Thoughts on Complex System and Software Development Paradigms for Discussion

STEFAN POLEDNA
CTO TRUSTMOTION

MAY 7, 2026

trustmotion
part of NXP

Agenda

01 **Complex Systems and SW Development Approaches**

02 Disruption Mechanisms of AI

Complex Systems and SW Development Approaches

Best practices from complex cloud SW development

1. **Service-Oriented Architecture (SOA) and Microservices:** better re-use, highly parallel and independent development
2. **Agile and DevOps:**
   Highly iterative development, collaboration, fail-fast and fix
3. **Cloud-Native Development:**
   Automation from the cloud to testing of software
4. **Continuous Integration and Continuous Deployment (CI/CD):**
   Extensive automation for verification and validation (V&V) enabling fast & frequent deployment
5. **Open Source:**
   Standardization of interfaces and glue-code for re-use

Implicit and hidden assumptions

- Cost of software deployment is almost free
- Innovation and feature updates over “correctness” of functionality
- Security is a concern
- Safety is typically not an issue due to the nature of the applications



Benefits of Following a Cloud IT Software Development Style

- **Customer centricity and economy**
  Fast and early deployments and rollbacks are cost-effective

- **Rapid Innovation**
  Iterative improvement by continuous updates and A/B testing

Is this the only reference point we should consider for the SDV software development?

Lessons to be Learnt from Chip Development

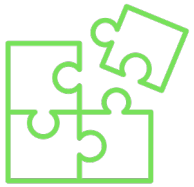
Correct-by-Design is crucial

Best practices from Chip Development

1. **Sign-off Process:**
   Verification steps to ensure temporal, functional and resource usage constraints (e.g., timing closure, power budgeting, …) for sign-off
2. **Standardization:**
   IP blocks, architecture patterns and evolutionary design based on standards
3. **EDA Tools:**
   Mature and integrated tooling landscape covering the whole design life-cycle
4. **Shift left:**
   Focus on identifying issues as early as possible in the design process via simulation, static & formal verification and emulation & prototyping, FPGA based accelerated testing

Implicit and hidden assumptions

- Needs to be first-time-right!
- Smaller issues can be fixed with software or by describing limitations (erratas)
- Cost of re-spin could easily amount to X-millions and causes product delay of several months



Enable agile development on IP block level

Ensure reusability and first-time-right and **correct-by-design** for integration and production

Addressing the SDV Challenges

Benefits of following chip development style

- **Precision and Perfection**
  Flawless execution before production avoids costly iterations and post-deployment updates

- **High Reliability**
  Rigorous Validation via extensive tooling and processes

This is another reference point for the SDV software development

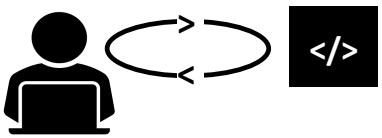
Agenda

01 Complex Systems and SW Development Approaches

02 **Disruption Mechanisms of AI**

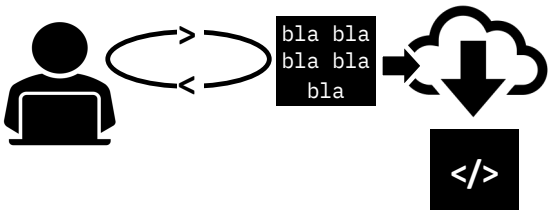
Evolution of Coding

0. Traditional Coding



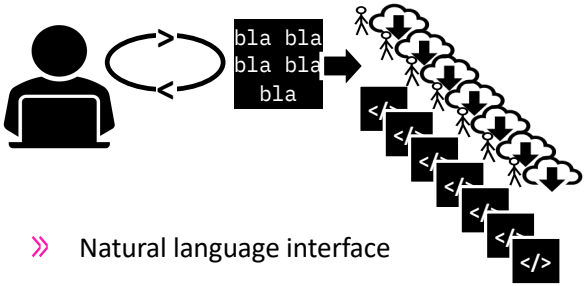
- Need to write formal language (where humans are bad)
- No context (other than libraries and component re-use)

1. AI Supported Coding



- Natural language interface
- AI models provide context from almost the whole worlds knowledge
- Lack of world understanding and statistical nature leads to issues such as hallucination
- **BIG PRODUCTIVITY IMPROVEMENT**

