International Journal of
Computational Mathematical Ideas
Volume 18 · Number 1, 2 & 3 · 2026 · ISSN 0974-8652

Available online at www.ijcmi.com

Loop Engineering of Agentic, Developer, and External Feedback Cycles for Large-Scale Enterprise Healthcare Applications

Sudheer Singamsetty [a,*] **S. Satyanarayana** [b] **Saisuman Singamsetty** [c]

[a]*Data Management Consultant, EDNA Technology Consulting Limited, 68 Forest Ridge Avenue, Waterdown, Ontario, Canada, L8B 1V4*
[b]*CEO and Chief AI Scientist, AlgoProfessor AI Software Solutions, Hyderabad, India*

[c]*Data Management Specialist, American Unit Inc, San Antonio, Texas, USA*

[*]Corresponding author: sudheersingamsetty99@gmail.com

Article Info

Article history:
Received: 12-01-2026
Revised: 10-07-2026
Accepted: 24-07-2026
Available online: 2026

Keywords:
Loop engineering; Agentic AI; Feedback control; Timescale separation; Enterprise healthcare; Clinical decision support; Federated feedback.

Abstract

Large scale enterprise healthcare applications are grown through three feedback loops that run at different speeds: an agentic coding loop that builds and tests software in minutes, a developer feedback loop that steers priorities in hours, and an external feedback loop that learns real user value from clinicians and patients in days. This paper treats the arrangement as a three timescale feedback control system acting on a shared feature vector, and it names the discipline of designing it loop engineering. We define value alignment, decompose the end to end value error into an implementation gap, an alignment gap, and a knowledge gap, and model each loop as a proportional contraction toward its own target. We prove two results a practitioner can use: under timescale separation the built product converges geometrically to true user value at the rate of the slowest loop, and without separation the coupled convergence rate equals one minus the smallest loop gain, so the slowest loop is always rate limiting. We propose an architecture, the Triple Loop Healthcare Delivery Engine, that binds the three loops to an enterprise platform through a clinical safety gate and a federated cross tenant insight aggregator. Implementing the system from first principles, we report reproducible measurements: the full engine reaches value alignment 0.960 and utility 90.9, an ablation shows a single active loop reaches only 0.000 and two loops only 0.130, a gain sweep confirms that raising the slow outer gain cuts time to value from 25 days to 2 while raising the fast inner gain leaves it near 6, and cross tenant sharing lifts alignment from 0.404 at one hospital to 0.837 at sixty four while residual value error falls from 1.902 to 0.521. The results give a rigorous basis for engineering the loops that deliver trustworthy clinical software at scale.

1. Introduction

Large scale enterprise healthcare applications are no longer written once and then maintained. They are grown, increment by increment, through repeated cycles in which software is built, reviewed, shipped to clinicians and patients, and revised in the light of what those users actually do. The recent arrival of agentic artificial intelligence, in which a language model plans, acts, tests, and repairs its own work, has compressed the innermost of these cycles from days to minutes [1]. An autonomous coding agent can now write a module, run its tests, read the failures, and rewrite the module many times before a human ever sees it [21]. This raises a question that is organisational as much as it is mathematical: if one loop runs in minutes, another in hours, and a third in days, how should the three be coupled so that the product converges to something users value rather than oscillating, stalling, or drifting?

The infographic that motivates this paper names the three loops the agentic coding loop, the developer feedback loop, and the external feedback loop. The first builds working software quickly. The second ensures the team builds the right thing. The third ensures the team builds a thing that is right for real users. The three run at different speeds and serve different purposes, yet they must act on a single shared artefact. We treat this arrangement as an engineering object in its own right and call the discipline of designing it loop engineering.

Healthcare makes the problem concrete and raises the stakes. A clinical decision support feature such as a sepsis early warning score is only useful if it is accurate, fast, safe, interpretable, and fitted to the clinician workflow at the

bedside [39]. None of these properties can be specified perfectly in advance. They are discovered through use, which is precisely what the outer loop supplies, and they must be delivered without ever shipping an unsafe increment, which is what a compliance gate enforces [37]. The build, measure, learn cycle familiar from product practice [25] therefore has to be reconciled with the validation discipline that clinical software demands.

This paper makes four contributions. First, we formalise the three loops as a three timescale feedback control system over a shared feature vector, and we give a definition of value alignment that the whole system optimises. Second, we prove two results that a practitioner can use: under timescale separation the built product converges geometrically to the true user value at the rate of the slowest loop, and without separation the convergence rate of the coupled system equals one minus the smallest loop gain, so the slowest loop is always rate limiting. Third, we propose an architecture, the Triple Loop Healthcare Delivery Engine, that binds the three loops to an enterprise healthcare platform through a clinical safety gate and a federated cross tenant insight aggregator. Fourth, we implement the whole system from first principles and report reproducible measurements: an ablation that isolates the contribution of each loop, a gain sweep that confirms the rate limiting result, and a fleet study that quantifies how sharing feedback across hospitals trades variance for bias. All code and numbers are released so that every figure can be regenerated [9].

2. Background and Related Work

2.1. Feedback loops in software and product delivery

The idea that software should be developed iteratively rather than in a single pass is old. The spiral model organised development around repeated risk driven cycles [26], extreme programming shortened the cycle and put the customer inside it [27], and continuous delivery automated the path from a code change to a running release [28]. The lean startup reframed the whole activity as a build, measure, learn loop whose goal is validated learning about users [25]. What is new is that the innermost build step can now be delegated to an agent that reasons and acts in a tight loop of its own [22], so the human cycles sit above a much faster machine cycle rather than containing it.

