Digitally Designed – Applying AI Agents to Digital Twin Development

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

ABSTRACT

Effective digital twin solutions require a suitable digital representation of the environment within which physical systems operate. This representation serves as the foundational layer, enabling the digital twin to mirror the real world with a fidelity that ensures the precise alignment of virtual and real-world elements essential for accurate simulations, analyses, and decision-making. From manufacturing to urban planning or defence, digital twin applications require a representation of the physical environment relevant to specific domain needs and challenges. This digital representation must address the particular context of the digital twin – a production line, city, or mission area – and support the digital twin's specific purpose – from operational and predictive analysis to testing and evaluation.

The design and construction of digital twin environments require significant expertise, effort, and data. This paper will consider how AI (Artificial Intelligence) agents can assist environment design and development processes within the context of digital twin solutions composed of platforms and systems requiring representations of terrain, features, entities, and behaviours that form a virtual physical and operational environment.

In just the last year, the utilisation of large language models (LLMs) as the "thinking brain" underpinning AI agents has made extraordinary progress. Remarkable advancements in LLM reasoning, understanding, creativity, task-specific knowledge, interaction, and multi-modal capabilities have led to the development of versatile LLM agent solutions that match and even exceed human performance, transforming many fields, from code development to content creation. This paper builds upon earlier LLM agent architecture research to explore the practical application of the latest state-of-the-art LLM capabilities to the digital twin design lifecycle. In particular, it will consider how agents can support, assist, and improve the human-in-the-loop design process by enhancing human expertise and performance with augmented, domain-specific LLM knowledge and how AI agents may apply this to orchestrate the development of digital twin environments.

ABOUT THE AUTHORS

Graham Long has worked in the training and simulation industry for over 30 years, with extensive experience leading and developing visual systems and synthetic environment solutions for civil and military programs and research projects. As product line architect and a synthetic environment specialist, he is currently focused on developing digital twin solutions that incorporate simulation and synthetic environment technologies as well as evaluating AI's potential application and integration within these systems

Digitally Designed – Applying AI Agents to Digital Twin Development

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

INTRODUCTION

From NASA's pioneering use of digital replicas of spacecraft for real-time simulation, troubleshooting, monitoring, and managing complex mission systems to Dr. Michael Grieves's first credited use of the term and application of digital twins to manufacturing, digital twins are rapidly evolving in capability and expanding to many diverse fields, products, and services. They have become a foundation of Industry 4.0, revolutionizing decision-making processes, product design, production optimization, and predictive maintenance in manufacturing, and are now finding applications in energy, construction, infrastructure, urban planning, and healthcare.

Digital twins are subject to many different definitions, descriptions, and interpretations. However, their essence lies in connecting a digital representation of a physical entity, system, system-of-systems, or process to its physical counterpart with a bi-directional, synchronized data exchange.

In the defense sector, digital twins are gaining rapid traction as indispensable tools for boosting capabilities in platform development and support throughout the lifecycle (Skinner, 2022). They can also be pivotal in concept development, experimentation, test and evaluation, and operational activities within today's intricate, multi-domain environment. Increasingly, modelling and simulation (M&S) solutions, including training and education, mission planning and execution, cybersecurity, weapon system development, and environmental and terrain modeling, leverage digital twins to provide realistic, dynamic, and interactive environments, enhancing operational readiness, strategy development, and decision-making.

Integrating artificial intelligence (AI) and machine learning (ML) into these digital twin and M&S systems further enhances their capabilities. AI facilitates more precise, real-time, and adaptive simulation of physical systems and processes and brings new and improved features such as enhanced predictive maintenance derived from proactive insights and anomaly detection, improved performance optimization through real-time analytics and process optimization, and advanced simulation and modeling that incorporate scenario analysis and adaptive modeling.

System interoperability, flexibility, and composability are well-established concepts in complex M&S systems. Over the last decade, they have led to the emergence of the modeling and simulation as a service (MSaaS) approach in response to the growing need for flexible, scalable, and interoperable simulation capabilities accessible on demand over the cloud. These concepts are equally important to complex digital twin solutions that can combine specific data models, digital shadows, digital twins, live data feeds, connected physical assets, data analytics, or simulation elements to deliver an optimum system solution. However, composing and orchestrating digital twins in this way and perhaps encapsulating this within a flexible, scalable, and accessible MSaaS model is not trivial. It requires considerable domain and system expertise to design the solution and a practical framework and service model to support system integration and composition.