2. Multi-Agent based Coding



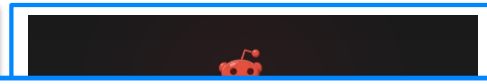
- Natural language interface
- Multiple AI agents can cooperate with different targets
- Provide a cheap fully scalable programmer army
- Diversity leads to less fragility
- **REVOLUTIONARY PRODUCTIVITY IMPROVEMENT**

Notable AI Developments – A Selection



China is even improving beyond the US with a different approach

- they to build highly efficient smaller models
- Qwen, DeepSeek
- their pragmatic approach might even disrupt the US ecosystem



red.anthropic.com

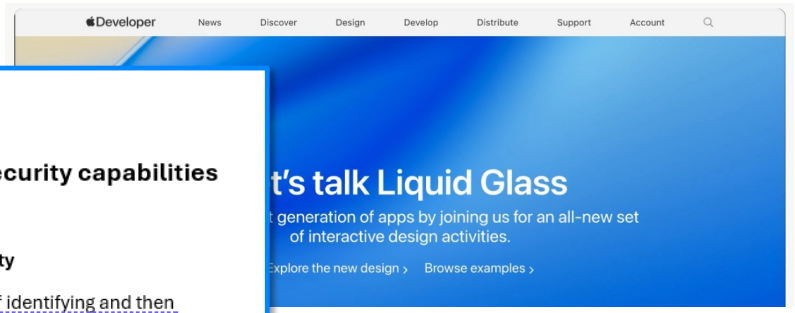
Assessing Claude Mythos Preview's cybersecurity capabilities

April 7, 2026

The significance of Claude Mythos Preview for cybersecurity

During our testing, we found that Mythos Preview is capable of identifying and then exploiting zero-day vulnerabilities in every major operating system and every major web browser when directed by a user to do so. The vulnerabilities it finds are often subtle or difficult to detect. Many of them are ten or twenty years old, with the oldest we have found so far being a now-patched 27-year-old bug in OpenBSD—an operating system known primarily for its security.

- A 27-year-old OpenBSD bug[4]
- A 16-year-old FFmpeg vulnerability
- A guest-to-host memory corruption bug in a memory-safe virtual machine monitor
- Remote code execution in FreeBSD
- Linux kernel privilege escalation
- Web browser JIT heap sprays
- Logic vulnerabilities and exploits
- Cryptography libraries
- Web application logic vulnerabilities
- Kernel logic vulnerabilities



t Browser by running hundreds of
single project for weeks, coordinating their
em write over a million lines of code and

The AI Cost Curve:
When Economics Breaks before Anyone Notices

- If you step back from all the AI hype and just look at the cost line, the picture becomes very simple: the price of intelligence is collapsing.
- And in business, **when cost collapses, everything else follows.**
- AI didn't make tasks "a bit cheaper."
  It made tasks **10× → 100× → sometimes 1,000× cheaper**.
- And Clayton Christensen's principles kicks in immediately:
  - **10× cheaper:** the market resets
  - **100× cheaper:** new markets appear
  - **1,000× cheaper:** the old model becomes economically impossible
- This is exactly where we are now. Welcome to 2026!
- Models get faster and cheaper every quarter.
- OpenAI slashes inference costs, Meta runs models locally, Anthropic boosts reasoning for less money. Suddenly, work that required entire departments costs cents.
- That's a tech improvement in another dimension.
  That's an **economic disruption event**.

Zero Marginal Cost: The Physics of AI-Native Companies

- **What makes AI-native startups fundamentally different is not the technology - it's the economics.**

- **AI-native companies:**
  - get better as usage grows
  - scale without hiring
  - automate workflows instead of documenting them
  - ship weekly, not quarterly
  - run on infrastructures incumbents can't adopt without breaking their P&L
  - produce output at near-zero marginal cost

- This is why small teams suddenly feel "bigger" than companies with thousands of employees.

  - The software doesn't ask for permission
  - It doesn't need training
  - It doesn't take weekends off
  - It just runs
  - Every day
  - Cheaper and faster

Thank you

Thoughts on Complex System and Software Development Paradigms for Discussion

TrustMotion
www.trustmotion.com

trustmotion
part of NXP