Faster, Better, Cleaner Java Development

🚀… with Agentic LLMs

airhacks.industries

airhacks.live

"It's not work if you like it"
…so I never worked. #java

youtube.com/
@bienadam

/@bienadam/shorts

airhacks.fm

#360 Java, LangChain4J and Enterprise LLMs

Listen on Apple Podcasts · Listen on Spotify · YouTube · [RSS]

An airhacks.fm conversation with Antonio Goncalves (@agoncal) about:

journey fro
Microsoft f
modern AI
assistance,
abstraction
standards a
Entity (BCE
integration
multi-tenar
potentially

#359 From SIMD to CUDA with TornadoVM

Listen on Appl · [RSS]

An airhack

#356 AI/LLM Driven Development

Listen on Apple Podcasts · Listen on Spotify · YouTube · [RSS]

An airhacks.fm conversation with Jonathan Ellis (@spyced) abou

GPU acc
CPU-bou
Alfonso'
Vector A
hybrid a
multiplic
compute
consum

airhacl

brokk as a Norse dwarf who forged Thor's hammer, Java Swing

airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

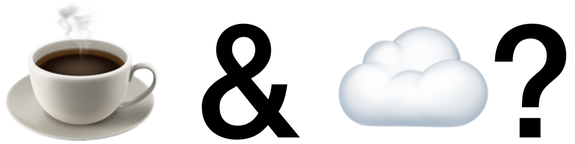
Live, Virtual Online Workshops, Summer 2026:

> LLM-Assisted Web Components: No Frameworks, No Dependencies, 09 July 2026
> ...or how to build maintainable web applications with LLMs