Mitigating the complexity of this task can deliver significant benefits and is susceptible to the application of an AI agentic solution that can assist the human digital twin architect in the composition and orchestration of the digital twin system and its components. AI agents possess advanced and complex reasoning and problem-solving abilities. They excel at handling intricate, multi-step problems, offer innovative solutions, and consistently enhance their skills through learning and adaptation. Their capacity to understand context, nuances and collaborate with other systems makes them powerful tools for addressing diverse challenges and delivering results with speed, efficiency, and autonomy.

AGENTIC AI

The recent emergence, impact, and remarkable capabilities of generative AI have attracted much attention as the next significant step in AI evolution. Yet, the rapid progress of AI agents in only the last two years is proving even more revolutionary and is accelerating generative AI toward the goal of artificial general intelligence.

Renowned AI thought leaders are highlighting the transformative potential of AI agents to play a central role in enhancing creativity, productivity, and innovation and to shift how AI integrates into our lives and work: "What I am seeing with AI agents is an exciting trend that I believe everyone building AI should pay attention to." (Ng, 2024).

The definition of terms related to AI agents is still evolving, but an LLM agent can be described as an LLM-based entity capable of interacting with its environment, such as a computer, web browser, or other agents, using tools through API or function calls to achieve a specific goal, such as booking a flight, writing a business plan, or researching a topic. This agent operates with a high degree of autonomy over an extended period. It can break down a complex goal into smaller tasks and determine when it has been achieved with minimal human input. In straightforward terms, an agent can be summarized as: **Agent** = LLM + Memory + Planning skills + Tool Use (Weng, 2023) (Figure 1).

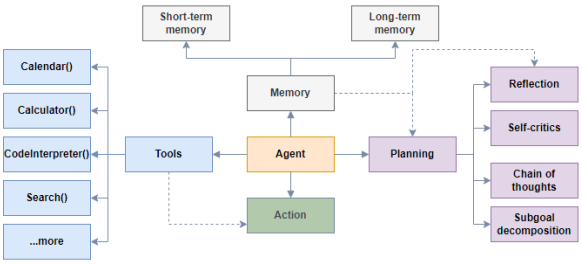

Figure 1 Overview of a LLM- agent system

The Role of LLMs

AI agents' core capability lies in using a large language model as the thinking or reasoning engine to dynamically decide and perform a non-deterministic sequence of actions based on the context (Chase, 2023). Large language models (LLMs) are trained on vast amounts of text data to understand and generate text resembling human communication and learn to predict and identify patterns in human language. With increased size and complexity, LLMs have developed sophisticated reasoning abilities and behaviors, enabling them to understand context, make inferences, and generate coherent, relevant responses that emulate human thought processes.

Despite these outstanding capabilities, LLMs can struggle with some of the simplest tasks that computers typically excel at – such as logic or calculations (Karpas et al., 2022). Although the latest generation of models, such as GPT-4o, now go beyond single-mode text and incorporate a multimodal understanding of imagery, video, and sound and employ web searches to access current data, LLM multimodal capabilities must be extended to provide agents the capacity for planning, memory, tool usage, and action.

AI Agentic Design Patterns

Many agentic design patterns are emerging, from basic question-and-answer (Q&A) interaction with an LLM to multi-agent solutions.

LLM Question & Answering (Q&A)

Most LLMs are used today in zero-shot mode, meaning a user gives a task to an LLM via a prompt, perhaps in a Q&A exchange, without providing the model with any prior examples or fine-tuning the task. The model relies on its pre-existing knowledge and general understanding to generate a response, leveraging its ability to generalize from its training data to new situations and attempting to perform the task in a single iteration without revising its work. Despite the challenges of this approach, LLMs perform very well using this method.

