International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

# SDLC, AI, FDE and Frontier AI

**From SDLC to Frontier AI: Same Play, New Tools**

**S. Muralikrishnan**

**Abstract:** ***The article compares traditional software development with AI-assisted engineering and questions whether terms such as AI-DLC, Frontier AI, and Forward Deployed Engineering represent a real change or simply repackage familiar practices. AI can speed up requirements analysis, design, coding, testing, deployment, and operations, but human judgement remains central to business needs, architecture, security, regulation, production approval, adoption, and revenue. A 10% human intervention test is proposed to assess whether AI has genuinely changed the full journey from business idea to commercial value. The discussion also argues that customer-embedded engineering existed long before FDE. Its present value should therefore be judged by faster delivery, greater autonomy, measurable business outcomes, and the conversion of field experience into reusable product knowledge. New labels have practical meaning only when supported by clear evidence of reduced intervention, improved quality, wider reuse, and shorter time to production.***

**Keywords:** AI-assisted development, Software development lifecycle, Forward Deployed Engineering, Frontier AI, Human intervention

An honest comparison on what has actually changed, where the hype comes in and what should make Forward Deployed Engineering genuinely different.

The argument on this
Decades back, learning coding did not build a complete system. Engineers had to understand the business requirement, architect the solution, prepare the high-level and low-level design, connect databases, APIs and middleware, communicate with upstream and downstream systems, test the full application, support UAT, move it to production and monitor it.

The same expectation should apply now. Learning a Claude API, Gemini API or any other AI model is only the starting point. The engineer has to use the AI elements that are relevant, connect them with the customer's data, applications and processes, test the complete solution, take it to production, make it consumable and support the business outcome.

The technology is new. The responsibility to build the full working system is not new

## 1. The 10% Human Intervention Test

There can be a major difference between the earlier era and now only when the complete chain can be handled with, say, 10% human intervention. That means the AI should not stop with generating code or suggesting an architecture. It should be able to take the business intent and carry almost the full system through architecture, design, development, integration, testing, security, deployment, monitoring, issue resolution, productisation, consumption across geographies and continuous commercial improvement.

The Internet evidence available today does not establish that this complete enterprise chain can generally be delivered with only 10% human intervention. Current agents can work independently for longer periods and can automate selected activities. However, leading AI-DLC descriptions still keep humans in critical decisions, business context and oversight. Published evidence also highlights continuing quality, integration, governance and production risks. So, the 10% number should be treated as a test for a real structural change, not as a current industry fact.

*Unless AI can carry nearly the entire idea-to-revenue chain with only limited human judgement and intervention, the lifecycle is accelerated, but the fundamental play has not changed*

Even if an AI model can propose the architecture, generate most of the code and create tests, humans still remain responsible for the business idea, market need, customer context, risk acceptance, regulatory interpretation, production sign-off, commercial model, adoption and revenue. These are not small activities around the edge. They decide whether the system is useful at all.

## 2. Is the overall play really different?

| | Pre AI era | Now (AI-assisted) |
|---|---|---|
| Business idea | Human identifies the need and expected value. | Human still identifies the need, market and expected value. |
| Requirements | Analysts and engineers discover, refine and document. | AI can draft and challenge; humans still settle intent and trade-offs. |
| Architecture | Architects select patterns, platforms and interfaces. | AI can propose and document; humans still own context, constraints and acceptance. |
| Build and integration | Teams code, connect APIs, middleware, data and infrastructure. | AI can generate more of the implementation; the same enterprise integrations remain. |
| Test and release | Unit, integration, system, UAT and controlled production release. | The same controls remain, with additional model evaluation, grounding and safety checks. |
| Operate | Monitoring, incident handling, fixes and enhancements. | AI can assist or automate selected actions; accountability and difficult exceptions remain human. |
| Market and monetise | Humans drive positioning, channels, adoption and revenue. | AI can assist content and analysis; humans still own product-market judgement and commitments. |

The honest answer is: not yet in the full business sense. AI changes how much work can be performed inside each stage and how quickly the stages can move. However, the system still has to cross the same business and engineering boundaries from idea to architecture, build, integration, acceptance, production, adoption and revenue.

The difference becomes major only when AI can reliably carry those boundaries by itself and the human role reduces mainly to the original business idea and a few high-value decisions. That is not the general enterprise position today.

## 3. SDLC vs AI-DLC

AI-DLC is useful when it means that AI is placed across the lifecycle, not only inside the coding activity. AI can help elaborate requirements, propose architecture, prepare work plans, generate code, create tests, review defects, analyse logs and assist operations.

However, AI-DLC does not create a completely different destination. The destination is still a secure, usable and supportable production system that delivers the intended business outcome.

**What AI-DLC changes**

- The speed at which requirements, designs, code, tests and documents can be produced.
- The amount of routine work that can be delegated to agents.
- The ability to run multiple engineering activities in parallel.
- The need to evaluate non-deterministic outputs, context, grounding, tool use and model cost.

**What AI-DLC does not remove**

- Human understanding of the business problem and customer environment.
- Enterprise architecture, security, data, integration and operational constraints.
- UAT, release accountability, regulatory responsibility and production risk.
- Product adoption, market positioning, geography-specific needs and revenue ownership.