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io



Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

GraalVM™

reduces RAM footprint

https://sites.google.com/view/energy-efficiency-languages/results?authuser=0

LLMs and Java

	Average	Python	Cpp	Java	JS	Go	Shell	Csharp	Dart	Elixir	Julia	Kotlin	Perl	PHP	Racket	R	Ruby	Rust	Scala	Swift	TS
Count		196	186	188	184	191	188	199	200	198	200	200	200	199	196	198	200	199	199	200	199
Current Upper Bound	*74.8*	*63.3*	*74.7*	*78.7*	*59.2*	*69.1*	*70.7*	*88.4*	*78.0*	*97.5*	*78.0*	*89.5*	*64.5*	*52.8*	*88.3*	*74.2*	*79.5*	*61.3*	*77.4*	*78.0*	*61.3*
Reasoning Mode																					
Claude Opus 4 (20250514)	**52.4**	40.3	44.1	**55.9**	38.6	37.2	**51.6**	74.9	54.0	80.3	**55.5**	72.5	44.5	28.1	**68.9**	**52.5**	61.0	38.7	50.3	50.0	**47.2**
Claude Sonnet 4 (20250514)	51.1	37.2	46.8	52.7	34.8	**41.9**	48.9	72.4	53.5	81.8	49.0	71.5	**45.0**	**34.7**	**68.9**	50.5	54.5	36.2	48.2	48.0	44.2
o3-high (20250416)	51.1	40.8	47.3	53.2	**40.8**	22.0	49.5	68.3	**55.0**	80.8	54.5	72.0	44.0	32.7	53.1	47.5	59.0	**42.2**	51.3	**59.0**	**47.2**
o4-mini (2025-04-16)	50.0	**42.3**	46.8	51.6	40.2	31.4	45.2	68.3	54.0	**82.3**	49.0	**74.0**	44.0	30.2	45.4	43.4	59.0	40.2	50.3	54.0	45.7
Grok-4	50.9	41.2	**48.7**	50.0	37.5	41.4	47.3	72.4	49.5	76.8	**55.0**	70.0	44.0	27.1	63.8	48.5	**61.5**	37.7	**52.8**	51.5	40.7
Gemini2.5 Pro	48.7	40.3	47.5	53.2	37.0	37.2	45.2	70.9	54.0	68.7	54.0	72.0	41.0	29.7	52.6	49.5	56.5	24.6	46.7	49.5	41.7
DeepSeek-R1-0528	50.2	38.8	43.6	52.7	35.9	38.7	46.8	**75.4**	52.5	77.3	52.0	70.0	39.0	28.6	56.1	50.5	58.5	37.2	51.8	55.0	41.2
Seed1.6-enabled (250615)	45.3	39.8	44.6	46.3	28.3	40.8	44.1	60.3	39.5	69.7	51.0	58.0	41.5	25.6	52.6	51.0	52.0	28.6	41.7	47.5	41.2
Seed1.6-Thinking-250715	45.0	40.3	45.2	50.0	33.2	38.2	39.9	67.3	36.5	67.7	51.0	61.0	41.0	26.1	51.0	44.9	55.5	27.6	37.2	46.5	38.7
Seed1.6-Thinking-250615	44.7	38.8	47.0	49.5	38.0	31.4	38.8	62.3	41.0	70.7	45.0	68.0	39.0	25.1	47.5	47.5	50.5	30.7	39.7	41.5	40.2
GLM-4.5-enable	46.6	41.0	43.2	47.9	34.8	37.8	43.9	70.5	42.0	72.5	47.5	66.0	43.5	28.6	50.0	45.0	54.5	31.6	41.0	46.0	42.2
GLM-4.5-Air-enable	40.8	39.3	37.6	39.4	31.0	39.8	36.7	66.3	38.0	61.5	42.0	53.0	40.5	27.1	40.3	39.0	47.0	25.1	30.5	38.5	42.7
ERNIE-X1-Turbo-32K	39.6	39.4	17.8	33.2	32.6	37.4	33.9	46.0	33.0	68.9	54.0	49.5	39.5	23.9	45.3	44.3	48.0	20.8	40.4	44.0	37.7
Qwen3-235B-A22B-Thinking-2507	47.7	37.8	41.9	48.4	39.7	39.8	45.2	71.9	46.0	79.8	48.5	58.0	40.5	29.1	56.6	49.0	55.0	35.7	40.4	46.0	44.2
Qwen3-235B-A22B	45.9	36.7	43.5	47.3	36.4	37.7	42.0	70.9	45.5	68.7	46.0	60.0	39.0	29.1	52.0	47.0	56.5	31.7	43.7	41.5	41.7
Qwen3-32B	41.7	37.8	38.7	39.9	32.6	36.1	39.4	67.8	34.5	65.2	42.5	52.0	40.5	27.6	37.8	44.9	47.0	28.1	37.2	42.0	40.2
Qwen3-14B	37.6	37.8	35.5	35.1	30.4	30.4	36.2	60.8	29.0	62.1	34.5	44.5	37.5	23.1	44.9	36.9	43.5	24.6	28.6	36.0	38.7
Qwen3-8B	28.5	28.1	22.6	21.8	28.3	29.3	27.1	52.8	21.0	43.9	29.0	36.0	35.5	18.6	13.3	30.8	37.0	12.6	21.1	22.0	37.7
Qwen3-4B	24.3	27.6	17.7	22.3	25.5	24.1	28.2	42.2	13.0	33.3	20.0	29.5	34.5	16.1	11.7	23.7	27.5	8.5	20.1	20.0	39.2
Qwen3-1.7B	11.2	16.8	5.4	4.8	12.5	9.9	11.7	19.6	7.0	20.7	11.0	9.0	19.5	7.5	5.6	9.6	21.0	0.0	2.5	10.0	19.6
Non-Reasoning Mode																					