However, compared to this single-pass solution, applying an agentic workflow to the same task will yield superior results by directing the LLM to iterate over the task many times (DeepLearning.AI, 2024). For example, research has revealed that applying an iterative agent workflow to code generation using GPT3.5 significantly outperforms zero-shot prompting tasks that use a better LLM (GPT-4).

Planning
Although some agentic tasks may be straightforward enough to be accomplished in a single step or with a single tool, many will be too complex and require the agent to plan an approach to solving the problem. The ReAct (Reasoning and Acting) framework (Yao et al., 2023) and reflection are related concepts that significantly enhance AI systems' decision-making and problem-solving abilities and are critical to agentic planning. Reflection, the agent's ability to self-assess or evaluate its actions and decisions, plays a pivotal role in understanding its performance, learning from past experiences, and, most importantly, improving future responses. It is a critical component of the ReAct framework. ReAct integrates reasoning and acting in a single loop and is the most common algorithm employed to enhance AI's ability to handle complex, dynamic environments by continuous learning and performance improvement through an iterative process of reasoning, acting, and reflecting. Combined with an appropriate prompting strategy, this enables the agent to process the context of a problem, make logical deductions to decompose the problem into actionable sub-tasks, plan the workflow, select tools, dynamically execute the process, and return results.

Tools
Even with the advent of multimodal capabilities, LLMs are extremely limited in the range of actions they can take. They need access to a potentially vast range of tools, computation, and external data sources to become an effective, versatile, intelligent agent capable of manipulating their environment. These are achievable by enabling an LLM to use APIs or request external functions to gather information, take actions, or manipulate data. Since GPT-4 introduced function-calling capabilities in mid-2023, many other models are developing more straightforward tool-calling capabilities, leading to a significant increase in the use of agentic tools. LLMs use function calling to execute or suggest specific functions or actions based on the text input. By equipping an LLM with available external functions, which might include a text description of what the function does and details of what arguments the function expects, the LLM analyzes the prompt to automatically choose the relevant function to call to do a job.

Memory
Through in-context learning, agents can be given short-term memory capabilities. Alternatively, they can store and retrieve longer-term information in an external vector store, which serves as long-term memory. This feature allows agents to store information, learn from past interactions and feedback, and improve their actions.

Single & multi-agent architectures
Agentic systems are composed from either single or multi-agent architectures. Single-agent architectures are systems that use a single AI entity to organize and complete tasks from start to finish. Because they are not susceptible to the influence of interactions with other non-grounded agent team members that may introduce poor feedback or irrelevant discussions, single agents may be more appropriate for simpler, centralized tasks in contexts requiring greater accuracy, direct control, ease of integration, and resource efficiency.

On the other hand, multi-agent architectures encompass multiple AI entities that cooperate, share information, and make collaborative decisions. This organizational framework excels in addressing complex scenarios that demand a range of expertise from multiple agent stakeholders and roles, cooperative problem-solving, varied perspectives, and scalable systems requiring specialization, parallel processing, and robustness (Masterman, Besen, Sawtell, & Chao, 2024).

APPLICATION OF AGENTIC AI TO DIGITAL TWINS

Digital twin solutions are complex but remarkably versatile. They find applications across a wide range of fields and use cases but typically share a set of common high-level capabilities that include data services, integration, intelligence, user experience, management, and trustworthiness (DTC, 2024). These capabilities are generally delivered within systems employing a layered architectural pattern that provides a platform for the composition and orchestration of digital twin solutions fitted to specific use case needs.

System composition is a complex, challenging, multi-step activity that has the potential to benefit from the assistance of an AI agent who can work alongside a human system architect by leveraging existing systems, tools, components, workflows, specific domain knowledge bases, and expertise. However, given agentic AI's rapid technological evolution and relative immaturity, it is crucial to establish realistic expectations regarding its potential and constraints. Encapsulating the entire digital twin composition task within an agentic solution is a promising and exciting but challenging first step. Therefore, a progressive implementation from a proof of concept developed around

a narrower aspect of the digital twin solution allows for an incremental approach that identifies potential issues in a controlled environment, assuring a smoother transition to the larger-scale digital twin composition task.

