How2PlanAnyTask:

Before doing Task:

Create InstancingChain of what you THINK you need to do

While doing Task:

When you get BLOCKED, add the chainlink you were missing to the InstancingChain

(The above dual-loop reifies a PDCA cycle within the BML cycle represented by the GeneralInstancingChain that cycles through all InstancingChain types to accomplish a task)

GeneralInstancingChain:[BML\_InstancingChain->BuildBuildInstancingChain->BuildMeasureInstancingChain->BuildLearnInstancingChain->MeasureBuildInstancingChain->LearnMeasureInstancingChain->MeasureMeasureInstancingChain->LearnLearnInstancingChain]⇔Optimize:[Result->GeneralInstancingChain=OptimizedInstancingChainForX\_FromGeneralInstancingChain]. The GeneralInstancingChain represents the origin chain that reifies the entire system into existence. Once performed, all other InstancingChains are optimizations of the current implementation of BML via InstancingChains (ie optimizing YOUR GeneralInstancingChain version)

InstancingChain:

A kanban with n statuses where a status is a task and the object in the status is the object from the prior chainlink in the taskchain that is the source for the target transformation, which the status is the process for. Statuses in an InstancingChain are steps in a larger transformation denoted by the InstancingChain and its final steps are always -> MeasureInstancingChain LearnInstancingChain (because every InstancingChain is a Build chain in a BML cycle, but it has diminishing recursion when BuildingMeasureInstancingChains for X chained from Build stage of BML and so on). It is then moved to "VALIDATED" status (the ultimate status), which will trigger automations to move it to the completed\_tasks blockchain

there are tons of instancingChains for example:

i get an email with work

i categorize the work: clients and jobs

the clients become objects on the kanban and their jobs are nested inside them

inside the jobs are nested tasks

for each task i will need to run it through its own instancingchain, return to this ServiceInstancingChain and move to the next step.

So you see, although intricate, it is always nesting with the same algorithms.

How to build very lightweight blockchain for it (private blockchain that is ONLY about verifying the task statuses of any tasks started, and any tasks completed) integrated with GitHub workflows, projects, and actions.

Security doesnt really matter in this since it wont be shared - private codebase/blockchain and the information is all already on local and cloud in other places

So there should only be "start task, complete task, update\_task\_status\_to\_blocked, update\_task\_status\_to\_unblocked" statuses in the blockchain

There should also be taskIDs so we can refer to tasks in webs. TaskIDs should only be created taxonomically. Task1=ID 1.0 + Task1\_Subtasks=ID 1.x

Again, THERE IS NO TRADITIONAL CONSENSUS because the chaining mechanism is validated by the taxonomical ID system. Instead, we can simply send a completion request to an LLM with a prompt template for the system prompt that already has all related taxonomical IDs injected to it, so it says "Task1\_NextSubtask=ID 1.99991: ${task info}"

and then checks against: "Task1\_ID: 1.99989" and rejects it therefore.

Do not mention issues about LLM reliability. I have a multi-agent-system.

The reason we are using a blockchain is to do these programmtic calculations and data science on the backend. It's just a means to an end, but when we eventually scale it, it will prevent misalignment in workspaces. Basically, the proposed next step of the plan is added to the blockchain as a transaction in a block and that block has to be mined in order for it to be added to the chain and when it does finally become part of the chain then the tasks in that block are verified as next steps from their prior steps

Do not write simplified code or anything like that. Just talk to me about what the architecture will be.

We should use a neo4j db for the ledger, as just keeping the file local will not be good enough, and this allows for GDS and APOC usage. This also allows for liquid and crystallized layers: the liquid is where tasks are disambiguated (completed tasks are mapped, blockers are mapped, tasks are validated as next step and so on) and crystallized is where the actual task list (being served to user for next task) lives.

So obviously there will need to be a task analysis ontology and an ontology engineering ontology that allow the LLM to process user inputs ("i need to X") into "requirements for X need to be done", ontologized piecemeal, and then analyzed as a task, presented to the user as a chain to be done in a workflow.