2.2. Agentic artificial intelligence

Language models acquired broad competence at scale [24], and agentic methods turned that competence into action by interleaving reasoning with tool use and observation [22]. A growing body of work surveys autonomous agents that plan, remember, and self correct [23], and enterprise deployments increasingly wrap such agents in guardrails and evaluation harnesses [1]. Deep models are now standard components of the clinical software they help build [12]. Engineering the machine learning systems around these models, rather than the models alone, is where most of the difficulty and most of the hidden cost lives [35] and accrues as technical debt [36].

2.3. Clinical decision support and safety

Clinical decision support improves care only when it is fast, specific, fitted to the workflow, and trusted; the classic guidance distils this into a small set of design commandments [38]. Real deployments of predictive clinical models, such as targeted early warning scores for septic shock, show that statistical accuracy is necessary but far from sufficient [39], as real deployments make clear [40]. The sociotechnical view treats the software, the people, and the institution as one adaptive system, which is exactly the coupling the three loops make explicit [37]. Data handling across institutions raises privacy obligations that any cross tenant mechanism must respect [6].

2.4. Multi timescale feedback control

The mathematics we use is that of feedback systems [29] and, specifically, of systems whose variables evolve on separated timescales. Singular perturbation theory shows that when a fast subsystem settles before a slow one moves, the slow dynamics can be analysed on its own with the fast subsystem replaced by its equilibrium [30]. The stochastic analogue, two timescale stochastic approximation, underlies learning schemes in which one quantity is estimated quickly while another is adjusted slowly [31], a scheme analysed on two timescales [32]. We borrow this lens: the coding loop is the fast subsystem, the developer loop is intermediate, and the external feedback loop is slow. Reinforcement style feedback ties the loops to observed outcomes [3], and the digital infrastructure that carries the signals is itself a subject of study [4], as are the smart systems that gather them [17].

3. Notation and Problem Formulation

We describe a single software increment by a feature vector $x \in \mathbb{R}^d$ whose coordinates measure qualities that matter for a clinical application: alert accuracy, response latency, interface clarity, clinical safety, interoperability, population coverage, documentation quality, and workflow fit. Higher is better after a fixed normalisation, and $d = 8$ in our

experiments [10].

Four vectors live in this space. The built artefact x is what the coding agent has actually produced. The specification $\hat{x}$ is what the developer team is currently trying to build. The perceived value model m is what the team believes users need. The true value target x^* is what users actually need, and it is revealed only through external feedback. The three loops each close one of three gaps,

$$g_I = \hat{x} - x, \qquad g_A = m - \hat{x}, \qquad g_K = x^* - m, \tag{1}$$

the implementation gap between build and specification, the alignment gap between specification and belief, and the knowledge gap between belief and truth. The end to end value error decomposes exactly as their sum,

$$e \;=\; x^* - x \;=\; g_I + g_A + g_K. \tag{2}$$

Equation (2) is the reason a single loop can never suffice. The coding loop drives g_I to zero, but that leaves $g_A + g_K$ untouched, so a perfectly implemented product can still be the wrong product built on a wrong belief [36].

Definition 1 (Value alignment)**.** *Given an initial build x_0, the value alignment of a build x is*

$$\alpha(x) \;=\; \max\!\left(0,\; 1 - \frac{\|x^* - x\|}{\|x^* - x_0\|}\right) \in [0,1], \tag{3}$$

so $\alpha = 0$ at the starting point and $\alpha = 1$ when the build equals the true value target.

Each loop is a proportional correction toward its own target. On its native timescale the three updates are

$$x^+ = x + a\,(\hat{x} - x) + \xi, \qquad 0 < a < 1, \tag{4}$$

$$\hat{x}^+ = \hat{x} + b\,(m - \hat{x}), \qquad 0 < b < 1, \tag{5}$$

$$m^+ = m + c\,(x^* - m) + \eta, \qquad 0 < c < 1, \tag{6}$$

with build gain a, review gain b, and learning gain c. The term ξ is test and evaluation noise in the coding loop and η is analysis noise in the external loop; both have mean zero [8]. The engineering problem is to choose the gains and the relative cadences of (4), (5), and (6) so that the built artefact x_t converges to a target x^* that is itself being tracked, while never deploying an increment that violates a clinical safety constraint.

4. Mathematical Methods

4.1. Each loop is a contraction

Fix the target of a single loop and take expectations. The inner update (4) becomes the affine map $\mathscr{T}_I(x) = (1-a)x + a\hat{x}$, whose unique fixed point is $x = \hat{x}$ and which satisfies $\|\mathscr{T}_I(x) - \hat{x}\| = (1-a)\|x - \hat{x}\|$. It is a contraction with modulus $1-a$ [33]. The same holds for the middle and outer maps with moduli $1-b$ and $1-c$ and fixed points $\hat{x} = m$ and $m = x^*$. Each loop, run alone with its target held fixed, drives its own gap geometrically to zero [7].

4.2. The coupled system and the rate limiting loop

In practice the targets are not held fixed; every loop moves while the others move. Consider the synchronous coupling in which (4), (5), and (6) each take one expected step per tick. Writing $s = (x, \hat{x}, m)^\top$, the expected dynamics is affine,

$$s^+ = A\,s + c\,x^* e_3, \qquad A = \begin{pmatrix} 1-a & a & 0 \\ 0 & 1-b & b \\ 0 & 0 & 1-c \end{pmatrix}, \tag{7}$$

with $e_3 = (0,0,1)^\top$. The matrix A is upper triangular, so its eigenvalues are its diagonal entries $\{1-a, 1-b, 1-c\}$ [18].