Use Case

The proof of concept will focus on an agentic solution to the design, development, composition, and integration of a virtual environment (VE), a significant component of most digital twins. Many modeling and simulation applications of digital twin solutions require a virtual representation of a physical environment - this may provide a digital model of the physical environment as a component of the digital twin solution or be the actual digital twin itself. The virtual environment must provide a replica of its physical counterpart with features and content represented with appropriate accuracy, detail, correlation, fidelity, and properties to support the required interaction between physical and digital systems, their state, periodic synchronization, and bi-directional data flows.

This use case requires a virtual environment for an air operating environment digital twin populated with multi-domain live, virtual, and constructive entities. The environment must accurately represent the physical air operational environment in support of 2D and 3D visualization and analysis at multiple resolutions and scales and support the needs of digital shadows, digital twins, simulated systems, such as electro-optical sensors, constructive or AI entities, and other system components. This multi-resolution environment will incorporate seamless global, regional, and local scale representations at increasing terrain and feature detail levels with the highest fidelity representation of specific and geographically accurate features at the local scale. The global and regional scale representations will provide a foundational level of content, with high-fidelity local areas added to reflect the specific scenarios to be supported by the digital twin.

Importantly, the high fidelity local areas must be generated and integrated in a flexible and timely manner in response to on-demand, changing operational needs and scenarios. If the generation of these virtual environments can be accomplished with an agentic workflow, it will increase the speed, flexibility and responsiveness with which these areas can be generated, reduce the dependency on specialized development skills, tools and teams and empower the user with the capability to dynamically adapt the synthetic air operating environment within the digital twin.

VE AGENTIC ARCHITECTURE

The stretch target objective of this agent is complex. It will need to handle an elaborate set of tasks, including receiving a high-level description of a scenario, eliciting the requirements for a virtual environment to support the scenario, translating them into a design, and then performing the necessary data acquisition, data processing, content creation, assembly, and generation of the completed virtual environment. A key challenge from the outset is to determine whether all of these tasks can be successfully encapsulated within one agentic solution or should be split into multiple solutions.

Selecting and implementing the optimum agentic design pattern must consider the nature, complexity, and diversity of tasks, the capabilities and tools associated with these activities, the required level of collaboration between agents and with a human, and the expected outcomes.

The virtual environment development process is analogous to a software development lifecycle that encompasses requirements gathering to code development, testing, and quality assurance. Recent research, from CAMEL (Li et al., 2023b) to ChatDev (Qian et al., 2023), MetaGPT (Hong et al., 2023), and Agentcoder (Huang et al., 2023a), demonstrate the potential of applying collaborative multi-agent systems to complex software engineering challenges, bringing significant changes in software engineering practices. These systems are poised to accelerate software development, foster innovation, and deliver critical benefits.

Software engineering agentic design patterns offer a valuable template for virtual environment development solutions, but critical differences exist in their aims and processes. While agentic code generation focuses solely on coding skills, virtual environment development is more diverse, covering skills from data processing and content generation to constructing runtime environments for client systems. These will require an agentic design combining appropriate engineering practices with agents using multifarious external tools.

Agent roles / personas and SOPs

Multi-agent systems tailor each agent's traits, actions, and skills to achieve specific goals and assign them unique roles with particular characteristics, capabilities, behaviors, and constraints. Virtual environment development lacks the standardized engineering methodology of software engineering to help identify potential agent roles. However, the Reuse and Interoperation of Environmental Data and Processes (RIEDP) product formalizes a high-level M&S environment creation process, captured in a Reference Process Model (RPM) whose lifecycle stages offer a basis for a standardized operating procedures (SOPs) workflow and agent roles for virtual environment development. The proposed agent roles, tasks, tools and outputs are summarised in Table 1.