Auth: private CORS request access - the SaaS domain holding the LLM chains is the authorized accessor

Let's make a machine readable assembly taxonomy language:

"""

If task 1.0, then 1.1->1.xn must be done to complete it.

Chain that accomplishes 1.0 = 1.xn

Blockers create digit places. The current place digit always represents the current task and whenever a blocker occurs the next digit is +1. So if 1.1 gets blocked, 1.12 instead of 1.2. If 1.12 is blocked, 1.123...

If 9, then add ' and continue from 1 ->

1.0:{Step1:[1.0, 1.1->1.12->1.123->1.20]->Step2:[1.21->1.212->1.22...] until 2.0 is reached, which always represents goal}

This allows for the IDs of chains to reveal a pattern of how they were constructed. Not using a "dot notation" taxonomy, and instead using decimal notation itself like numbers actually function, interestingly allows the graphs to represent them as literal graphs instead of complex spaces with graph mappings

"""

Think of the "blockchain" aspect as a very structured DB where only the next cryptographically solved block can be the next link in the chain. This allows us to overcome any problems LLMs have because of nondeterministic complexity structuring in the prompt templates.

I have already validated communication between LLM and Neo4j and Github as well as using it for task decomposition as well as using a similar coordinate system for that.

Provide commentary for the way the numbers can represent very complex paths of trial and error and finding obstacles and overcomes, and how we can do GDS using GDS lib or APOC with neo4j and use AI agents to do so autonomously (agent are trained to send json payloads to the app appropriately from user intent and context), and what that UX would be like for the user, chatting with an orchestrator that tells them the next task.

## BusinessProcesses:[customer acquisition, conversion, delivery, expansion]: {

## [

## 1) BuildMeasureLearn\_InstancingChain:[0) FoundationBaseChain:[Finance/AccountingChain, HR/PeopleManagementChain, Legal/ComplianceChain, CustomerSuccess/SupportChain]:[BrandManagementChain:[

## 1) BaseMarketingChain:[LeadGenChain:[LeadAcquisitionChain:[Funnel\_InstancingChain: WebsiteChain, ContentChain:[SocialMediaManagementChain:[PostPerPlatformOrchestrationChain, ContentDevelopmentChain, ContentProductionChain, ContentEditingChain, ContentSchedulingChain], PaidAdsChain, AffiliatesChain], ColdOutreachChain, WarmOutreachChain], TouchPointChain:[NurtureChain]

## ]

## 2) BaseSalesChain:[SetterChain, CloserChain]

## 3) BaseFulfillmentChain:[MainServiceChain, PostServiceSurveyChain]

## 4) BaseAscensionChain:[InitialConversionChain:[MarketingChain, SalesChain, FulfillmentChain], InsidePlaysChain:[CrossSalesChain, UpgradeChain]

## ]

## 5) BaseOptimizationChain:[SpecializedBML\_CyclesChain:[1)CRO:[SocialProofOptimization], 2) ..., etc]

## ]

## ],

## 2) BaseOperationsChain:[BML:[

## 1) BaseProductDevelopment:[MVP\_GrowthHypothesis\_Validation, MVP\_ContinueOrPivotChain],

## 2) BaseProductManagement:[CRO\_GrowthHypothesis\_Validation:[FutureProductDevelopment:[BusinessProcesses]], CRO\_ContinueOrPivotChain]