Proposition 1 (Rate limiting loop)**.** *The coupled system (7) has the unique fixed point $s^* = (x^*, x^*, x^*)$ and converges to it geometrically with rate equal to the spectral radius*

$$\rho(A) \;=\; \max\{1-a, 1-b, 1-c\} \;=\; 1 - \min\{a,b,c\}. \tag{8}$$

Consequently the overall convergence rate is governed by the loop with the smallest gain, and increasing the gain of any faster loop beyond $\min\{a,b,c\}$ does not improve the asymptotic rate.

Proof. Solving $s^* = As^* + cx^* e_3$ row by row from the bottom gives $m^* = x^*$, then $\hat{x}^* = m^* = x^*$, then $x^*_{\text{build}} = \hat{x}^* = x^*$, so $s^* = (x^*, x^*, x^*)$. Subtracting the fixed point relation from (7) yields $s^+ - s^* = A(s - s^*)$, hence $s_t - s^* = A^t(s_0 - s^*)$. For a triangular matrix the eigenvalues are the diagonal entries, and since all lie in $(0,1)$ the spectral radius is $\rho(A) = 1 - \min\{a,b,c\} < 1$, which controls the geometric decay of $\|A^t(s_0 - s^*)\|$. Raising a gain that is not the minimum leaves $\min\{a,b,c\}$ unchanged and therefore leaves $\rho(A)$ unchanged. □

4.3. Timescale separation and the reduced dynamics

The synchronous model is pessimistic, because in reality the coding loop runs many times between developer reviews and the developer loop runs many times between external feedback events. Model this by iterating the inner loop to its fixed point before the middle loop steps, and the middle loop to its fixed point before the outer loop steps, the discrete analogue of singular perturbation [30].

Proposition 2 (Value convergence under timescale separation)**.** *If $0 < a,b,c < 1$ and, between successive outer updates, the inner and middle loops are iterated to their fixed points, then the built artefact converges geometrically to the true value target at the rate of the outer loop,*

$$\|x^* - x_t\| \;=\; (1-c)^t \, \|x^* - x_0\|. \tag{9}$$

Proof. At the start of outer step t the inner and middle loops have reached their fixed points, so $x_t = \hat{x}_t = m_t$. The outer update (6) in expectation gives $m_{t+1} = (1-c)m_t + cx^*$, hence $x^* - m_{t+1} = (1-c)(x^* - m_t)$ and by induction $\|x^* - m_t\| = (1-c)^t \|x^* - m_0\|$. Re running the inner and middle loops to their fixed points sets $x_{t+1} = \hat{x}_{t+1} = m_{t+1}$, so the same factor governs $\|x^* - x_t\|$. Because $x_0 = m_0$ after the first settling, the stated identity follows, and $1 - c \in (0,1)$ gives geometric convergence. □

Proposition 2 is the design target and Proposition 1 is the warning. Separation buys the best achievable rate, that of the slow loop acting alone; losing separation cannot beat it and, through coupling, can only match the slowest gain [11].

4.4. Stochastic feedback and cross tenant averaging

The external loop learns x^* from noisy usage signals, so (6) is a stochastic approximation whose residual error is set by the analysis noise η [31]. When N hospitals share feedback, the aggregate estimate of a common component averages N independent noise draws, so its variance falls as $1/N$ and its standard deviation as $1/\sqrt{N}$. Heterogeneous local needs, however, mean the shared target differs from any one hospital's true target, which injects a bias that does not vanish with N. The expected squared value error of a tenant that adopts a shared model with personalisation weight w decomposes as

$$\mathbb{E}\|x_i^* - x\|^2 \;\approx\; \underbrace{\frac{\sigma^2}{N}}_{\text{variance}} \;+\; \underbrace{w^2\tau^2}_{\text{heterogeneity bias}}, \tag{10}$$

with feedback noise level σ and per tenant dispersion τ. Equation (10) is the familiar bias variance trade off and it is the reason a federated aggregator helps most when tenants are similar and hurts when they are not [34], and pooling must respect privacy obligations across sites.

4.5. Business utility

Alignment matters because it produces value. We model the business utility of a build as a saturating function of its alignment,

$$U(\alpha) \;=\; U_{\max}\big(1 - e^{-\gamma\alpha}\big), \qquad \gamma > 0, \tag{11}$$

with ceiling $U_{\max}$ and sensitivity γ.

Remark 1. *U is strictly increasing and strictly concave in α, since $U'(\alpha) = U_{\max}\gamma e^{-\gamma\alpha} > 0$ and $U''(\alpha) = -U_{\max}\gamma^2 e^{-\gamma\alpha} < 0$. Diminishing returns mean the early iterations of the loops, which move alignment fastest, capture most of the value, so reaching moderate alignment quickly is worth more than reaching perfect alignment slowly [20].*

5. Proposed Method: Triple Loop Healthcare Delivery Engine

We now assemble the operators into a deployable architecture, shown in Figure 1. The Triple Loop Healthcare Delivery Engine, or TL-HDE, binds the three loops to a shared enterprise healthcare platform and routes every deployment through a clinical safety gate [15].