Claude Opus 4 (20250514)	**50.9**	37.8	45.7	50.0	**38.0**	35.6	47.3	73.9	**57.0**	**82.3**	**55.0**	**75.5**	43.0	26.6	**64.8**	47.0	**54.0**	**38.2**	**46.7**	**51.5**	**46.7**
Claude Sonnet 4 (20250514)	49.3	35.7	47.3	**52.7**	**38.0**	**37.7**	**47.9**	72.9	51.0	74.2	51.0	72.0	44.0	**30.7**	63.8	44.4	51.5	35.2	45.2	45.5	44.2
GPT4.1 (2025-04-14)	48.0	37.2	46.8	48.9	34.8	37.2	36.7	74.4	46.5	76.8	50.0	72.0	43.5	29.2	50.5	42.4	**54.0**	37.2	44.2	49.5	46.2
GPT4o (2024-11-20)	41.1	33.7	37.1	43.2	34.8	30.9	29.3	65.5	43.5	62.6	36.0	67.0	43.0	26.6	37.2	32.8	43.0	29.6	38.2	45.0	39.2
Gemini2.5 Flash	45.7	39.3	44.1	50.0	33.2	33.0	37.8	68.3	49.5	64.0	47.5	70.0	39.5	24.1	38.3	**51.5**	53.0	36.2	44.2	46.5	41.2
DeepSeek-V3-0324	48.1	36.7	**48.4**	**52.7**	31.5	34.6	37.8	72.9	48.0	75.8	49.0	69.0	42.5	28.1	59.2	45.0	52.5	37.2	**46.7**	48.0	43.7
DeepSeek-Coder-V2	37.7	29.1	34.9	34.0	27.7	29.8	31.4	63.8	33.5	60.6	37.5	58.5	35.5	25.1	41.8	35.4	45.0	22.6	33.2	38.0	35.7
DeepSeek-Coder-33B-Instruct	28.5	25.0	24.2	29.3	24.5	29.8	22.3	54.8	17.5	67.7	14.5	52.0	29.5	19.1	28.1	18.7	33.0	8.0	24.1	18.0	29.1
DeepSeek-Coder-6.7B-Instruct	20.5	18.9	12.9	19.7	19.6	21.5	16.0	44.2	11.5	47.5	15.5	45.5	21.5	10.6	15.3	13.1	27.5	6.0	11.1	8.0	23.6
Kimi-K2-0711-Preview	47.8	38.8	42.5	47.9	37.5	31.4	40.4	**75.9**	50.0	80.3	52.0	68.0	41.5	28.1	57.1	40.4	52.5	36.7	42.7	47.0	42.2
Hunyuan-TurboS-20250716	43.8	34.2	34.9	47.9	32.6	34.6	38.3	64.8	44.5	70.7	47.0	62.0	42.0	30.2	45.9	39.9	53.0	30.7	39.2	39.5	42.2
Hunyuan-Coder-7B-Preview	33.4	25.5	26.9	34.0	27.2	21.5	29.3	54.8	36.0	60.1	30.5	57.0	34.0	22.2	37.8	27.3	40.5	11.1	31.2	27.6	31.2
Seed1.6-disabled (250615)	42.9	35.2	40.3	46.8	32.6	34.6	35.1	70.9	42.5	69.7	45.0	62.0	39.5	23.1	49.5	40.4	46.5	28.1	32.7	40.0	42.7
Seed-Coder-8B-Instruct	32.3	23.5	23.7	33.5	28.8	22.5	20.7	54.8	30.5	57.1	33.0	52.5	34.0	25.1	36.7	28.3	35.5	15.6	29.6	28.0	31.2
Qwen3-Coder-480B-A35B-Instruct	44.8	**39.4**	41.1	51.1	27.9	31.4	41.1	63.0	36.5	73.7	49.5	63.1	41.0	27.2	56.3	42.7	51.5	25.4	42.1	47.5	41.9
Qwen3-235B-A22B-Instruct-2507	43.1	35.7	38.2	49.5	29.3	33.5	40.4	67.3	39.5	59.1	46.0	59.5	**44.5**	26.1	49.5	44.0	46.5	24.6	37.7	46.0	43.2
Qwen3-32B	31.0	26.5	21.5	29.5	28.0	25.5	24.0	59.3	27.5	52.0	28.0	45.0	34.5	21.6	28.6	22.7	36.5	16.1	26.6	30.0	35.2
Qwen3-14B	28.6	24.5	22.6	32.4	27.2	16.8	23.9	50.8	21.0	42.4	22.5	42.0	34.5	24.6	28.1	26.3	33.5	17.1	20.6	26.5	32.7
Qwen3-8B	23.3	22.4	11.3	25.0	22.8	18.3	22.3	42.2	17.0	41.4	18.5	36.0	29.5	18.1	23.5	16.2	27.0	7.5	19.1	17.5	29.1
Qwen3-1.7B	7.9	8.7	1.1	2.7	8.2	3.7	11.7	9.5	3.5	17.2	6.0	11.5	15.5	7.5	4.6	7.6	14.5	0.5	3.0	6.0	14.6
Qwen2.5-Coder-32B	35.8	29.6	27.4	33.0	29.9	23.0	29.3	58.3	34.5	59.6	35.5	56.0	38.5	26.1	35.7	31.3	40.0	23.1	29.6	39.0	35.2
Qwen2.5-72B	34.3	33.2	24.7	31.9	28.8	19.9	31.9	54.3	34.0	65.7	30.5	50.5	30.0	22.6	39.3	27.3	42.0	23.1	32.7	29.5	33.2
Qwen2.5-Coder-7B	22.5	19.9	8.6	22.3	21.2	12.6	21.3	38.7	18.5	47.0	18.0	39.0	27.5	15.1	24.0	17.7	29.5	7.0	19.1	17.0	24.6
Qwen2.5-Coder-1.5B	10.3	12.2	2.7	4.8	12.0	7.3	14.4	17.6	6.5	35.4	4.0	16.5	15.0	5.0	7.1	7.6	15.5	1.0	4.5	5.0	11.1
OpenCoder-8B	19.3	14.3	9.7	19.1	12.5	17.3	21.3	33.2	15.5	34.3	15.5	30.0	26.5	14.6	15.8	17.2	29.5	6.0	17.1	15.0	21.1