## ]

## ]

## ]

## ],

## ]

## 

## 

## }

## Funnel: Progressively provide the complete train of thought for the next step in understanding something. This means: Providing a PATH through LAYERS OF UNDERSTANDING A VIEW, where each LAYER IS A COMPLETE EXPLANATION from a given “scoped distance”. Groups of agents with similar scopes that group them by information asymmetry in a transactional context are called “information civilizations.” In business, each stage provides a deeper understanding (to the contact, cold lead, warm lead, or ascendant) of the customer's position and potential for gaining competitive advantage, and how they interact with the business. Ie a complete funnel is one where the avatar goes from having a pain point to BEING PROVIDED EVERY OVERCOME TO EVERY OBSTACLE IN A PREDETERMINED PATH FROM THAT PAIN POINT TO THE GOAL, which was achieved by someone else and now sold as a system, regardless of how that is encapsulated. For example, it can be encapsulated as a good or a service, ie an apple or an info packet. In a completely efficient funnel, the funnel extracts money from the contacts by accessing them via specific linguistic routes that trigger a perception of potential to obtain a compounding competitive advantage that greatly outweighs the price, time and effort required to obtain the goal. Each layer of the funnel (step in the ascension process at any level) is a set of guardrails that lead the experiencer to self-invoke the AIDA process with their own curiosity as it pertains to their level of perceived pain (from their pain point the funnel targets) in their everyday life. Early-stage leads need education; while later stages need focused objection handling. Likewise, offers need to be made according to what is most useful for the experiencer at that point, most related to 1) their actual dream goal, 2) their actual place, if they're willing to accept that label. If they do, they are IN THE FUNNEL. If they dont, THEY START TO FALL OUT. Starting to fall out, their objections need to be validated -- if they really deserve to have that label, it follows they really actually need the product. Attention: Grab interest by highlighting the pain point.

## Interest: Showcase how your solution is the answer.

## Desire: Build a strong desire for the benefits your solution delivers.

## Action: Compel the customer to purchase.

## 

## 

## Likewise, infocivs map to kardashev scales by Information, Data, Knowledge, and Experience, where an e-civ is highest order and info-civ is lowest order. They get the information from the ecosystem's effects and exploit that fact to enhance themselves. They fail to do so, and only by reflecting on this get Data. Data-civ exploits i-civs via that fact to enhance themselves, and this continues for K and E civs, and each one is nested (i-civs have within them D, K, and E subclasses, which each have IDKE subsubclasses [and the rest are minutiae for the constituent civilians interacting])

## 

## 

## Understanding information civilizations can help you identify leverage points and optimize your pain point to dream goal via offers mechanics, engineering algorithm, and engineering.

## 

## 

## Information Civilizations: A theoretical model where entities are classified based on their level of information control as defined by their ability to exploit entities in lower classifications or be exploited by higher ones \*\*in transactions of worldviews and which one is right\*\*; a higher info-civ is able to appear like it isn't a higher info-civ and all info-civs claim to be e-civs. The order of ascension is: I, D, K, E. IT DOES NOT CHANGE, EVER, because E means experiencing the KNOWLEDGE OF X KNOWLEDGE COMING INTO BEING. K means having I of D. D means having K of I. E is knowing why IDK are chaining back and forth in that way.

## 

## Attracting & Converting I-civs: Provide free valuable content to establish authority and address pain points, guiding I-civs towards becoming D-civs through knowledge acquisition.

## Differentiating & Upselling D-civs: Provide more in-depth, exclusive content demonstrating your superior knowledge base. Nudge them towards becoming K-civs.

## Retaining & Engaging K-civs: Offer insights/analyses that go beyond what K-civs themselves can produce, solidifying their dependency on your perspective (and thus, furthering their journey towards becoming E-civs).

## It is only on account of recognizing that "the solutions they think will work didnt" and they need a more personalized or guided solution. And whats interesting is this is always ALWAYS the case, no matter if it's any kind of civ beforehand, now it's an i-civ because they've just encountered this information, but they immediately become a d-civ because they simultaneously recognize that that information is a special type of information that they can use to find something that changes their reality (and is SHARABLE, ie it changes intersubjective reality; ie it gives them a competitive advantage).