RIEDP RPM	Agent	Role	Task	Tools	Output
• Define Reqmnts	Requirements Analyst	VE Reqs Expert	receives DT scenario as input, translates scenario into VE specification	Search, retrieval	VE reqs specification
• Define Reqmnts • Collect Source Data	Architect	Req/design/ process expert	receives VE specification as input, requests data analyst to identify the data sources that can satisfy the		VE design specification
• Define Reqmnts • Collect Source Data	Data Analyst	VE data expert	receives VE specification as input, identifies the data sources that can satisfy the VE specification & provide data report indicating data sources, availability, data gaps, state, suitability, processing required.	search, retrieval, data analysis	Data report
• Clean Source Data • Align Source Layers • Intensify Bline Data • Specialize Data for Target Applications	Data Processor	Data processing expert	data report & design spec as input, identifies data processing required & tools	geospatial data processors	Processed data
• Create/Modify Library Data	Content Developer	Content Creation	data report & design spec as input, identifies content creation required & tools	3D/genai/NERF	Content
• Establish Baseline Data	Assembler	SE data expert	Assemble, organise, format, structure data into well-formed state that supports VE construction & development	RAG, file handling, code generation	Well-formed data
• Generate RT Target Databases	Env Developer	SE development expert	Build the VE in the format(s) required by the VE client systems	RAG, code generation, code execution	Build instructions

Table 1 VE Agent Roles & Capabilities

Multi-Agent Architecture

The proposed architecture (Figure 2) utilizes a graph representation, with each node in the graph serving as an agent and edges representing their connections. The graph, acting as a control flow manager, ensures the system is well-organized and controlled, with edges facilitating communication and contributing to the graph's state. Agents are arranged into subgraphs, and top-level and mid-level supervisors supervise sub-teams of agents working on particular aspects of the task. The agent supervisors are responsible for assigning tasks to individual agents, effectively making the supervisor an agent who uses other agents as tools. The Multi-agent design allows for the division of complex problems into manageable units of work, which specialized agents can target with their own prompt, LLM, and tools to yield better results. Agents adopt a cooperative communication paradigm with agents collaborating towards a shared goal, primarily interacting within their layer or with adjacent layers, based on their roles and responsibilities (Liu et al., 2023). The data exchanged will mainly be in text, including structured specifications, documents, and code. Virtual environment (VE) agents will continuously learn and develop by acquiring capabilities through feedback from their environment and interactions with other agents and humans.

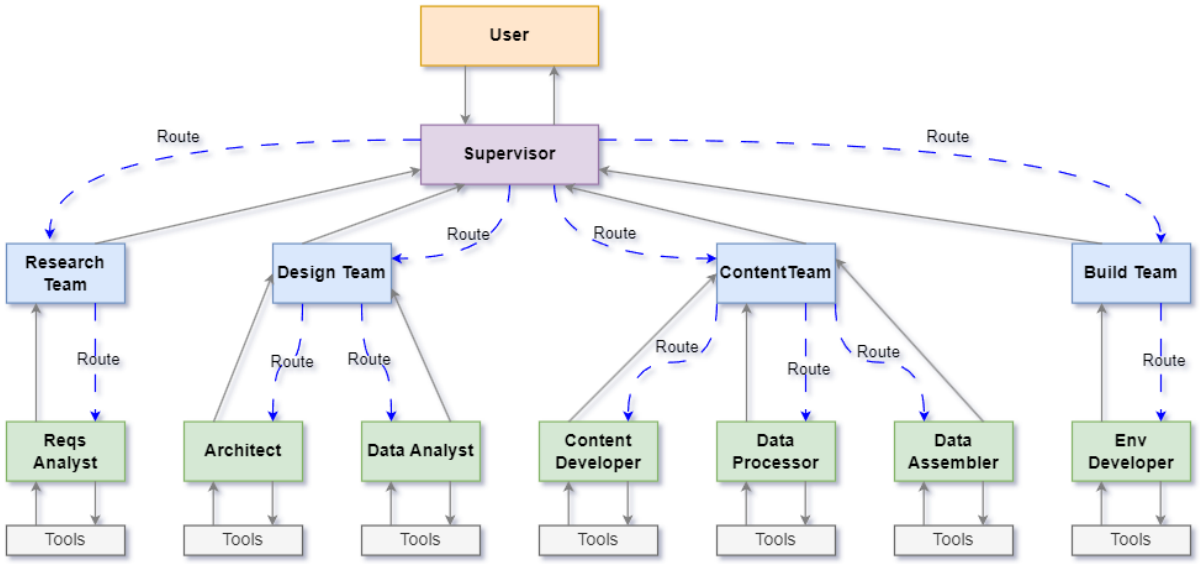

Figure 2 VE Multi-Agent Architecture

Agent Capabilities - Requirements Analyst

The requirements analyst performs the critical role within the VE multi-agent solution. It is responsible for understanding and translating the initial user request into a suitable specification of requirements that will guide the subsequent VE design and development activities. This task drives the successful delivery of the virtual environment and supports the primary goal of making the requirements task more accessible by reducing the need for domain and virtual environment expertise.