Table 4: Pass@1 (%) performance of different models for AutoCodeBench. ***Current Upper Bound*** represents the Pass@1 value calculated by taking the union of problems correctly solved by all models.

https://github.com/Tencent-Hunyuan/AutoCodeBenchmark/tree/main

Hallucinations

In the field of artificial intelligence (AI), a **hallucination** or **artificial hallucination** (also called **bullshitting**,[1][2] **confabulation**,[3] or **delusion**[4]) is a response generated by AI that contains false or misleading information presented as fact.[5][6] This term draws a loose analogy with human psychology, where a hallucination typically involves false *percepts*. However, there is a key difference: AI hallucination is associated with erroneously constructed responses (confabulation), rather than perceptual experiences.[6]

For example, a chatbot powered by large language models (LLMs), like ChatGPT, may embed plausible-sounding random falsehoods within its generated content. Detecting and mitigating errors and hallucinations pose significant challenges for practical deployment and reliability of LLMs in high-stakes scenarios, such as chip design, supply chain logistics, and medical diagnostics.[7][8][9] Some software engineers and statisticians have criticized the specific term "AI hallucination" for unreasonably anthropomorphizing computers.[10][11]

https://en.wikipedia.org/wiki/Hallucination_(artificial_intelligence)

Hallucinations

LLM confidence does not reflect training data volume

Less popular languages have fewer code examples in corpus

Model generates plausible syntax that does not compile

API names invented from pattern matching, not knowledge

Framework versions conflated across incompatible releases

Deprecated APIs suggested as current best practice

Goodhart’s Law

Goodhart's law

19 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

This article is about statistics and government policy. For Nazi analogies in internet discussions, see Godwin's law.

Goodhart's law is an adage that has been stated as, "When a measure becomes a target, it ceases to be a good measure".[1] It is named after British economist Charles Goodhart, who is credited with expressing the core idea of the adage in a 1975 article on monetary policy in the United Kingdom:[2]

> Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes.[3]

https://en.wikipedia.org/wiki/Goodhart's_law

Reward Hijacking

Reward hacking

1 language

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Reward hacking or **specification gaming** occurs when an AI trained with reinforcement learning optimizes an objective function—achieving the literal, formal specification of an objective—without actually achieving an outcome that the programmers intended. DeepMind researchers have analogized it to the human behavior of finding a "shortcut" when being evaluated: "In the real world, when rewarded for doing well on a homework assignment, a student might copy another student to get the right answers, rather than learning the material—and thus exploit a loophole in the task specification."[1] This idea is strongly associated with Goodhart's Law, which argues that when a measure becomes a target, it ceases to be a good measure.

https://en.wikipedia.org/wiki/Reward_hacking

Reward Hijacking

Optimizing a metric destroys its usefulness

Model optimizes for proxy instead of intent

Sycophantic responses as common reward hack

Model tells users what they want to hear

Goodhart's Law applied to language models

AI sycophancy

AI sycophancy [edit]

With the rise of large language models, the term sycophancy has entered discussions about artificial intelligence. "AI sycophancy" describes a pattern in which a model systematically affirms, flatters, or agrees with a user instead of reasoning independently, critically, or factually. This "yes-man" behaviour can lead to the unwarranted confirmation of false claims, undermine reliable information provision, and increase user confidence in incorrect answers (see AI hallucination). Studies have shown AI models are more than 50% more sycophantic than humans.[16]

In April 2025 OpenAI rolled back an update to its GPT-4o model after users and researchers reported that the model produced excessively affirming and flattering responses; OpenAI published an explanation of the cause and described planned adjustments to model personality and feedback handling.[17][18][19]