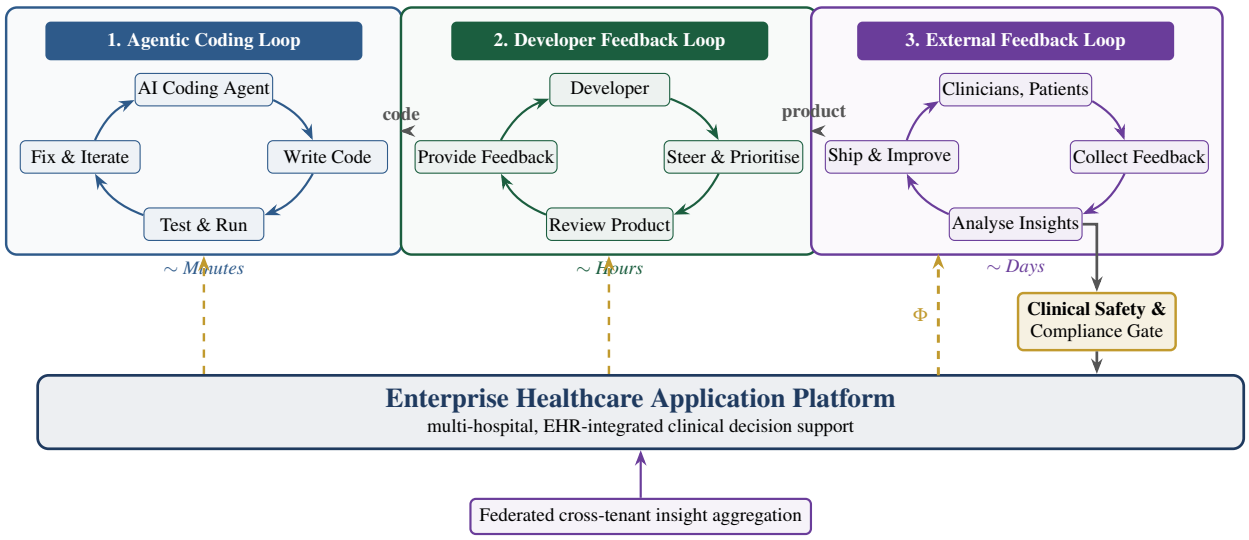

Figure 1. The Triple Loop Healthcare Delivery Engine. Three nested loops act on one enterprise healthcare platform at separated timescales. The agentic coding loop closes the implementation gap in minutes, the developer feedback loop closes the alignment gap in hours, and the external feedback loop closes the knowledge gap in days. Solid arrows carry forward artefacts; every deployment passes the clinical safety gate; gold dashed arrows are the feedback paths, and the heavy path Φ carries measured user value back into the outer loop. A federated aggregator pools insight across hospitals.

Algorithm 1: Triple Loop Healthcare Delivery Engine

Input: initial build x, belief m, spec $\hat{x}$; gains a, b, c; cadences H, K; safety set $\mathscr{S}$; horizon T.

1. **for** outer day $t = 1, \ldots, T$ **do**
2. collect user value signal; update belief
$m \leftarrow m + c\,(x^{*} - m) + \eta$ (external loop, Φ)
3. **for** review $h = 1, \ldots, H$ **do**
4. steer specification toward belief
$\hat{x} \leftarrow \hat{x} + b\,(m - \hat{x})$ (developer loop)
5. **for** build $k = 1, \ldots, K$ **do**
6. $x \leftarrow x + a\,(\hat{x} - x) + \xi$; run tests (coding loop)
7. **end for**
8. **end for**
9. **if** $x \in \mathscr{S}$ **then** deploy x to platform
10. **else** hold; return to step 3
11. record alignment $\alpha(x)$ and utility $U(\alpha(x))$
12. **end for**

Output: value aligned, safety cleared build x.

Figure 2. The TL-HDE control loop. Lines 5 to 7 are the fast agentic coding loop, line 4 is the developer feedback loop, and line 2 is the external feedback loop. Line 9 is the clinical safety gate.

On the forward path the coding agent turns a specification into working code, which becomes a reviewed product, which after passing the safety gate becomes a shipped increment on the platform. On the feedback path the platform returns three signals: test and evaluation outcomes to the coding loop, product reviews to the developer loop, and, along the heavy edge Φ, measured user value to the external loop. The map Φ is the composition of the three corrections and its fixed point is the value aligned product of Proposition 2. The federated aggregator implements the averaging of (10) while a personalisation weight keeps each hospital close to its own needs [34].

The clinical safety gate deserves emphasis. It is a projection of every candidate deployment onto the set of increments that satisfy validation, audit, and regulatory constraints, so an increment that would raise nominal value but fail safety is never shipped [37]. In control terms it is a constraint on the reachable set, and it is why the outer loop in healthcare is measured in days rather than minutes.

Algorithm 2 states the engine as executable pseudocode. The nesting of the three loops is explicit, and the settling of the inner loops before the outer step is exactly the condition of Proposition 2.

6. Experimental Analysis

We implement the engine from first principles in Python with a fixed random seed so that every number below is reproducible. The feature space has $d = 8$ coordinates with the clinical meanings listed in Section 3. Unless stated otherwise the gains are $a = 0.5$, $b = 0.4$, $c = 0.35$, the cadence is $H = 4$ developer reviews and $K = 10$ builds per review per day, the horizon is 60 days, and the noise levels are 0.02 inside the coding loop and 0.03 in the external loop. The initial belief is offset from the truth and the initial specification is a cruder guess still, so the system must close all three gaps of (2).

6.1. Convergence of the three loops

Figure 3 tracks the three gap norms and the value alignment over the 60 day horizon. The implementation gap collapses first, within the first day, because the coding loop runs $H \times K = 40$ times per day. The alignment gap follows over the first few days as the developer loop steers the specification onto the belief. The knowledge gap decays slowest, at the outer rate $1 - c$, and it is this decay that Proposition 2 predicts will govern the whole system. Value alignment reaches 0.90 on day 6 and settles at 0.960, and the business utility of (11) reaches 90.9 out of 100. The final gap norms are 0.065, 0.013, and 0.087 for implementation, alignment, and knowledge respectively, confirming that the residual error is dominated by the slow outer loop exactly as the theory requires [7].