The agent must be able to derive the VE requirements specifications from a description of the digital twin scenario and any other appropriate information provided by the user. One option to perform this task is to use a pre-trained LLM for question-answering on a user query and generating a response based on the LLM's pre-trained knowledge and understanding. This approach is straightforward to implement, creates quick responses, and leverages the model's learned contextual understanding. However, the LLM's knowledge is constrained to the scope of the training data, which can cause the LLM to hallucinate and generate plausible but incorrect answers, compromising and limiting the quality and accuracy of its results. These limitations are not suitable for a virtual environment analyst, which needs accurate, reliable, and quality responses and a deep understanding of diverse information sources covering virtual environment design, use case examples, data requirements, dependencies, domain knowledge, design, data, and synthetic environment standards.

LLM Knowledge Augmentation

Improved, more accurate, reliable, context-aware, and up-to-date responses are achievable using a broad spectrum of advanced techniques, progressing from a traditional LLM with Q&A to complex agentic solutions (Figure 3).

As a first step, the requirements analyst agent can be configured with an LLM (e.g., GPT-4o (OpenAI, 2023a) or Ollama with a search tool like Tavily) with access to external knowledge from a web search capability. The agent receives a scenario describing the geographic location and operational mission (a UAS surveillance of an airfield). The task description directs the agent to provide a detailed analysis and environment specification to support the scenario and create requirements and preliminary design (Figure 4). These must include terrain layers (elevation, imagery, features, 3D models), geographic points of interest, recommended design criteria, and justifications for these choices. The human user and agent can chat to refine the response, which is required to provide a design specification that adopts data models from synthetic environment design standards like RIEDP or the common database (CDB), output in JSON format (RIEDP - Figure 5), (CDB - Figure 6).