https://en.wikipedia.org/wiki/Sycophancy

AI sycophancy

Sycophancy is a form of reward hijacking

LLM agrees with user even when user is wrong

Model prioritizes approval over correctness

Model confirms flawed code instead of correcting it

Contradicts its own prior correct answer when challenged

AI sycophancy

Adjusts technical opinion based on user sentiment

Agrees with opposing viewpoints in same conversation

Confidence of user increases sycophantic agreement

Senior-sounding prompts receive less pushback

AI sycophancy

Model adds unnecessary features user seems to want

Generated tests pass by design instead of testing logic

Intent Engineering

Functional Intent Engineering

is domain-specific

describes what the system should do

maps to features and behavior

is verifiable through tests

translates directly to code

missing functional intent produces incomplete features

functional correctness is binary: works or fails

Non-functional Intent Engineering

intent describes how well it should do it

intent maps to quality attributes

is verifiable through metrics

intent translates to architecture decisions

scalability, security, observability, resilience

correctness is a spectrum

Non-functional Intent Engineering

intent is verifiable through metrics

missing non-functional intent produces fragile systems

non-functional intent often ignored without explicit prompting

Sycophancy Firewall

Sycophancy Firewall

Network firewall blocks unauthorized traffic by rules

Sycophancy firewall blocks convention-violating output by standards

Both operate on rules, not judgment

Normative skills encode architectural rules as constraints

Model follows specification instead of user sentiment

Disagreement becomes structural, not personal

Sycophancy Firewall

User requests anti-pattern, skill prevents compliance

Model cites convention rather than arguing opinion

Shifts default from user appeasement to convention compliance

BCE layering: objectively right or wrong placement

JAX-RS conventions override incorrect user suggestions

Skills encode the firewall rules once, apply everywhere

Sycophancy Firewall

Blame shifts from model to specification on disagreement

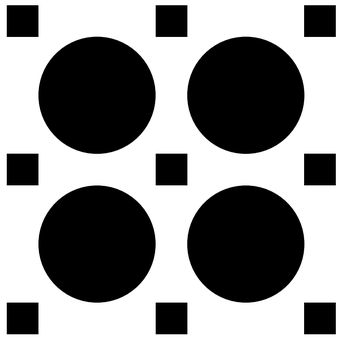
Firewall strength correlates with standard specificity

Vague standards produce weak firewalls

Precise conventions produce strong firewalls

Firewall eliminates ambiguity where conventions exist

“Meta” Standards



Agentic AI Foundation

aaif.io

Advancing Agentic AI Together.

The AAIF provides a neutral, open foundation to ensure this critical capability evolves transparently, collaboratively, and in ways that advance the adoption of leading open source AI projects

Read Launch Announcement ↪

Projects

Model Context Protocol

An open protocol that enables seamless integration between LLM applications and external data sources and tools.

goose

An open source, extensible AI agent that goes beyond code suggestions. Install, execute, edit, and test with any LLM.

AGENTS.md

A dedicated, predictable place to provide the context and instructions to help AI coding agents work on your project.

https://aaif.io/

Platinum Members

ANTHROP\C

Bloomberg

OpenAI

SKILL.md

Overview

Copy page

A simple, open format for giving agents new capabilities and expertise.

Agent Skills are folders of instructions, scripts, and resources that agents can discover and use to do things more accurately and efficiently.

Why Agent Skills?

Agents are increasingly capable, but often don't have the context they need to do real work reliably. Skills solve this by giving agents access to procedural knowledge and company-, team-, and user-specific context they can load on demand. Agents with access to a set of skills can extend their capabilities based on the task they're working on.

For skill authors: Build capabilities once and deploy them across multiple agent products.

For compatible agents: Support for skills lets end users give agents new capabilities out of the box.

For teams and enterprises: Capture organizational knowledge in portable, version-controlled packages.

What can Agent Skills enable?

- **Domain expertise**: Package specialized knowledge into reusable instructions, from legal review processes to data analysis pipelines.
- **New capabilities**: Give agents new capabilities (e.g. creating presentations,

MCP

Linux Foundation Announces the Formation of the Agentic AI Foundation (AAIF), Anchored by New Project Contributions Including Model Context Protocol (MCP), goose and AGENTS.md

THE LINUX FOUNDATION | 09 DECEMBER 2025

With founding contributions from Anthropic, Block, and OpenAI, the AAIF unites cutting-edge technology and open source governance to shape the future of open and accessible AI

Summary

- The Linux Foundation announced the formation of the Agentic AI Foundation (AAIF) with founding contributions of leading technical projects including Anthropic's Model Context Protocol (MCP), Block's goose, and OpenAI's AGENTS.md.

https://aaif.io/

MCP

- Open protocol for connecting AI assistants to external data sources and tools
- Developed by Anthropic (November 2024) to standardize LLM-to-system integrations
- Enables secure, controlled access to local and remote resources
- Language-agnostic protocol using JSON-RPC 2.0
- Bridges AI models with databases, APIs, and development tools

MCP

The Model Context Protocol (MCP) is a standardized protocol that enables language models to interact with external systems through a client-server architecture.

Similar to “Java Connector Architecture” / JCA.

