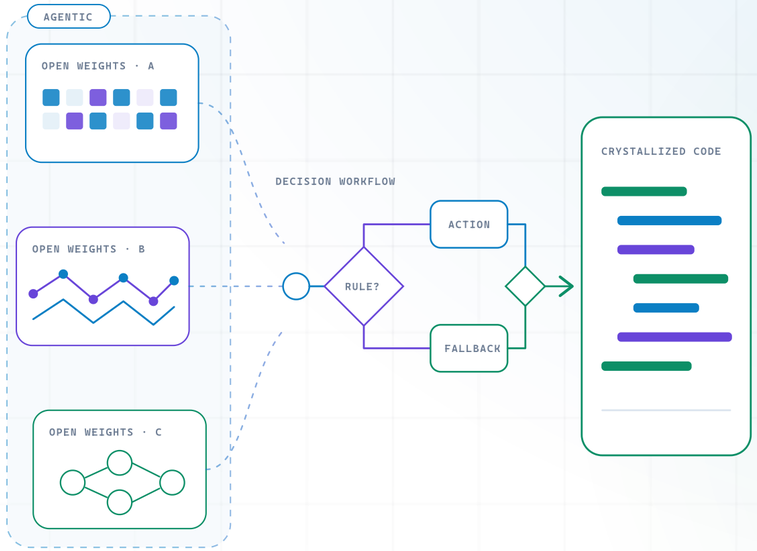
PROGRESSIVE CRYSTALLIZATION

From agent exploration to verified software.

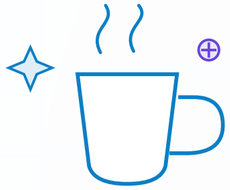
- **Arun Malik**



WHERE REASONING BELONGS

Not every coffee order needs the same kind of intelligence.

01 · NEW DRINK



How should we make it?

EXPLORE

No recipe exists. Agentic discovery and judgment are useful.

02 · AMBIGUOUS ORDER

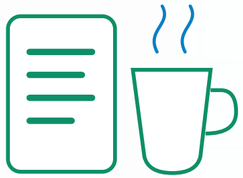


What did the customer mean?

INTERPRET

The procedure may exist, but one bounded ambiguity still requires judgment.

03 · COVERED ORDER



Which recipe applies?

ROUTE

Map the request to a supported recipe, refresh current conditions, then let software execute and verify it.

Which decisions still need to be decisions?

THE REUSABLE PATH

The order supplies parameters.
The recipe supplies the workflow.

● 01 · ORDER

Customer request

ORDER #042 — 1 ITEM

TURMERIC OAT MILK LATTE

Milk	Oat
Temperature	Extra hot
Sugar syrup	1 shot

Runtime parameters ✓

PARAMETERS CAPTURED

→

● 02 · RECIPE

Reusable workflow

TURMERIC-LATTE.RECIPE

```
milk = order.milk
temp = order.temperature
syrup = order.syrup
```

1. Pull the espresso shot
2. Steam milk with turmeric
3. Add syrup and combine
4. Check temperature and finish

RECIPE MATCHED

→

● 03 · EXECUTION

Espresso machine



Apply the recipe using current ingredients and machine state.

WORKFLOW RUNNING

→

● 04 · RESULT

Turmeric latte



Oat milk, extra hot, one shot of sugar syrup.

160°F · VERIFIED

Different parameters. The same repeatable workflow.

ONE EXECUTION IS NOT A REUSE CONTRACT

A receipt is not a recipe.

RECEIPT · ONE EXECUTION

Order	#042
Drink	Turmeric oat latte
Milk	Oat
Temperature	160°F
Sugar syrup	1 shot
Outcome	Verified

Evidence that this order worked once.



The cup is the **verified outcome.**
The documents preserve different knowledge.

RECIPE · REUSABLE CONTRACT

Turmeric oat latte

01 Check that the drink and requested modifiers are supported.

02 Bind milk, temperature, and syrup from the current order.

03 Execute the known preparation steps consistently.

04 Verify temperature and completion; route exceptions for investigation.

A reviewed contract for the next covered order.