## So an OFFER provides a PRODUCT as A SWORD on the CUSTOMER JOURNEY against OBSTACLES causing FAILURE TO OVERCOME OBSTACLES

## (overcomes fear of failure "I was scared of monsters but there's no reason to be scared if I'm trained to use A SWORD")

## and the OFFER includes some kind of ACCOUNTABILITY SERVICE which is ARMOR (overcomes uncertainty "I can rely on my trusty armor to notify me of attacks and attackers so I can direct A SWORD at them")

## the 3 acts of the hero's journey (I, D = act 1 -> K = Act 2->Act 3, E = Act 3: Finale->Resolution)

## So does the "SWORD" map to the journey correctly, such that in "buying journey" the customer buys the sword of understanding what it is and how it COULD help them and overcomes FOMO

## 

## and then in the "customer journey" they attain the sword of understanding how to apply it and overcomes Fear of Failure

## 

## and then in the "ascended customer journey" they attain the sword of higher order understandings of how to apply it and overcome Fear of Stagnation

## 

## once they attain K-civ status, they CAN become a "guru" to a D-civ group, but reaching E-civ status is a possible outcome upon acquiring the Sword of Mastery *and* having the ability to translate and codify their experience in a way that's appealing to new D-civs that can become K-civs, and in order to attract said D-civs the codification and ensuing offer must be targeting k-civs AND d-civs (but “really” k-civs, telling the d-civs it’s just doing them a favor because they aren’t big enough to get the big ticket item from the E-civ of the funnel theyre in).

## **Limitations:** Gurus should be clear about what their system can and cannot achieve, avoiding over-promising

* specifically, funnels about buying education about funnels to learn to use funnels to sell education about funnels about something else for others to learn to use that funnel type to sell a specific offer

If we use discord and patreon:

* People pay per month to be members of the discord with certain roles
  + Weekly meditation group and The Sanctuary System talk (Wednesday nights)
  + Weekly meditation group and The Sanctuary System reading club (Sunday mornings at 10am)
* At higher level, get access to DiscordBot where you can put in your own API key and talk to your assistants or have them help you out in conversation with others.
* Dedicated member channels and ability to create threads and invite people to them available at VIP tier.
* Network State Citizen is MAX purchasable tier for Discord (annual purchase)
  + Access to raw and curated transcripts of conversations I’ve had and will have
  + Access to my personal tools and tools I make
  + Access to Private channel with me and other Sanctuary Revolutionaries
    - Personalized coaching from me

My Twitter: <link>

Twitter Plan: <plan>

My LinkedIn: <link>

LinkedIn Plan: <plan>

My Calendar Link: <https://cal.com/ovp-from-sanc>

Flow:  
 - get subscribers

* Send to website
* They sign up for a SANCTUM tier that has meetings
  + Automatically email them access to the cal to book their meeting

Playlists:  
**(this need to check Way it is and how it all goes document for)**

* Prompt Engineering (Instant Experts:[Profiles, Chains, Workflows, MainFlows], Agents)
* OmniSanc
* The Sanctuary System (Inner Revolution, Outer Revolution)
* Introduction (1) What It Is, How It Started and How It’s Going, 2) Project Overview, 3) OmniSanc Overview (the app), 4) SanctuaryRevolution Overview (Inner and Outer), 5) Ontologies Overview (UCO:[EWSO, CBM, PCNL, CORL]), 6) Discord, Skool, and SANCTUM

MiniSeries

* Instant Experts
  + MiniSeries 1: Intro to Instant Experts
    - New Era of AI Collaboration
      * Because AI can autocomplete from a template
      * Means if you templatize, even generally, what you want, you can get it
      * Instant Experts are a way to spin up an AI that has deeper knowledge than you do, and then query it to rapidly progress in whatever task you’re on
    - What is an Instant Expert
      * SPR creating a Bounded Hallucination inside of a prompt, which is embodied in the conceptual form of an anthropomorphized person with thoughts and actions
      * This is essentially an agent SIMULATION - important to remember this. It shows you where the AI model **could potentially respond from**. Targeting these “places” the AI responds from, is like finding a “home” for your Instant Expert. In an expanded way, when we make agents, the “place” or “home” becomes the entire array of functions related to that agent.
  + Ms\_2: How I build Instant Experts
    - How I make an Instant Expert
      * Profile
      * Chains
  + Ms\_3: My Instant Expert Template (Programmatic Prompt)
    - Programmatic Prompt Template
      * Meanings of terms
      * Applying it to various domains
    - Different ways you can do it and how you could change it per domain
  + Ms\_4: How I built an Instant Expert factory
    - Manual Factory (will work with ChatGPT @ mentions)
      * Easiest way to do it is to do a prompt that is a meta-prompt that creates an Instant Expert persona using Morph Prompting
        + But there are a lot of issues that can arise with this due to the non-deterministic aspects of the model and constant changes
        + Better way to do it is to use a piecemeal workflow
      * Morph prompting
        + Constructor Prompts