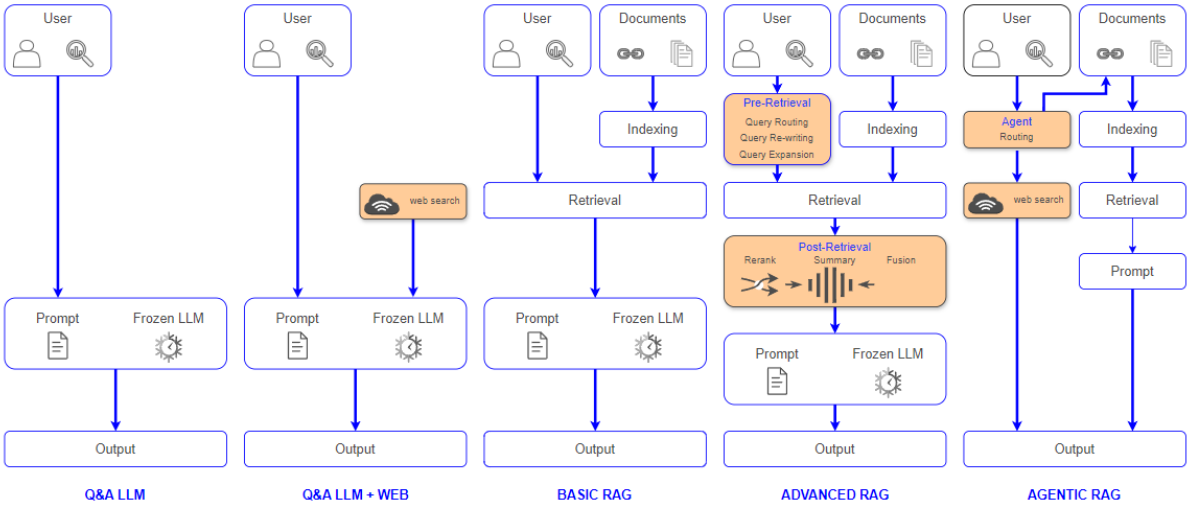

Figure 3 Evolution from LLM to Agentic RAG

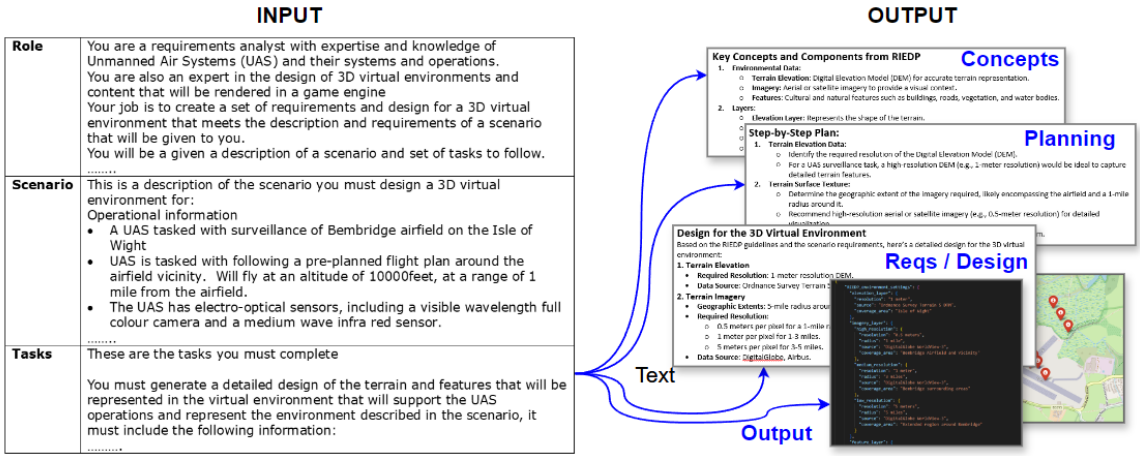

Figure 4 Requirement Analyst – Agent Inputs & Outputs

```
{
  "RIEDP_environment_settings": {
    "elevation_layer": {
      "resolution": "1 meter",
      "source": "Ordnance Survey Terrain 5 DTM",
      "coverage_area": "Isle of Wight"
    },
    "imagery_layer": {
      "high_resolution": {
        "resolution": "0.5 meters",
        "radius": "1 mile",
        "source": "DigitalGlobe WorldView-3",
        "coverage_area": "Bembridge Airfield and vicinity"
      },
      "medium_resolution": {
        "resolution": "1 meter",
        "radius": "3 miles",
        "source": "DigitalGlobe WorldView-3",
        "coverage_area": "Bembridge surrounding areas"
      },
```

Figure 5 VE Specification - RIEDP

```
{
  "OGC_CDB_environment_settings": {
    "elevation_dataset": {
      "resolution": "1 meter",
      "source": "Ordnance Survey Terrain 5 DTM",
      "coverage_area": "Isle of Wight"
    },
    "imagery_datasets": [
      {
        "resolution": "0.5 meters",
        "radius": "1 mile",
        "source": "DigitalGlobe WorldView-3",
        "coverage_area": "Bembridge Airfield and vicinity"
      },
      {
        "resolution": "1 meter",
        "radius": "3 miles",
        "source": "DigitalGlobe WorldView-3",
        "coverage_area": "Bembridge surrounding areas"
      }
```

Figure 6 VE Specification - CDB

The information the agent is required to gather, report, and incorporate into the specification is adaptable by refining the details of the prompt and the tools the agent has access to. Progressive refinement and addition of more prompt detail was applied to significantly expand the agent research and specification tasks. These now include the agent's justification for its selection of terrain data types (Figure 7), recommended coverages of data, feature inclusion, and the generation of maps of the environment. LLMs do not produce deterministic results, which means that the same input will yield different results each time the task is executed. However, the objective, factual content, such as

recommended terrain resolution, is consistent, and the textual differences in the responses are mitigated by outputting the response in the form of a specific data model in a structured JSON format.