One successful execution is evidence, not permission to repeat it everywhere.

WHAT THE RECIPE BECOMES

The reusable recipe becomes executable, inspectable software.

01
Check coverage
Use the recipe only when it supports the requested drink and modifiers.

02
Bind the parameters
Map oat milk, extra-hot temperature, and syrup into the reusable workflow.

03
Execute and verify
Run the machine, check the result, and recover explicitly when verification fails.

TURMERIC-LATTE.TS

EXPLICIT · INSPECTABLE · TESTABLE

```
function makeTurmericLatte(order, machine) {
  if (!recipe.supports(order))
    return agent.investigate(order);

  const parameters = recipe.bind(order);
  const result = machine.execute(
    recipe.steps, parameters);

  return recipe.verify(result)
    ? result
    : recipe.recover(result);
}
```

Coverage · parameters · execution · verification

The order changes the parameters, not the execution contract.

FROM ANALOGY TO PRECEDENT

The coffee is the analogy. The engineering move is not new.

SLIDE 5 · COFFEE ANALOGY

A recipe becomes inspectable software



Reviewed recipe

→



Maintained code

Parameters vary. The reviewed execution contract persists.

THE METAPHOR

SHARED ENGINEERING PATTERN

Capture knowledge
RECIPE · RUNBOOK

↓

Engineer the stable path
REVIEW · SAFEGUARDS

↓

Maintain execution
CODE · AUTOMATION

HOW ENGINEERING WORKS TODAY

Operational guidance becomes maintained automation



TSGs · runbooks · playbooks

Operational guidance

→



Engineers encode the stable path

Safeguards and integration

→



Scripts · workflows · automation

Maintained with the software

People execute the guidance first. Engineers encode the repeatable path with checks, recovery, and integration.

A FAMILIAR ENGINEERING PATH

Progressive Crystallization extends a familiar engineering practice.

TRADITIONAL AUTOMATION → EXECUTABLE PLAYBOOKS

The alert is the same. The operating model changes.

Alert or task

SAME INPUT

TRADITIONAL

Automation plus documentation

Gaps leave the executable path.

Scripts and workflows → Unsupported gap → Human investigates → TSG or runbook updated

NEW · HIGHLIGHTED PATH

Executable playbook

Automation, agents, and human judgment share one contract.

How much is known?

Select the playbook type that matches the uncertainty.

DETERMINISTIC PLAYBOOK	Fully known and repeatable	Scripts and workflows. No agent needed.
HYBRID PLAYBOOK	One ambiguous decision does not make the whole workflow agentic	Collect evidence deterministically → agent makes one bounded decision.
AGENT-LED PLAYBOOK	Goal known, path ambiguous	Agent plans and uses tools with human guardrails.

Verified outcome

Or human escalation when authority or confidence is insufficient.

DESIGN TIME · SEPARATE FROM EXECUTION

From manual TSGs to executable assets

Engineers can encode directly; agents can reduce the authoring cost.

Manual TSG and execution evidence → Engineer writes directly or agent drafts code and tests → Engineer reviews and validates → Executable playbook or reusable skill ships

Agents handle bounded gaps today. Engineers remove repeatable gaps tomorrow.

REASONING IS NOT PERMISSION

A recommendation cannot grant its own authority.

AGENT REASONING

Proposed action

Evidence, rationale, intended target, requested operation, and expected consequence.

Capability

→

GRANULAR AUTHORIZATION CONTRACT

Every consequence crosses explicit gates

01	02	03	04	05
Identity	**Tool**	**Function**	**Parameter**	**Approval**
Who or what is acting?	Which tools are available?	Read, write, or destructive?	Which targets and values?	What requires a person?

`Actor → reasoning decision → tool call → service response → verified outcome`

→

PERMITTED CONSEQUENCE

Execute, audit, verify

The authorized operation runs within scope. Its result is independently checked and attributable.