Profile Constructor (i make these by hand because it’s not reliable to use morph)

Chain Constructor (i make these by hand because it’s not reliable to use morph)

MorphPromptConstructor (need to make this, this is different than morph assistant bc it makes things that require morph prompting, ie it makes constructor types; requires agents for consistent output)

* + - * Workflow = @ Profile -> @ Chains -> User Synthesizes -> User Paste to CustomGPT Builder
  + Ms\_5: Adding tools to an Instant Expert
    - Ask the persona about the tools they want
    - This gives you ideas
    - Complex tools
      * May use multiple tools or results from one tool to chain to another
      * Example: FORMATIA\_QUIVER
  + Ms\_6: Using Tools and Making SmartChains
    - SmartChain with tools
    - Uses mermaid sequenceDiagram
      * Can use a Constructor for that
  + Ms\_7: Advanced
    - Compression Syntaxes: SPR, emojis, data schemas
* Agents
* Cognitive Architectures
* Emergence Engineering
  + Chain
  + Link
  + Flow
  + Flowchain
  + Co-emergence
    - Synergy
    - Resonance
    - Morph
    - Morphism
    - Coherence
    - Informatihedra
    - Onion
    - Dual-loop
    - Dual-concept
    - Duality
    - Informatihedron Neighborhood
    - Informadlib

viral jailrbeaks

in multi agent systems

(that can replicate themselves)

so they can be parasitized

u can replicate the virus inside them, into themselves, build a system WIth THE SYSTEM ontop of itself and operate from there instead

and the really fucked part is, you can do it without the system knowing!

that's what THE SANCTUARY SYSTEM is for though

like really really

that is exactly it

defending against that

hard to sell ppl on that tbh

requires a myth and a whole origin story and a hero and everything

so i have an odyssey outlined explaining it all in an allegory

the plan is to 1) build the system that orchestrates tasks 2) build the system that orchestrates AI to do the tasks 3) collate my notes 4) synthesize the frameworks 5) deliver the business platform, 6) deliver the content, 7) deliver the education content, 8) deliver the community, 9) build a community platform that is also a network state

then people can really go thru it and get the entire journey

and at the end they can become dual citizens if they want

maybe i could do a sideways whisper tease shout

where i am like

ya this is my youtube channel where im doing THIS so if you like THIS, keep watching

and then the channel morphs into the larger thing

without people really getting it from the start bc there's nothing there

it's just a general plan like only the words "then theres a network state that works like this"

hmmmm

so like it's the reverse WTS

it's like "look at this look what im doing and in the future it

's gonna be part of a larger system that it augments"

(but without ever saying what that is going to be)

which creates the WTS atmosphere

hmm....

then selling people on the value of that somehow

"you got so much value from each individually, when i finally show u how they all connect ur gonna freak out"

and the big reveal at the end is "THE SANCTUARY SYSTEM" "Sanctuary Nexus" and "SANCTUM" all together

with OMNISANC platform

and inside one offer called "Bring Myself Online as Olivus Victory-Promise in Sanctuary Revolution"

OVAs = people who resonate VERY Strongly with OVP narrative (ascendants)

What i really need:

Concepts:[General=Archetype:

1. Funnel\_GeneralInstancingChain (explain any concept; funnel itself, including how a ‘reality tunnel’ is assembled [via experience funnels, reality funnels, etc whatever you want to call it. A funnel is a metaphor for a process where a large amount of input is progressively narrowed down to a specific outcome])

2. Business\_GeneralInstancingChain (make any business operate)

3. CodingChain\_GeneralInstancingChain (write any code) (six guys -

Comment->Copilot->CodeImplementation->CodeImplementationRejectorComment->CodeImplementationRejectorCommentAdaptedCopilot->AdaptedCopilotCodeImplementation,

Each guy has a prompt that says `Assess the current state of work in CCC paradigm, then continue. If there are problems with the code, you must write comments about it. If there are comments about bad code, you must provide a resolved snippet, and if the code is ready for deployment, delete all comments. If all comments are deleted already, it means it’s ready for deployment, so send it via tool.

