Goals to Systems: Planning, Maintaining, and Engineering AI Applications with Foundation Models

---

Milestone Planning

- Start by evaluating existing (off-the-shelf) models to understand baseline capability.
- Stronger base models reduce required effort; weak models increase cost and risk.
- Goals often change after evaluation due to ROI, resource, or feasibility constraints.
- Early demos can be misleading—moving from ~60% to near-perfect performance is disproportionately hard (“last-mile problem”).
- Real products take months or years, not weekends; incremental gains become slower and more expensive.

Maintenance Challenges

- AI products require continuous maintenance due to rapid model, cost, and capability changes.
- Even positive changes (cheaper, faster, better models) can disrupt workflows.
- Decisions like build-vs-buy can quickly reverse due to market shifts.
- Model swapping is easier as APIs converge, but quirks still require prompt, data, and workflow changes.
- Regulations (data privacy, compute access, IP rights) are high-risk and can be disruptive or fatal.
- IP uncertainty remains a major concern for IP-heavy industries.

AI Engineering Stack

- AI engineering evolved from ML engineering but emphasizes adaptation over training.
- Three layers:
    1. Application development – prompts, context, UX, evaluation
    2. Model development – training, finetuning, datasets, inference optimization
    3. Infrastructure – serving, compute, data, monitoring
- Most recent growth is in applications and app-level tooling, not infrastructure.
- Core ML principles still apply: experimentation, evaluation, optimization, feedback loops.

AI Engineering vs ML Engineering

Key differences:

1. AI engineering uses pre-trained foundation models instead of training from scratch.

2. Models are larger, more compute-intensive, and latency-sensitive.
3. Outputs are open-ended, making evaluation much harder.

Model Adaptation

- Prompt-based techniques: no weight updates, fast iteration, low data needs.
- Finetuning: updates weights, higher complexity and data needs, better quality/latency/cost.
- Prompting is often sufficient initially; finetuning is required for stricter requirements.

Model Development Details

- Modeling & training tools remain (TensorFlow, PyTorch, Transformers), but are no longer mandatory for app builders.
- Training terminology:
    - Pre-training: train from scratch (most expensive)
    - Finetuning: continue training for specific tasks
    - Post-training: often provider-side finetuning
- Prompt engineering is **not** training, despite common misuse.

Dataset Engineering

- Open-ended outputs make annotation harder than traditional ML.
- Focus shifts from tabular data to unstructured data handling.
- Key tasks: deduplication, tokenization, retrieval, quality control, safety filtering.
- Data remains a major competitive advantage.
- Data needs decrease from pre-training → finetuning → prompting.