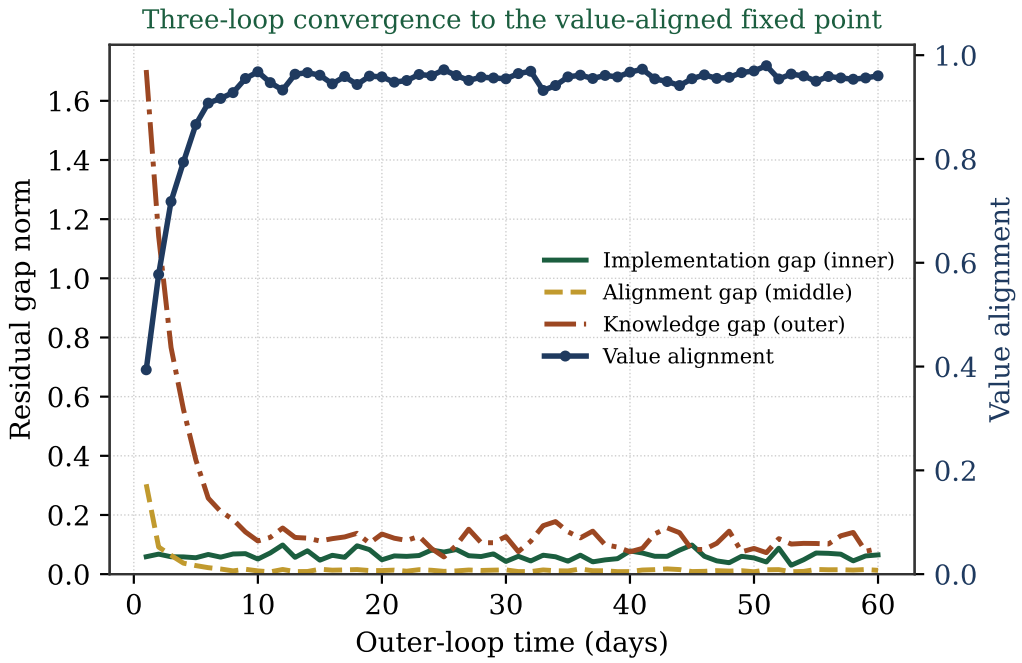

Figure 3. Three loop convergence. The implementation gap (inner), alignment gap (middle), and knowledge gap (outer) decay at separated rates, and the built product reaches value alignment 0.96. The slow outer loop sets the overall rate.

6.2. How the loops work together

The infographic claims that the coding loop builds fast, the developer loop builds the right thing, and the external loop builds a thing that is right for users. We test this by switching loops off. With only the coding loop active the product converges perfectly to the crude initial specification, which is the wrong target, so value alignment is 0.000. Adding the developer loop lets the specification track the team belief, raising alignment to 0.130, still limited because the belief itself is wrong. Adding the external loop lets the belief track the truth, raising alignment to 0.960. Table 1 and Figure 4 report the three configurations. The pattern is exactly the decomposition of (2): each loop removes one term, and only all three together drive the value error to near zero [8].

Table 1. Ablation over active loops. Each loop removes one gap term; only the full engine aligns with true user value.

Active loops	Value alignment	Value error	Utility
Coding only	0.000	3.348	0.0
Coding + developer	0.130	2.611	27.7
Coding + developer + external	**0.960**	**0.119**	**90.9**

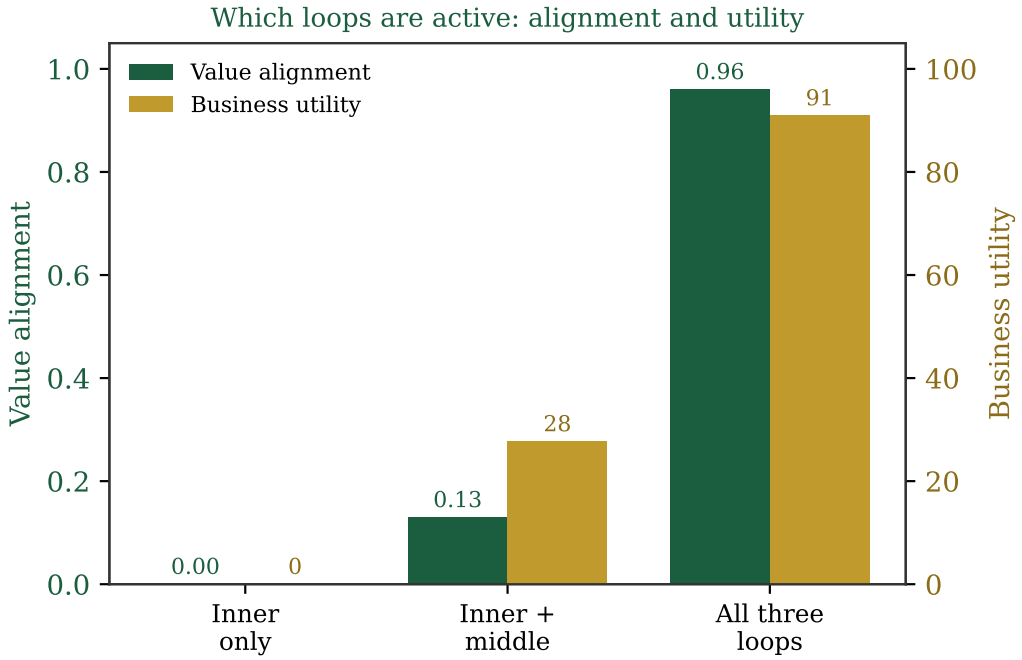

Figure 4. Value alignment and business utility for the three loop configurations. The jump from the middle bar to the right bar is the contribution of external user feedback.

