TODO and NOT TODO

Goal

Suppose you are the user of a complex domain specific software/programming language. You want to use LLM/agent to improve your productivity.

Typical roles including and not limited to model providers (e.g. OpenAI, Anthropic), agent platform providers(e.g. LangSmith, Coze Studio, etc.), and software companies themselves also doing their own LLM/agent applications. The problem is their solutions don't have good performance on your tasks. And you want to maximize their power to avoid duplicated development for the same functionalities.

What should you do and what should you ask from external players to build a good LLM/agent system on your domain-specific tasks?

- What is a typical workflow from building dataset to developing LLM/agent application software?
- What are the roles in it? And each role typically in charge of which stages?
- What stage/task that can only be implemented by you exclusively? Verify this by imagining what will happen if other roles developing the same functionalities and compete with you
- What stage/task that you should avoid because other roles will do and render you no advantages?

Roles

- You (domain expert + actual user of the software)
- External AI tech providers (model/agent infra + generic LLM capabilities)
- The software vendor (the people who built your domain-specific software)

A Typical Workflow

Stage	Description	Outputs
1. Problem Definition	Define exact user tasks, workflows, pain points, and success metrics.	Clear task specs & KPIs
2. Data Identification & Collection	Find where task-relevant domain data exists (logs, scripts, docs, code, examples, configs).	Raw domain data. Including and not limited to row logs, documentation, user queries, and correct outputs
3. Data Cleaning & Labeling	Remove junk, standardize format, annotate examples, add edge cases.	High-quality training/eval datasets. Requires domain expertise to ensure correctness
4. Prompt/Template Design	Create instructions, few-shot examples, structured outputs for the LLM/agent.	Prompt library
5. Model Selection & Adaptation	Choose base LLM, fine-tune or RAG with domain knowledge.	Fine-tuned model / RAG index

Stage	Description	Outputs
6. Evaluation & Benchmarking	Measure performance on real domain tasks (accuracy, efficiency, safety).	Leaderboard & performance reports
7. Agent Orchestration	Build multi-step reasoning, tools integration, retrieval systems and execution loops.	Working agent pipeline.
8. Integration with Software	Connect agent to APIs, DSL interpreter/compiler, or automation hooks in the software.	Deployed integration
9. Deployment & Feedback Loop	Release to users, collect feedback, retrain/improve.	Continuous improvement cycle

Roles and responsibilities

Role	Typical Responsibility in Workflow
Model Providers (OpenAI, Anthropic, etc.)	Stage 5 – Supply the LLM backbone; sometimes help with fine-tuning; sometimes supply eval tools.
Agent Platform Providers (LangSmith, Coze Studio, etc.)	Stage 4–7 – Orchestration frameworks, RAG infra, tool/plugin integration, experiment tracking.
Software Vendor (domain software company)	Stage 8 – Provide APIs, tool access, and sometimes embedded LLM features.
You (Domain Expert)	Stage 1–4, Stage 6, Stage 8–9 – Define tasks, curate data, design prompts, create eval datasets, verify outputs, guide integration into actual workflows.

TODO List: Stages You Must Own Exclusively

These are things only you can realistically do, because they require deep domain knowledge + access to proprietary context:

Stage	Why only you?	What happens if others try?
Stage 1: Problem Definition	You know the actual bottlenecks, workflows, and success metrics.	Outsiders will make generic assumptions, leading to irrelevant agents.
Stage 2: Domain Data Identification and Collection	You know where relevant DSL/code/data hides and which parts are critical. Only you have access to proprietary datasets(e.g. internal codebases, user interactions)	Others will miss important datasets or misinterpret them.

Stage	Why only you?	What happens if others try?
Stage 3: Labeling & Edge Cases	Only you can create correct annotations for tricky DSL/domain cases.	Wrong labels lead to hallucinations or critical failures.
Stage 4: Domain specific prompt engineering	Only you can optimize prompts for your exact use case.	Generic prompts will get worse performance
Stage 5: Domain specific finetuning	Optimize model weights for your exact use case.	Generic model will get worse performance.
Stage 6: Domain-Specific Evaluation	Only you can judge correctness in nuanced cases.	Outsiders may accept "plausible but wrong" outputs.
Stage 8: Workflow Integration	Only you can decide how the agent plugs into your actual work process.	Outsiders might make an integration that doesn't fit real usage.