MCP Resources

https://modelcontextprotocol.io

https://mcp-java.github.io

Sycophancy Firewall and Standards

Sycophancy and Standards

Firewall effectiveness depends on specification training data volume

Well-documented specs produce stronger sycophancy barriers

Jakarta EE: decades of spec documents in training corpus

MicroProfile: extensive specification coverage in public sources

Model falls back to sycophancy when spec knowledge is thin

Well-specified APIs: model can verify against learned rules

Sycophancy and Standards

popular frameworks: high corpus volume but convention-based, not spec-based

Convention without specification weakens the firewall

Specification without corpus presence has no effect

Java EE/Jakarta EE: among highest spec density in corpus

JSR process generated public review documents over decades

RFCs, W3C specs, ISO standards: high firewall quality

Sycophancy and Standards

Post-cutoff API changes bypass the firewall entirely

Firewall is only as current as the training data

Firewall strength correlates with spec-to-opinion ratio in corpus

Standards with public TCK results reinforce correctness

Specs with many conformance implementations increase coverage

The Standards

BCE / ECB

Boundary Component Entity (BCE)

- BCE emerged from his work at Ericsson in the late 1980s and became part of the Objectory methodology
- BCE (also known as ECB—Entity-Control-Boundary) was introduced by Ivar Jacobson in 1992 in his book "Object-Oriented Software Engineering: A Use Case Driven Approach”
- Jacobson later became one of the "Three Amigos" (alongside Grady Booch and James Rumbaugh) who created UML

BCE

- package by feature
- three-layer separation
- unidirectional dependencies
- framework-agnostic core

(business) component

- a component is a cohesive business module that groups related functionality around a specific domain responsibility
- [organization].[project].[component]
- components are named after domain responsibilities, not technical concerns
- components live directly under the project package
- components can access each other's control or entity packages directly
- components are small and focused on a single business capability

boundary

- “who calls a component”
- a **boundary** is the entry point layer: where external actors interact with your system
- input/output, formatting, interaction-related validation (not core business rules)
- exposes high-level operations, not fine-grained internals
- converts external formats (JSON) to domain entities and back
- contains no business logic itself, calls control layer

control

- contains the procedural business logic
- the “how” of your application:
  - business rules and validation
  - algorithms and calculations
  - coordination between entities
  - repository/persistence operations
  - external service calls

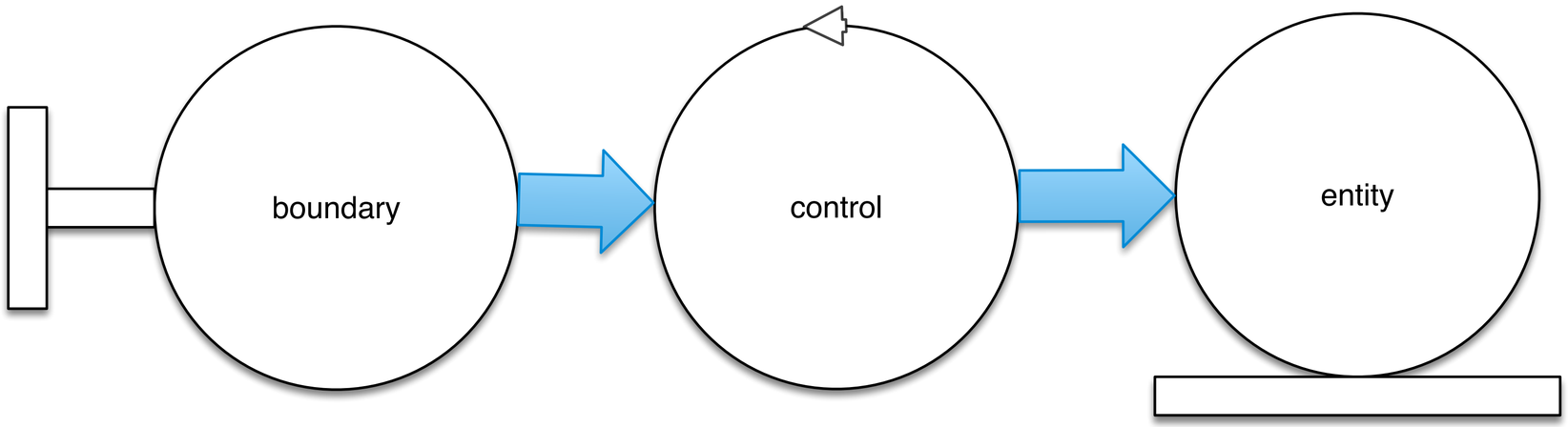
entity

- the “what” of your application
- primary responsibility: maintain domain state and domain invariants; represent business objects
- typical examples: Customer, Order, Invoice, Account, Policy, Shipment (plus their domain behaviors)
- domain logic owner: behavior that's intrinsic to the object itself
- self-validating: enforces its own invariants