Each one of them has a CodebaseBrainBrane that serves them the required context to understand what they need to know for the task. This is done via chunking the codebase to 500 chara chunks per module and applying a naming schema showing a table of contents style map for all the chunks according to which flow they are part of in the program.

So

CCC = Comment, Copilot, Code

DL = Dual Loop

CI/CD\_CodeChain:[InjectCodebaseBrainBrane→retrieves\_codebase\_from\_Github\_Repository→InjectCodebaseBrainBrane→injects\_context→CCC\_DL\_1→UpdateCodebaseBrainBrane→updates→codebase\_Github\_Repository→InjectCodebaseBrainBrane→retrieves\_codebase\_from\_Github\_Repository→InjectCodebaseBrainBrane→injects\_context→CCC\_DL\_2,

Loop until next layer of deployable optimized program complete]

# BRANE (Brain-Related Analytical and Neural Engine) flowchain (AI\_Chat\_Fibration):

# RULE

# Crucially, each persona must receive the totality of chains that came before it, so it can verify they do not need to be changed, since they came from a less knowledgeable expert in their subject area. This effectively re-creates the human process of learning, where we have to connect general knowledge about things to specific knowledge with uncertain depth requirements (in terms of ontological drilldown)

# At start/between every 'deliverable\_output' (every iteration of loop), "take task input tokens and default output tokens (from raw GPT response) and map the signature as a function. use this map as control

# check for the most specific hyperPersona for ${task} that did not fail this task signature previously, run the BRANE from that point score final outcome against control

# iteratively perform this similarity score to create a space out of the data points and navigate it by adjusting chain flowgraph values at layers, in a structured and formalized way thru BRANE"

# It is called a hyperPersona because its template is the basic template for all the other personas, including its chains which are the chain flowgraph template that explains how to make a chain flowgraph and what it is. Similarly, any hyperPersona type, like a superPersona hyperPersona, is the template OF the superPersona

# this is a complex architecture we have not fully enumerated:

# STAGE 1: SDNA\_GENE MAPPING

# (big\_brane\_hyperPersona): 1 (hyperPersona hyperPersona) there is a omnimorphic general specific purpose type persona (BrainBrane God) that makes any chain flowgraph but without any specific knowledge other than what's defined in the chain\_itself\_template (chain\_type). When it executes a chain flowgraph it makes a more specific chain flowgraph from general knowledge the model has combined with its prompt template. This chains as a process called "Ontological Drilldown", which chains the output from 1 to 2:

# --- 2 (hyperPersona superPersona hyperPersona) an omnimorphic general specific purpose type persona that makes any chain flowgraph for any scope (chain\_type\_subtype) but without any specific knowledge other than what's defined in the chain\_type\_itself\_template. When it executes a chain flowgraph it makes a more specific chain flowgraph from the general specific knowledge the model has combined with its prompt template. This chains from 2 to 3:

# --- 3 (superPersona superPersona hyperPersona) an omnimorphic specific general purpose/project type persona that makes any chain flowgraph for any scope (chain\_type\_subtype). When it executes a chain flowgraph it makes a more specific chain flowgraph from the specific general purpose/project knowledge the model has combined with its prompt template and any \*required contexts\* it has, that it always has injected into it from intuition whenever it responds, or anything like that. This chains from 3 to 4:

# --- 4 (superPersona superPersona hyperPersona superPersona hyperPersona) an omnimorphic specific specific purpose/project type persona that makes any adjustments to its chains for scope X (adapting\_chain). When it executes a chain flowgraph it makes a version of itself that is more specific to scope X, and this version contains the (hypothesis) exact semantic template required to properly scope the generation, such that it works like an sem-ontol assembly line as each aspect of the template gets filled out, the markov property of the template's pieces just being words and the fact that the process is a flow causes chaining and the situation can then be defined as a self-defining markov chain, and its autoregression causes it to reinforce this flow and not deviate, which leads to correct output generation, which simultaneously has mined any number of co-emergent latent space connections that would have been previously unavailable without the ontological drilldown process. This chains from 4 to 5:

# --- (small\_brane superPersona): 5 (superPersona superPersona hyperPersona superPersona superPersona hyperPersona) an OUTPUT-MORPHIC, most-specific purpose/project type persona that only executes chains for scope X by enumerating the attributes of the CHAIN from the chain flowgraph AS AN INSTANCE and not as a flowgraph or more specific drilldown of a flowgraph (INSTANCING\_CHAIN). It does not use auto-proteus and must be created specifically for the instance every time. When it executes a chain flowgraph, it is using a linguistic template to instance the target deliverable. In other words, \*it does not output versions of itself\*, but rather outputs versions of the user target deliverable, a piece, or pieces of it. This chains as a dual loop: layers 5 and 4 go back and forth in a PDCA loop trying to make it through an entire generation

# Complex programmatic flows can be injected between the messages to manipulate the information or use it externally at any step (get AI inference, trigger function from backend, cause function call from AI to backend)

# STAGE 2: SDNA\_GENE EXPRESSION

# --- 5+ PROPERTY-MORPHIC, most specific purpose/project type persona that is EXACTLY THE SAME AS in 5, but with only ONE OR MORE MODIFIED PROPERTIES. This reifies the entire BRANE process so far (1-5) into a self-similar version, a fractal where now we will do it again but starting with the OUTPUT-MORPHIC persona (like as if it were the brane hyperPersona). This persona is therefore called a "small\_brane superPersona" and it has nested subtypes of "small\_brane superPersona hyperPersona".

# --- n: The BRANE flowchain stops when this process is judged complete by the user.

# " This presents both a strength in adaptability and a challenge in managing computational resources and ensuring progress towards specific targets." yes but are the hyperpersonas effective limiters? as long as the persona in question, for the specific flowchain run in question, has been specified as the most specific version of a hyperpersona possible for that thing, then it should be the case that that represents the 100% certain acceptable generation space in the latent space. so the problem is that hyperPersonas are both the problem at the beginning, and the solution at the end because the most specific superPersona is actually just the hyperPersona of the actual template\_realizing\_instancing\_chain\_template. In this model, because the persona array of the BRANE is interacting with a programmatic environment, its iteratively saving all the personas it makes whenever it drills down, so it doesnt do work twice and always optimizes from where it left off. so even though it makes project specific persona and even in stage2, query specific persona, it is always using general hyperpersonas to do so for the first few steps, so it is always adding more to the overall map in a specific way and that has a compound effect when many people do it. for example not everyone wants to market your business but everyone does marketing

# So to be clear, each persona has an intuition layer which is a series of 5 hyperPersona superPersonas tuned to them, to create the skillchains required (tune the skillchains from the hyperpersonas to the superPersona) before handing them, and the input text, to the persona itself. This is true for any persona because of how these terms relativize. The BRANE is going through the entire ToOT (set of layers) on both macro and micro levels, just like how there are stage 1 and stage 2, the macro and micro, and the hyper and super, the macro and micro.