NOT TODO List

Stages You Should Avoid Owning

These are best left to others, because they can do it better/cheaper/faster and you won't gain much by duplicating:

Stage	Why avoid?	Who should do it?
Base LLM Training (Stage 5)	Hugely resource-intensive; your advantage is in domain adaptation, not general reasoning.	Model providers
Agent Framework Infra (Stage 7)	Platforms like LangChain, LangSmith, or Coze Studio already handle orchestration and logging well.	Agent platform providers
Generic Prompt Engineering	Outsiders can provide base prompt patterns; you just adapt for domain.	Open-source prompt libraries or platforms
Low-Level API Hosting	Hosting & scaling the base model is not your core advantage. E.g. For cloud scaling & optimization, specifically inference infrastructure, you should use APIs provided by cloud services.	Cloud model providers

What to ask from external players

You want to push their capabilities to the max so you don't re-implement them:

From Model Providers:

- Better fine-tuning API access for your DSL/code data.
- Logit bias / token bias controls for DSL token generation.
- Cost-efficient inference for small-batch queries
- Transparent tokenization & token budget info for your DSL.
- Model-side function calling & structured output enforcement.

From Agent Platforms:

- Easier integration with domain-specific tools
- Support for custom workflows and tools such as parsers & validators for your DSL.
- Native integration with your software's APIs/tools.
- Workflow debugging & replay tooling (LangSmith-like).
- Support for running agents in offline or air-gapped mode if needed.

From Software Vendor:

- Fully documented and open APIs for automation & control.
- Access to execution sandbox to run generated DSL safely.
- Event hooks/logging to feed into agent feedback loop.
- Agreement on data export rights for fine-tuning/evaluation.

Stage #	Workflow Stage	You (Domain Expert)	Model Provider	Agent Platform	Software Vendor
1	Problem Definition	▨ **Primary**	○ Support	○ Support	○ Support
2	Domain Data Identification	▨ **Primary**	○ Support	○ Support	○ Support
3	Data Cleaning & Labeling	▨ **Primary**	○ Support	○ Support	○ Support
4	Prompt / Template Design	▨ Partial (domain prompts)	○ Support	▨ **Primary**	○ Support
5	Model Selection / Fine-tuning	○ Support	▨ **Primary**	▨ **Partial** (RAG/adaptation)	○ Support
6	Domain Evaluation & Benchmark	▨ **Primary**	○ Support	▨ **Partial** (eval infra)	○ Support
7	Agent Orchestration	○ Support	○ Support	▨ **Primary**	○ Support
8	Integration with Software	▨ **Primary**	○ Support	○ Support	▨ **Primary**

Stage #	Workflow Stage	You (Domain Expert)	Model Provider	Agent Platform	Software Vendor
9	Deployment & Feedback Loop	▨ **Primary**	○ Support	○ Support	○ Support

legend

- ▨ Primary – Role owns this stage, sets direction, makes final decisions.
- ▨ Partial – Role contributes domain/integration specifics.
- ○ Support – Role assists but does not lead.

SWOT Analysis

Role & Stage	Strengths	Weaknesses	Opportunities	Threats
You – Stage 1 (Problem Definition)	Deep domain insight, direct access to workflows	Limited AI engineering experience	Shape LLM to actual pain points	If outsourced, risk of irrelevant agent
You – Stage 2 (Data ID)	Knows where real high-value domain data lives	May lack large-scale data pipelines	Build a proprietary dataset	Others can't find same data, keeping moat
You – Stage 3 (Labeling)	Can judge correctness for DSL/domain	Time-consuming, requires consistency	High-quality fine-tune data = big advantage	Outsiders mislabel → model degrades
You – Stage 6 (Eval)	Knows real success metrics	Hard to automate eval	Build domain-specific benchmarks	Without it, model drifts
You – Stage 8 (Integration)	Knows workflows & friction points	Requires coordination with SW vendor	Seamless productivity boost	Vendor lock-in risk
Model Provider – Stage 5 (Model Selection/Tuning)	Massive infra & AI research capability	No deep domain understanding	Provide high-quality fine-tuning infra	Can swap providers, risking dependence
Agent Platform – Stage 4 (Prompt/Templates)	Best practices from many domains	Generic templates may not fit	Provide structured prompt building tools	Might compete with your own agents
Agent Platform – Stage 7 (Orchestration)	Handles complexity & logging	May be over-engineered	Integrate with your tools quickly	Vendor dependency risk

Role & Stage	Strengths	Weaknesses	Opportunities	Threats
SW Vendor – Stage 8 (Integration)	Deep API knowledge, controls software roadmap	May deprioritize your requests	Can offer native LLM integration	If they build competing agent, could sideline you

Key takeaway from SWOT

- Your exclusive advantage = domain knowledge, proprietary labeled data, and realistic evaluation.
- Outsiders' advantage = infrastructure, scalability, AI R&D.
- Collaboration sweet spot = you feed them structured domain datasets & evaluation harnesses, they handle model infra + orchestration + hosting.