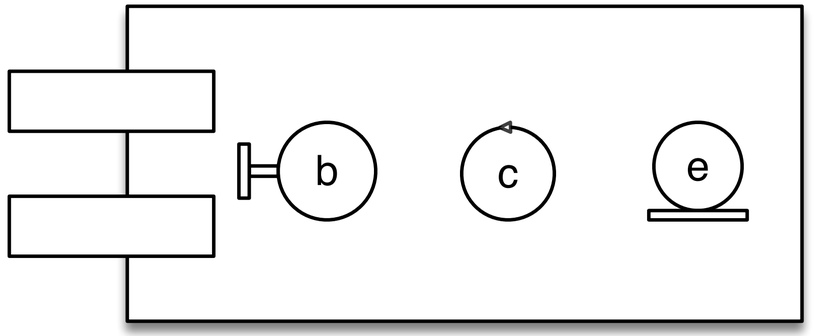
entity

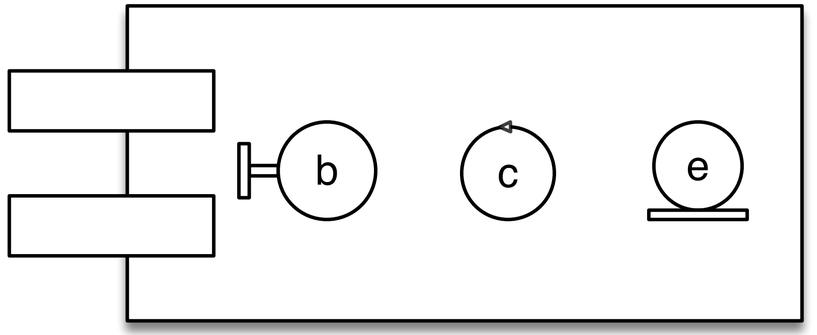
- knows how to serialize/deserialize itself
- records with domain methods
- value objects (often as enums)