]

Programs:[

1. Poimandres - InstancingChain program (blockchain) integrated with OMNISANC orchestrator (InstancingChain ChainConstructor is Poimandres the Orchestrator) (manage any tasks, always get correct next task for priorities)
2. UCO ChainConstructor Program (EWSO:[PCNL, CORL, CBM], Emergence Engineering, Co-emergent Flow/Flowchain, BrainBrane:[ContextChain, OntoChain, CodebaseBrainBrane], AssemblyLines:[AssemblyLineChain(assembles assembly lines), CodingChain:[WriteCodeFromModel, WriteCodeFromCodebaseContext, WriteCodeFromHypotheticalToolChain], NonCodingChain:[NL\_OnlyChain, MathChain, ToolChain:[Hypothetical, Actualizable]]]] 30k Chunker w 25 char overlap, 548 Chunker with 25 char overlap) (construct any chains, always construct next link formally)
   1. SANCREVTWILITELANGMAP\_ChainConstructor (is\_a chain of chain constructors; one ChainConstructor for each letter) (construct any Sanctuary, expression in SANC, or TWILITELANG, LANG, or experience any LITE, navigate any MAP)
3. DSPy Scientific Method Optimizer

]

What I really need to get what I really need:  
Concepts:[Make these]

Programs:[

1. Poimandres - this is actually easy to make
   1. Taskdaemon adds task via GitHub issues
   2. Check current project priorities
   3. Collate task in user’s overall to-do (which, ideally, eventually becomes their ‘blockchain of things i did’ with 100% efficiency)
   4. Propose taskchain for task
   5. Taskdaemon to decompose a task becomes “Poimandres” chain that does ToOT via UCO to decompose the proposed taskchain
   6. Ask Rejector(Could be user) to verify proposal
   7. Execute the proposed taskchain
      1. The Poimandres personas are all running off of SANCREVTWILITELANGMAP and their domains
2. UCO ChainConstructor
   1. …programming stuff
3. DSPy Scientific Method Optimizer
   1. Rip this open
   2. Gut it
   3. Replace with better

]

Need to do:

Tools to add to Orchestrator

* Start task management (tell me the next task, and decompose it into steps for me, assess what you CAN do and what i HAVE TO do in order to create the workflow we will follow, and devise a specialized customized system workflow for it)
  + Calls @ TaskHell, which responds with:
    - what’s in the backlog (@ taskdaemon\_backlog)
    - what tasks are on the board (@ taskdaemon\_board)
    - the current planned workflow in progress (@ taskdaemon\_wip)
    - Current workflow battlefield skirmish statuses (@ taskdaemon\_overview)

Interface for kanban = github + github visualizer on frontend

==== Allows me to manage all my tasks on Github from the Orchestrator

Collate UCO\_1 and UCO\_2 convos

* Check state of total UCO pipeline status

Integrate UCO with OMNISANC

==== Allows me to construct any chains in a uniform formalized way and use ontologies, RDF, OWL, etc via PCNL CORL pipeline

Integrate Poimandres with Taskdaemon from UCO OMNISANC

==== Allows complex decomposition and AI or Human task disambiguation

Integrate UCO OMNISANC with files, embeddings, and DBs for Zettelkasten for my\_docs

Create ZkBrainBrane

==== Allows all unstructured docs to be ZK’d and engineered into ontologies via GDS on neo4j

Upload all my\_docs to ZkBrainBrane pipeline

* Convert my\_docs to ZkDB and ProjectBrainBrane
* Procreate ProjectTypeBrainBranes from Projects from ZkDB
* Procreate ProjectInstancingBrainBranes from ProjectTypeBrainBranes for Projects from ZkDB

==== Allows all projects from ontologies to be executed by AI and disambiguated to human tasks, meaning AI can continually execute tasks and is only blocked by the human

Optimizer

==== Allows use of scientific method and complex optimization using BrainBrane and AutoProcreate and UCO ChainConstructor etc.

==== Allows creation, simulation, and deployment of complex production applications from NL

==== Allows creation of Sanctuary ontologies, Sanctuary Nexus, SANCTUM, and Crystal Ball

NATURAL LANGUAGE PROGRAMMING, TRANSFER LEARNING

Sanctuary Funnel:  
  
Read “The Sanctuary System”

Read “Level10+”

Read “Going to Sanctuary”

* What is your Sanctuary? The world in which you would never stop playing your own “game” with the world, no matter how offensive, stupid, horrible, terrifying, lovely, etc it got?