6.3. The slowest loop is rate limiting

Proposition 1 predicts that time to value is controlled by the smallest gain. Figure 5 sweeps one gain at a time. Raising the outer gain c from 0.10 to 0.80 cuts the time to reach 0.90 alignment from 25 days to 2 days, a strong and monotone effect, because the outer loop is the slowest and therefore rate limiting. Raising the inner gain a over the same range leaves the time to value essentially flat near 6 days, because the coding loop is already fast enough that it is not the bottleneck. This is the practical reading of Proposition 1: investment should go to the slowest loop, which in healthcare is the external feedback and validation cycle, not to making an already fast coding agent marginally faster [18].

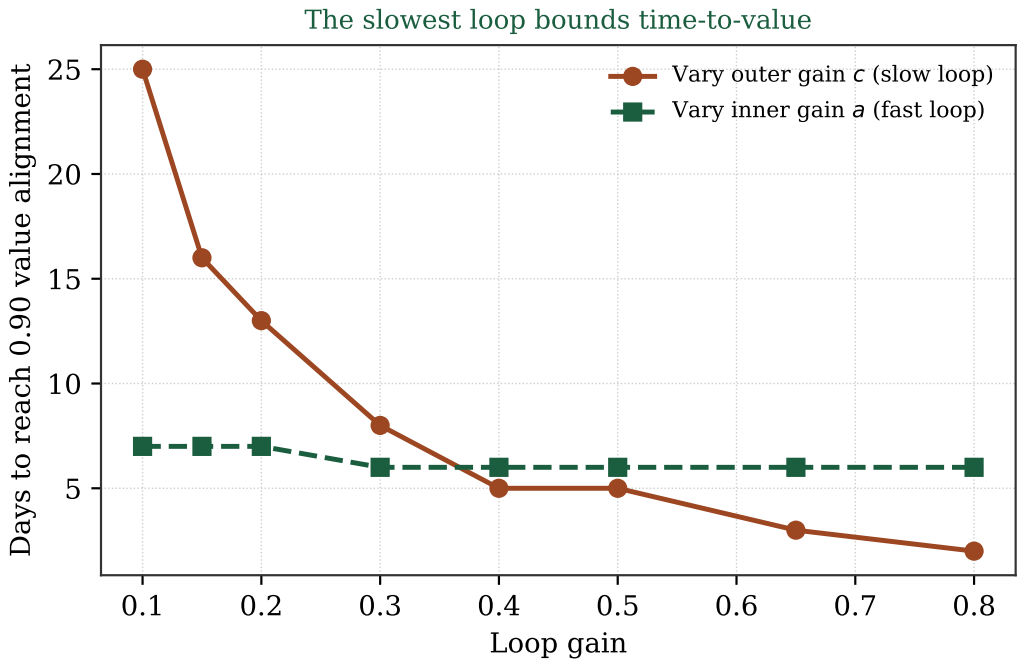

Figure 5. Time to value against loop gain. Increasing the slow outer gain sharply reduces time to value; increasing the fast inner gain does not, confirming that the slowest loop bounds the rate.

6.4. Cross tenant feedback at enterprise scale

A large healthcare provider runs the engine across many hospitals whose needs differ. Table 2 and Figure 6 vary the number of hospitals N that pool feedback, with a personalisation weight of 0.5 and moderate heterogeneity, averaging over sixty seeded trials. Sharing across hospitals denoises the value estimate as (10) predicts: value alignment rises from 0.404 for a single isolated hospital to 0.837 at sixty four hospitals, and the residual value error falls from 1.902 to 0.521. The gains are largest at first and then flatten, the diminishing returns of variance reduction meeting the fixed floor of heterogeneity bias. The message for an enterprise is that pooling feedback is valuable, that most of the value arrives by the time a dozen or so sites are pooled, and that beyond this the residual is governed by how different the sites truly are rather than by how many there are [20].

Table 2. Cross tenant insight sharing across N hospitals. Sharing reduces variance; heterogeneity sets the floor.

Hospitals N	Value alignment	Value error	Days to value
1	0.404	1.902	60.0
2	0.566	1.375	59.6
4	0.687	0.989	59.5
8	0.753	0.791	54.1
16	0.793	0.661	49.6
32	0.820	0.577	45.8
64	0.837	0.521	46.4