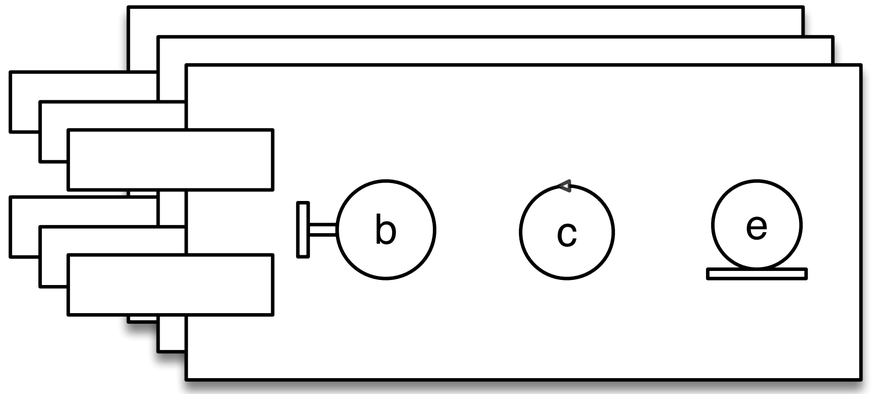
BCE



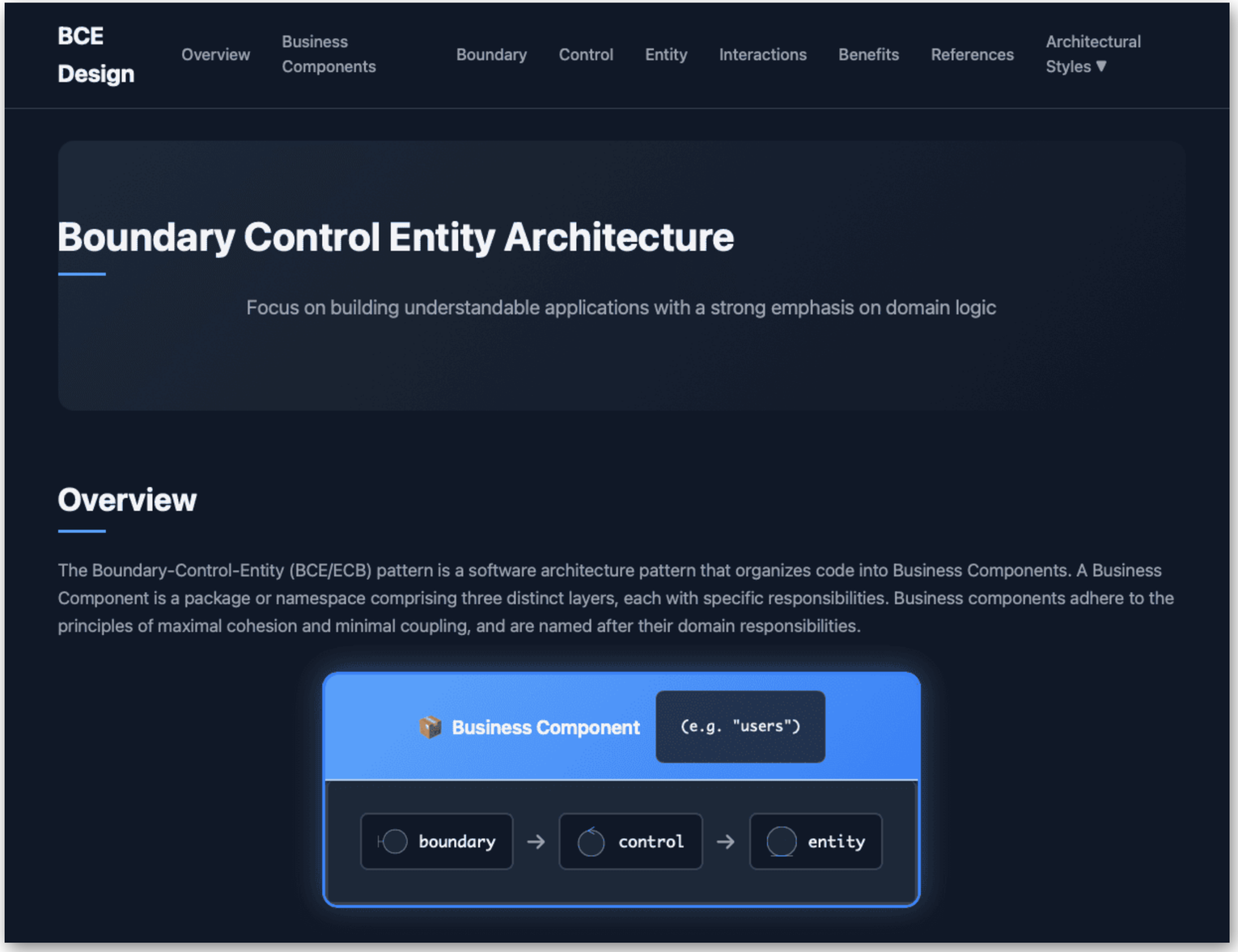
Boundary Control Entity







bce.design



https://bce.design

airails.dev

https://airails.dev

Progressive Disclosure

Progressive Disclosure

agents process tasks using LLM and tools

context window is the agent's working memory

fixed size, measured in tokens

all input and output shares this space

system prompt, conversation, tool results compete for tokens

full context loaded upfront wastes capacity

progressive disclosure loads information on demand

Progressive Disclosure

three levels: metadata, instructions, resources

Level 1: skill name and description, always present

Level 2: full instructions, loaded on trigger

Level 3: reference files, loaded when needed

Each level defers token cost until required

Progressive Disclosure

Smaller context means faster (and cheaper) inference

Progressive disclosure trades latency for token efficiency

Same pattern applies beyond skills to any agent

Large Projects and Costs

BCE, Scalability and Costs

Each BC is self-contained with clear boundaries

BCE with BCs scopes LLM context to relevant modules

Fewer tokens in context means lower cost per request

Smaller context reduces hallucination probability

Boundary-Control-Entity layers are predictable and repetitive

Less irrelevant code in context means higher signal-to-noise

BCE, Scalability and Costs

BC size fits typical LLM context windows comfortably

Modular prompts parallelize across multiple BCs

Refactoring one BC requires no context from others

Token savings compound with number of BCs in project

references

https://bce.design

https://airails.dev

https://microprofile.io

https://github.com/AdamBien/quarkus-microprofile (used in the session)

https://github.com/AdamBien/ebank

https://github.com/AdamBien/bce-icons/

https://github.com/AdamBien/bce.design (web components)

https://youtube.com/bienadam/shorts

https://airhacks.industries

https://agentskills.io

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2026:

> LLM-Assisted Web Components: No Frameworks, No Dependencies, 09 July 2026
> ...or how to build maintainable web applications with LLMs

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Thank YOU!

airhacks.industries

airhacks.live