*AI-DLC is an AI-accelerated and AI-participative lifecycle. It should not be presented as a replacement for engineering discipline*

## 4. Where Frontier AI becomes hype

Frontier AI can be a fair term for the latest capable models and for engineering that handles agent autonomy, context, evaluation, safety, model operations and inference economics.

The hype comes in when the term is used to suggest that the following expectations are new:

- Going beyond API knowledge.
- Understanding the customer's business.
- Architecting the complete solution.
- Connecting upstream and downstream systems.
- Troubleshooting inside the customer environment.
- Building a new component when the requirement demands it.
- Taking the system to production and supporting the outcome.

All these were part of serious systems engineering in the earlier era as well. Engineers and architects were present at customer locations, understood the live environment, fixed issues, changed interfaces, created components and worked with remote teams to stabilise the full system.

Therefore, Frontier AI is not unique merely because it combines AI knowledge with production engineering. The uniqueness has to come from a measurable change in autonomy, speed, scale, quality or commercial outcome.

## 5. FDE is congruent with the earlier customer-embedded model

Forward Deployed Engineering is fully congruent with how strong engineering teams operated earlier. People were placed close to the customer because requirements were not always complete, production environments behaved differently, integrations failed, business priorities changed and sometimes a new component had to be built on the fly.
Calling this role FDE does not make customer proximity new. Resident engineers, onsite coordinators, solution architects, implementation engineers, product specialists and senior developers have performed similar work across technology eras.
So, an FDE should not be positioned only as:

- An engineer sitting at the customer location.
- A senior developer who can troubleshoot.
- A person who can discuss with business and engineering teams.
- A person who can write production code quickly.

These are important capabilities, but they are not unique enough by themselves.

*Customer embedding is the base model. FDE uniqueness has to come from what the embedded engineer can now compress, connect and send back into the product at a speed and scale that was not earlier possible*

## 6. What should make FDE unique

The strongest difference is not onsite presence. It is the combination of customer context, production engineering, AI-assisted execution, decision authority and the field-to-product feedback loop.

| Pre AI era | Now (FDE) |
|---|---|
| Understands the requirement and coordinates delivery. | Turns an unclear business problem into a working production use case within the customer environment. |
| Troubleshoots and builds changes for the account. | Uses AI to inspect, build, test and debug faster, while retaining production accountability. |
| Solves the immediate customer issue. | Converts the field solution into reusable product capability, patterns, evaluation sets and accelerators. |

| | |
|---|---|
| Sends feedback to product or central teams. | Creates a short, evidence-led field-to-product loop where live gaps directly influence product priorities. |
| Works within a project or support boundary. | Owns the path from business intent to use, adoption, measurable value and repeatable scale. |
| Depends on a broad delivery chain for each change. | Has sufficient authority, context and tools to co-engineer across product, platform and customer systems without long hand-offs. |

A genuine FDE model should show that a small accountable team can reduce the distance between an unclear need and production value. It should also show that each customer deployment improves the reusable product, rather than becoming one more isolated custom solution.

## 7. A simple test for Frontier AI and FDE claims

Instead of accepting a new name, the claim can be tested through simple questions.

- Autonomy: What percentage of the complete idea-to-production chain is handled without human intervention, and not only the coding activity?
- Architecture: Can the model independently understand enterprise constraints and take accountable architecture decisions, or does it mainly provide options for a human architect?
- Production: Can the system handle release, security, failure, rollback, monitoring and support without continuous human judgement?
- Business: Can the system convert a business idea into a product that customers will consume, across geographies, and make it revenue generating?
- FDE: Is the engineer only embedded, or does the engineer compress the full cycle and convert field learning into reusable product value?
- Evidence: Is the claimed difference visible through time-to-production, quality, adoption, revenue, reuse and reduced intervention?

If these questions cannot be answered with evidence, the difference is mainly naming and positioning. If they can be answered, then there is a real change in the operating model.

## 8. Conclusion

The play from the older era to the current era has more continuity than the current hype normally acknowledges.

Earlier, learning COBOL, Java, C#, databases, middleware or APIs did not by itself create an end-to-end engineer. The engineer had to combine all these elements and make the system work in the customer's environment. Today, learning model APIs or agent frameworks does not by itself create an AI engineer. The engineer has to combine models, enterprise data, applications, security, evaluation, infrastructure and operations and make the complete system work.

AI provides a major acceleration opportunity. It can generate, analyse, test and operate more than the earlier tools could. But unless the complete system and business chain can be handled with only around 10% human intervention, it is difficult to say that the fundamental play has changed. We are still moving from a human business idea to a designed, integrated, tested, consumable and revenue-generating system.

The same honest view applies to FDE. The customer-embedded engineer is not new. What can be new is the speed, autonomy and closed-loop product improvement achieved by a small empowered FDE team using AI across the lifecycle.

***Frontier AI should be measured by real autonomy across the complete value chain. FDE should be measured by how quickly customer context becomes production value and reusable product intelligence. Without these two differences, the new terms remain largely a new label around a familiar engineering play.***