Rapid Engineering with AI

Building at the Speed of Thought

AI Fund Engineering Practices

The Paradigm Shift

From "Vibe Coding" to Rapid Engineering

Like architects who design buildings, not construct them:

- The effort and cost of coding → 0
- Engineering decisions remain critical

We're not writing less code, we're making more decisions

What is Rapid Engineering?

- **Effort of implementation** approaches zero
- **Engineering judgment** becomes everything
- **System thinking** over syntax knowledge
- **Architecture decisions** over implementation details

The engineer becomes a conductor, not a performer.

The 0-to-1 Journey

Venture Studio Reality

```
Idea → Prototype → First Customer → Enterprise
  ↓         ↓              ↓              ↓
Hours     Days          Weeks          Months
```

Optimization objective constantly shifts:

- Speed first
- Then connect the dots
- Without painting yourself into corners

Engineering = Multi-Axis Optimization

Early Stage: Optimize for speed
Growth Stage: Optimize for stability
Enterprise: Optimize for security & compliance

The New Reality:

- Put objectives in the spec
- AI generates code to match
- Code is disposable artifact

Architecture Exploration Advantage

Old World: Stuck with your first architecture forever

Rapid Engineering: Try 3 architectures in a week

- Test microservices vs monolith
- Experiment with data stores
- Validate scaling approaches
- Pivot based on real learnings

The Agentic Revolution

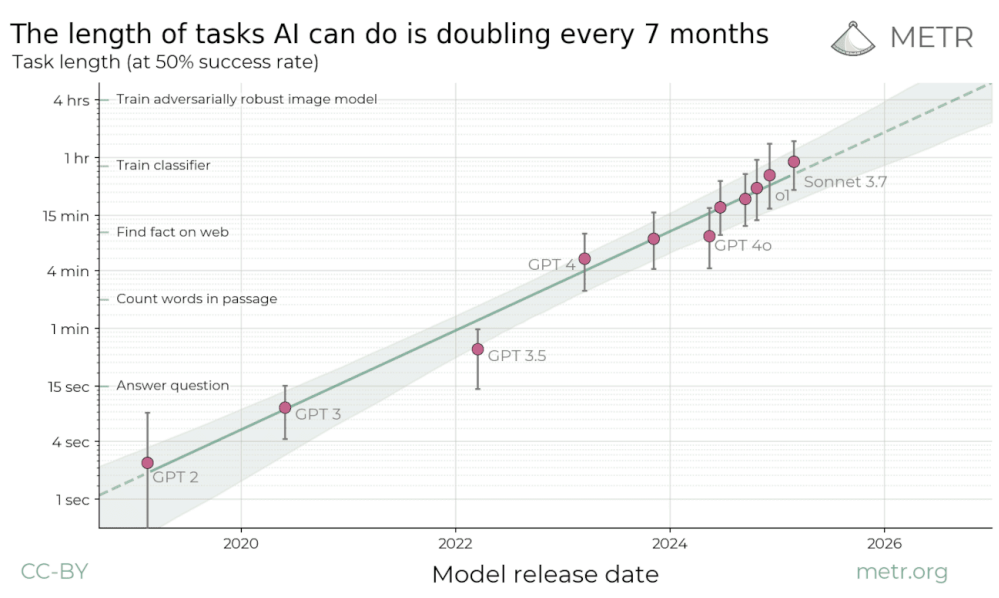
Why Agentic Search > RAG

RAG (Cursor): Retrieves similar code chunks
Agentic (Claude Code): Explores with purpose

Claude Code with Opus 4:

- Searches intelligently across files
- Builds understanding iteratively
- Maintains context between searches
- Adapts strategy based on findings

Tasks complexity exploding



- Task complexity increases exponentially
- What was impossible yesterday is routine today

How to Use Claude Code Effectively

Do:

- **Start with context**: "We're building X using Y"
- **Be specific**: Include file paths, function names
- **Think in tasks**: "Find all API endpoints..."
- **Iterate**: Build on previous responses

Claude Code: What to Avoid

Don't:

- Assume it knows your setup
- Accept first output blindly
- Skip testing
- Share secrets or credentials

Remember: Claude Code is powerful but needs guidance

Example: Effective Claude Code Usage

❌ **Bad**: "First open user.routes.ts, then copy line 15..."

✅ **Good**: "Add Joi validation matching auth.routes.ts"

Key: Describe the outcome, not the steps!

Engineers as Multipliers

New Engineering Superpowers:

1. **Tool Curation**: Which AI for which task
2. **Pattern Teaching**: Effective prompts & workflows
3. **Quality Gates**: Good vs problematic output
4. **Architecture Guidance**: Avoiding dead ends

One engineer can enable 10x more experiments

For Non-Engineers

Rely on Engineers for:

- **Tool Selection**: Claude vs Copilot vs Cursor
- **Best Practices**: How to prompt effectively
- **Quality Assessment**: Is this production-ready?
- **Architecture Review**: Will this scale?

Engineers aren't gatekeepers, they're enablers

Rapid Engineering Workflow

1. **Define the experiment** (Business)
2. **Architect the approach** (Engineering)
3. **Rapid implementation** (AI + Human)
4. **Validate with users** (Product)
5. **Iterate or pivot** (Team)

Days, not months

Technical Debt: Strategic Choice

Traditional View

"Technical debt is bad"

Rapid Engineering View

"Technical debt is a tool"

- **Know what debt you're taking**
- **Plan the payback timing**
- **Keep refactor paths open**

Rapid Engineering is Full Stack

AI handles all layers:

- Frontend (React, Vue, etc.) ✓
- Backend (APIs, databases) ✓
- Infrastructure (Docker, K8s) ✓
- ML/Data pipelines ✓

Engineers guide: Deployment strategy & architecture

Common Pitfalls with AI Coding

1. **Not describing the outcome**
   ❌ "Fix the bug"
   ✅ "Users should see real-time updates"

2. **Assuming limitations**
   Don't assume - just try it!

More Pitfalls to Avoid

3. **Micromanaging the approach**
   Let AI find creative solutions

4. **Accepting without review**
   Trust but verify

Remember: AI surprises on the upside

The Future is Already Here

Today's Reality:

- Autonomous debugging ✓
- Architecture generation ✓
- Cross-codebase refactoring ✓
- Continuous optimization ✓

These aren't future promises - use them now!

What Remains Human

AI Can't Replace:

- Business judgment
- User empathy
- System thinking
- Human-in-the-loop design

We guide the tools, not the other way around

Key Takeaways

1. **Rapid Engineering > Vibe Coding**
2. **Speed first, enterprise later** (but plan for it)
3. **Agentic AI changes everything**
4. **Engineers are multipliers**, not bottlenecks
5. **Strategic technical debt** is a superpower

Resources

Tools:

- **Claude Code** (Opus 4) - Agentic coding
- **Cursor** - AI-first IDE
- **Gemini 2.5 Pro** - AI Studio

Thank You!

Remember:

We're not automating engineering.
We're engineering at the speed of thought.

Questions?

Contact: eli@aifund.ai