Authority

Generated commands, configurations, or high-impact actions require stronger validation and explicit approval.

Capability is not access. Interpretation is not authorization.

TWO LINKED LOOPS

Recover the system now. Improve the procedure deliberately.

RUNTIME · OPERATIONAL RECOVERY

Handle this case

Software owns state, deadlines, retries, durable waits, and stopping conditions.

Observe — Collect fresh, validated evidence → **Decide** — Apply the known rule or bounded judgment → **Act** — Cross the authorization boundary → **Verify** — Recover, retry, roll back, or escalate

Runtime evidence and exceptions ↓

↑ Reviewed contract and routing update

DESIGN TIME · ENGINEERING LEARNING

Improve the next case

Human actions are evidence, not unquestionable labels. Review the outcome and context.

Preserve evidence — Inputs, decisions, tools, failures, unknowns → **Draft contract** — Scope, typed behavior, exceptions, owner → **Test and review** — Replay, counterexamples, dependencies → **Promote** — Release only the covered variant

No automatic promotion: unsupported, unsafe, or unverified cases remain visible.

Runtime handles the incident. Engineering changes the maintained system.

MEASURE THE WHOLE OPERATING SYSTEM

Not just the model call. Not just the successful path.

TOTAL-COST LEDGER

Zero inference is not zero cost

Judgment | Tools | People | Infrastructure | Maintenance | Exceptions

Pay for judgment where it changes the result; account for the complete path.

SCALE MULTIPLIER

Small recurring decisions compound

One repeated decision × Operational recurrence = Cumulative effort and risk

Scale motivates engineering discipline; it does not prove causal savings.

FULL DENOMINATOR

Keep every eligible outcome visible

Eligible cases →

- Verified outcome
- Human escalation
- Unsupported
- Unknown or unverified

Do not report only accepted cases or relabel every remainder as failure.

EXECUTION PORTFOLIO

Different scenarios keep different forms

- Agent-led path uncertain
- Hybrid judgment bounded
- Deterministic decisions specified

No target percentages. No mandatory maturity ladder. Compare like-for-like scenarios.

Same verified outcome contract — Coverage · recovery · cleanup · escalation · total effort · maintained correctness

Optimize verified recovery per unit of risk and total effort, not automation purity.

CLOSING SYNTHESIS

Give AI the unknown. Give software the understood.



UNKNOWN

Investigate

A new failure or unfamiliar request needs evidence gathering, hypothesis testing, and visible uncertainty.

→



DEVELOPING UNDERSTANDING

Capture and bound

A hybrid playbook keeps deterministic evidence collection while an agent interprets only the ambiguous decision.

→



UNDERSTOOD

Execute and verify

A reviewed procedure runs as maintained software and returns to investigation when its assumptions stop holding.

OBSERVE → INVESTIGATE → CAPTURE → REVIEW → PROMOTE → VERIFY → LEARN

Start tomorrow: choose one recurring question; define its evidence, authority, exceptions, and verified outcome.

Every stage keeps a visible path to human review and escalation

Progressive Crystallization places judgment where it belongs.



Questions?

Questions, counterexamples, and disagreements are welcome.

Merci
ありがとう
Dziękuję

Gracias
شكرا
Děkuji

Danke
감사합니다
Köszönöm

Grazie
Teşekkürler
Hvala

Obrigado
Dank je
Ďakujem

Спасибо
Tack
Mahalo

谢谢
Kiitos
Asante



Thank You

ધન્યવાદ
આભાર
Terima kasih

Salamat
شكريہ
Mulțumesc

ধন্যবাদ
Дякую
متشكرم

ਧੰਨਵਾਦ
Ευχαριστώ
መሰግናለሁ

நன்றி
הודה
ഉടൻ

ಧನ್ಯವಾದಾಲು
ขอบคุณ
Tak

ಧನ್ಯವಾದಗಳು
Cảm ơn
Takk