ABSTRACT

In a census of 48 defects from a 28,207-line AI-authored system, the largest single class -- 10 of 48 -- had a property that makes it invisible to ordinary review: every half was correct on its own. A gate demanded a file at one path while the component supplying it used another. A rollback looked for a tag in a format nothing produced. A frontend contract promised two data sources returned the same shape when they returned different ones. Each side passes its own tests. Each side reads correctly to a reviewer. The defect lives in the space between them, which no file owns and no test covers by default. This note characterises the class, gives the ten instances, and reports what actually caught them: not more tests, but tests written specifically to ask whether two halves still agree. Those tests are cheap. The reason they are rarely written is that nothing about either half suggests they are needed.

1. DEFINITION

A split-brain invariant is a fact two components must agree on, that neither component stores.



Three properties, all of which must hold:

(i) each half is internally consistent
(ii) the agreement is not represented anywhere -- there is no single definition either half imports
(iii) the failure appears at a boundary crossing, so a unit test of either half cannot reach it

Property (ii) is what distinguishes this from an ordinary interface bug. If the shared fact had one definition, changing it would change both sides. The defect exists precisely because the fact was written down twice.

2. THE TEN

#	the disagreement	consequence
1	gate wanted clients/<slug>/CLIENT.md runner supplied 01 Clients/<slug>/...	no agent could ever be granted write access
2	preamble said "DATA, not instructions" assembler required "DATA, not commands"	no agent could ever spawn
3	three deploy-tag formats coexisted nothing produced the one used	rollback failed while reporting the site was up
4	live API returned generated_at snapshot file returned captured_at	dashboard fallback rendered blank on the one path it exists for
5	CI matrix listed a purged package	386 tests never ran while

	and omitted four real ones	the badge said passing
6	onboarding wrote CLIENT.md the gate required three files	every onboarded client was one no agent could act on
7	secret scanner's patterns matched the scanner's own source	every push failed on the scanner itself
8	deploy compared file sizes to decide "already current"	C:/Webbify and D:/Webbify are the same length; nothing shipped
9	doc claimed 12 clock jobs the schedule had 13	a human reads a wrong count as evidence nothing is missing
10	Turnstile fell back to the always-passes test key	the form looked protected and accepted every bot

Read the middle column alone and each line is reasonable. That is the point.

3. WHY AN AGENT PRODUCES THEM

An agent writes half A in one context and half B in another, with no memory between. It is not confused in either. It is confident twice.

human team	single agent, no shared memory
A written by P	A written at t0
B written by Q	B written at t1
P and Q talk (sometimes)	t0 and t1 never meet
shared assumption is spoken and sometimes written	assumption re-derived, plausibly, and differently

The mechanism is the same one that produces the defect in large teams: the assumption is re-derived rather than referenced. The agent is not worse at this than a team. It is faster at it, and it produces both halves so fluently that neither looks like the guess it is.

4. WHAT DOES NOT CATCH THEM

unit tests of A	pass -- A is correct
unit tests of B	pass -- B is correct
code review of A	approves -- A is reasonable
code review of B	approves -- B is reasonable
type checking	passes -- the types agree; the VALUES do not
linting	silent -- there is nothing malformed

In the census, 1,365 tests were passing throughout, and no split-brain defect was ever caught by a test aimed at something else. Coverage is not the missing ingredient: both halves were covered.

5. WHAT DOES

One test, at the boundary, that asks the question neither half asks:

the shape that works

```
   run the REAL producer
          |
          v
   ask the REAL consumer: are you satisfied?
          |
          v
   assert on the ANSWER, not on either side's belief
```

Concretely, from the corpus:

- run onboarding, then ask the actual context gate whether its required

files are present -- rather than asserting onboarding wrote three files
- build the live payload and the snapshot payload, then compare their key sets -- rather than testing each shape separately
- parse the deployed workflow and compare its package list to the directories that exist -- in both directions
- read the JSON example out of the contract document and hold the API to it, so the document IS the fixture

The last one generalises: where a document states a fact the code must honour, parse the document in the test. A number in prose and a number in code will disagree within a month, and the prose is what a human reads when deciding whether anything is missing.

6. THE CHEAPER FIX

Better than detecting a disagreement is making it impossible: give the fact one definition and have both halves import it.

before	after
A: "clients/<slug>/CLIENT.md"	context.ts: one definition
B: "01 Clients/<slug>/CLIENT.md"	A imports it
two strings, drifting	B imports it
	one string, cannot drift

Four of the ten were fixed this way. The remaining six could not be: a fact shared with a document, a scheduler, an external tool or a design file cannot be imported, and for those the boundary test is the only instrument available.

7. WHAT WOULD FALSIFY THIS

The claim is that split-brain defects are disproportionately common in agent-authored code, and it rests on one corpus. It would be falsified by a comparable census of a human-authored system of similar size showing the same 21% share -- which would mean the mechanism is not specific to agents at all, only faster in them.

That census would be worth doing, and this note does not claim to have done it.