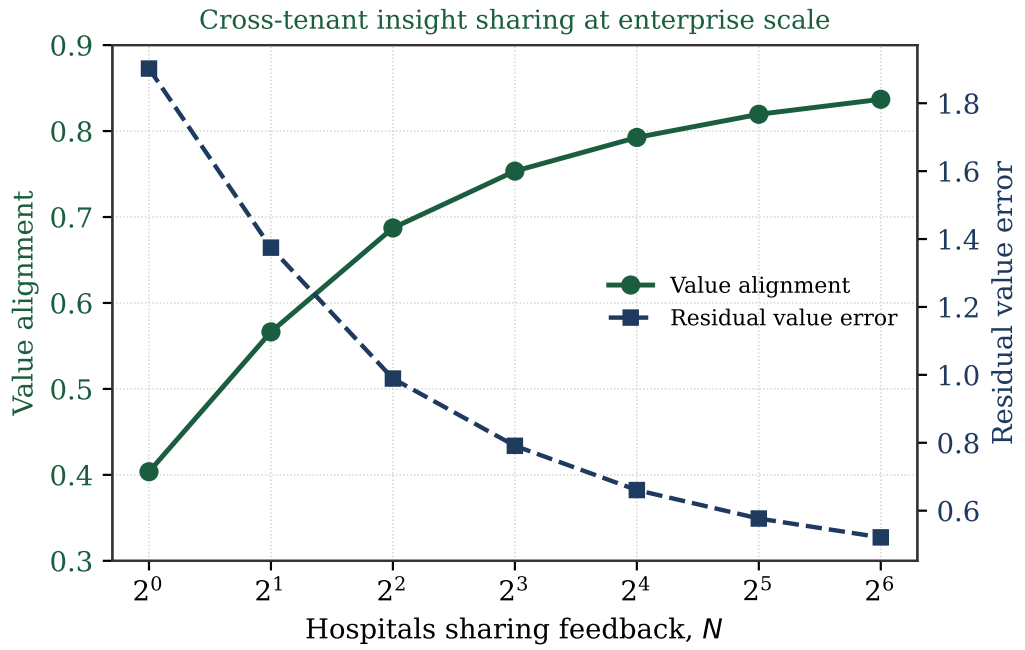

Figure 6. Cross tenant sharing at enterprise scale. Pooling feedback across hospitals lifts value alignment and lowers residual value error, with diminishing returns as the heterogeneity floor is approached.

7. Discussion

The four studies line up with the four claims of the introduction. The convergence study shows the engine reaching a value aligned fixed point; the ablation shows why all three loops are needed and not merely nice to have; the gain sweep turns Proposition 1 into an investment rule; and the fleet study turns (10) into a scaling rule for a multi hospital provider. Read together they say that building fast is the easy part, and that the scarce and rate limiting resource in enterprise healthcare software is trustworthy external feedback delivered through a validation cycle that cannot be rushed [13].

Several limitations are worth stating plainly. The model is linear and the true dynamics of a product team are not; the contraction assumption can fail if a loop overcorrects or if the target moves adversarially, in which case Proposition 1 no longer guarantees convergence and the system can oscillate [5]. The experiments are simulations of loop dynamics, not a clinical trial, and the feature vector is an abstraction of qualities that are in practice hard to measure. The safety gate is modelled as a hard projection, whereas real regulatory approval is itself a slow stochastic process. Finally, the federated averaging in (10) assumes honest independent feedback, and privacy preserving aggregation across institutions carries obligations that a deployment must meet in full [6], as does guarding against manipulation of the shared signal [14].

These limitations point to the future work. Replacing the linear operators with learned nonlinear ones, letting the gains adapt online through a reinforcement signal [3], and modelling the safety gate as a chance constraint would each bring the model closer to practice. Neuromorphic and event driven substrates may let the inner loop run faster still without raising its energy cost [16], and smart sensing at the bedside would enrich the outer loop with higher quality signals [19].

8. Conclusion and Future Work

We treated the three product development loops of agentic coding, developer feedback, and external feedback as a single three timescale feedback control system acting on a shared feature vector, and we applied it to large scale enterprise healthcare software. We defined value alignment, decomposed the value error into an implementation gap, an alignment gap, and a knowledge gap, and proved two results a practitioner can use: under timescale separation the built product converges geometrically to true user value at the rate of the slowest loop, and without separation the coupled convergence rate equals one minus the smallest loop gain. We proposed the Triple Loop Healthcare Delivery Engine, which binds the three loops to an enterprise platform through a clinical safety gate and a federated cross tenant aggregator, and we validated the whole picture with reproducible experiments. The ablation confirmed that

each loop removes exactly one gap, the gain sweep confirmed that the slowest loop is rate limiting, and the fleet study quantified the bias variance trade off of sharing feedback across hospitals. Loop engineering, understood this way, is the discipline of separating timescales, investing in the slow loop, and never shipping past the safety gate. The result is clinical software that is built fast, built right, and built for the people who use it [2].

Acknowledgements

The authors thank their colleagues and institutions for supporting this work. All code and data needed to reproduce every figure and table are available from the corresponding author.

References

[1] S. Singamsetty and S. Sanakkayala, "Agentic AI-driven portfolio optimization: a hybrid approach for optimized stock selection and deep learning in algorithmic trading," *Multimedia Systems*, vol. 32, no. 1, art. 70, 2026.

[2] S. Satyanarayana and S. Singamsetty, "Harnessing reinforcement learning for agile portfolio management in Nifty 50 stock analysis," *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC)*, vol. 4, no. 1, pp. 32–42, 2024.

[3] S. Satyanarayana, S. Kilaru, and K. Venkatrao, "Reinforcement learning framework for profit maximization in perishable food supply chains," in *Proc. 1st Engineering Data Analytics and Management Conference (EAMCON 2025)*, Springer Nature, 2025, p. 156.

[4] S. Singamsetty, S. Singamsetty, S. Satyanarayana, and P. R. Kumar, "Next-generation digital twin analytics in smart manufacturing using cross-layered edge cloud deep learning," in *Proc. 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, 2026, pp. 1–6.

[5] K. R. Rao, D. Nandan, and S. Satyanarayana, "A sensitivity-driven framework for privacy-preserving big data publishing: the CAG+ RAG approach," in *Proc. 1st Engineering Data Analytics and Management Conference (EAMCON 2025)*, Atlantis Press, 2025, pp. 189–198.

[6] P. S. Rao and S. Satyanarayana, "Privacy preserving data publishing based on sensitivity in context of big data using Hive," *Journal of Big Data*, vol. 5, no. 1, art. 20, 2018.

[7] S. Satyanarayana, Y. Tayar, and R. S. R. Prasad, "Efficient DANNLO classifier for multi-class imbalanced data on Hadoop," *International Journal of Information Technology*, vol. 11, no. 2, pp. 321–329, 2019.

[8] Y. Tayar, R. S. R. Prasad, and S. Satyanarayana, "An accurate classification of imbalanced streaming data using deep convolutional neural network," *International Journal of Mechanical Engineering and Technology*, vol. 9, no. 3, pp. 770–783, 2018.

[9] S. Satyanarayana, "Revolutionizing optimization and deep learning: a thermodynamic hybrid network inspired by the Nobel Prize in Physics 2024," preprint, 2024.

[10] S. Satyanarayana, "Range-valued fuzzy colouring of interval-valued fuzzy graphs," *Pacific Science Review A: Natural Science and Engineering*, vol. 18, no. 3, pp. 169–177, 2016.

[11] C. Mamatha, S. Satyanarayana, and T. K. Mohammad, "Enhanced automated system for early breast cancer prediction and classification using machine learning techniques," in *AIP Conference Proceedings*, vol. 3418, no. 1, AIP Publishing, 2026.

[12] A. Gupta, D. Nandan, S. Satyanarayana, and N. R. Rao, "Deep learning approach for brain glioblastoma detection: integrating efficient segmentation and survival rate prediction," in *Proc. 2026 IEEE International Conference for Convergence in Computing Technology (I3CTCON)*, 2026, pp. 1–7.

[13] S. Satyanarayana, T. Khatoon, and N. M. Bindu, "Breaking barriers in kidney disease detection: leveraging intelligent deep learning and artificial gorilla troops optimizer for accurate prediction," *International Journal of Applied and Natural Sciences*, vol. 1, no. 1, pp. 22–41, 2023.

[14] S. P. S. Appalabatla, A. S. Kumar, A. P. Sai, A. Sanjana, and S. Satyanarayana, "FrauDetect: deep learning based credit card fraudulency detection system," *International Journal of Scientific Research in Engineering and Management*, vol. 7, no. 6, 2023.

[15] N. M. Bindu and S. Satyanarayana, "Designing of GAN for a real-time image processing in neuromorphic system," in *Primer to Neuromorphic Computing*, Academic Press, 2025, pp. 21–44.

[16] S. Satyanarayana, T. Khatoon, and N. M. Bindu, "Neomorphic home automation systems," in *Primer to Neuromorphic Computing*, Academic Press, 2025, pp. 97–125.

[17] G. Vaisali, K. S. Bhargavi, S. Kumar, and S. Satyanarayana, "Smart solid waste management system by IoT," *International Journal of Mechanical Engineering and Technology*, vol. 8, no. 12, pp. 841–846, 2017.

[18] A. P. Begum, S. Satyanarayana, and K. Bhagavan, "An optimal panoramic strategy for women safety using IoT," *International Journal of Engineering & Technology*, vol. 7, no. 1.6, 2018.

[19] S. Marrapu, S. Satyanarayana, V. Arunkumar, and J. D. S. K. Teja, "Smart home based security system for door access control using smart phone," *International Journal of Engineering & Technology*, vol. 7, no. 1, p. 249, 2018.

[20] A. V. S. N. Murty, B. N. Jagadesh, K. Bhagavan, and S. Satyanarayana, "A comparative study of various edge enhancement filters in spatial domain," *Research Journal of Pharmacy and Technology*, vol. 9, no. 12, pp. 2403–2406, 2016.

[21] Chen, M. et al. (2021). Evaluating large language models trained on code. *arXiv:2107.03374*.

[22] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. and Cao, Y. (2023). ReAct: synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*.

[23] Wang, L. et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 186345.

[24] Brown, T. et al. (2020). Language models are few shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 1877–1901.

[25] Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

[26] Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61–72.

[27] Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

[28] Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley.

[29] Åström, K. J. and Murray, R. M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

[30] Kokotović, P., Khalil, H. K. and O'Reilly, J. (1986). *Singular Perturbation Methods in Control: Analysis and Design*. Academic Press.

[31] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3), 400–407.

[32] Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems and Control Letters*, 29(5), 291–294.

[33] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3, 133–181.

[34] McMahan, H. B., Moore, E., Ramage, D., Hampson, S. and Agüera y Arcas, B. (2017). Communication efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics (AISTATS)*, 1273–1282.

[35] Amershi, S. et al. (2019). Software engineering for machine learning: a case study. *IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300.

[36] Sculley, D. et al. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2503–2511.

[37] Sittig, D. F. and Singh, H. (2010). A new sociotechnical model for studying health information technology in complex adaptive healthcare systems. *Quality and Safety in Health Care*, 19(Suppl 3), i68–i74.

[38] Bates, D. W. et al. (2003). Ten commandments for effective clinical decision support. *Journal of the American Medical Informatics Association*, 10(6), 523–530.

[39] Henry, K. E., Hager, D. N., Pronovost, P. J. and Saria, S. (2015). A targeted real time early warning score (TREWScore) for septic shock. *Science Translational Medicine*, 7(299), 299ra122.

[40] Sendak, M. P. et al. (2020). Real world integration of a sepsis deep learning technology into routine clinical care. *JMIR Medical Informatics*, 8(7), e15182.