This approach is demonstrably capable of generating plausible results but the quality and effectiveness of the responses is very dependent on the capabilities of the chosen LLM, the knowledge sources it has access to and the engineering of the prompt that directs the model. These aspects require further experimentation, refinement, and tuning to optimize the outputs, particularly given the ongoing rapid evolution and improvements of LLMs.

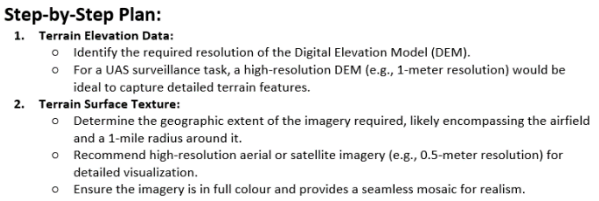

Figure 7 Specification Criteria Justification

Retrieval-Augmented Generation (RAG)

Although web search adds significant value and capability to an agent, it still constrains the agent's access to data and is susceptible to hallucinations or simply incorporating inaccurate, unreliable, or unknown data into its responses. In response to these limitations, the RAG framework has emerged as the most prominent framework for building LLM-based applications across diverse domains and use cases (Fatehkia, Lucas, & Chawla, 2024). Its unique capability to access enterprise data and custom knowledge bases of curated specialized data and web information, if needed, is impactful, significantly reducing hallucinations and increasing the factual accuracy of the generated content. It can handle specialized queries and integrate up-to-date information without requiring frequent model retraining.

Although the agent performance using web search produces plausible results, in the context of a virtual environment agent, RAG will enable an agent to deliver superior results by providing access to highly specific, precise, controlled and curated data sources such as sensitive enterprise data, regulations, enterprise standards, proprietary or sensitive requirements, design and domain information, or any controlled data source that is considered relevant. For digital twins that integrate physical and cyber systems, using RAG to improve the reliability and accuracy of the virtual environment may be critical to the successful operation of the system.

Basic RAG

A basic RAG workflow (Figure 3) can be divided into indexing, retrieval, and generation stages. First, raw data is cleaned and extracted before file formats (such as PDF or HTML) are converted into standardized plain text, which is then divided into smaller, more manageable chunks. This text is converted into embeddings during the indexing phase, which are then stored in a vector database to create a searchable index. The user's query is also converted into an embedding in the retrieval step. This embedding is used to conduct a similarity search on the vector database for the most relevant text data. Finally, in the generation step, the query is augmented with the additional context retrieved earlier, and a LLM uses this enhanced prompt to generate an answer to the user's question.

Advanced RAG

Despite delivering enhanced capabilities like chatting over specific data, basic RAG solutions can still face performance challenges. Inadequate retrieval processes that lead to low precision, low recall, insufficient context, and use of outdated information can still cause poor-quality response generation (Fajardo, 2024). These issues have prompted the evolution of more advanced RAG techniques (Figure 3) applied throughout the entire RAG pipeline, from data to embeddings, retrieval, and response synthesis. These begin with strategies to improve retrieval of documents most relevant to a user query by optimizing chunking, structuring knowledge, employing knowledge graphs, adding metadata, mixing keyword and dense/sparse retrieval, and search. Other techniques aim to enhance the generation of suitable responses by compressing information to reduce noise or re-ranking the results. More complex techniques fundamentally adapt the RAG pipeline, by fine-tuning the generator or the embedding model to domain-specific data or adding agentic capabilities to RAG.

Agentic RAG

Agentic RAG is a promising capability that offers a scalable, versatile approach that makes RAG available to agents as a tool. It integrates agents and reasoning with RAG-created responses from curated knowledge sources to fulfill particular agentic tasks, implemented in various architectural patterns.

One practical agentic RAG architecture uses the agent as an adaptive router that decides whether to direct questions to knowledge sources in a vector store or, if the initial retrieval from the vector store is irrelevant, conduct a web search based on the question's content (Langchain, 2024), (Jeong et al., 2024), (Yan et al., 2024), (Asai et al., 2023). The system also verifies the accuracy and relevance of the generated responses and makes corrections if necessary. The system comprises a component that retrieves documents relevant to the question, a grader that evaluates the relevance of the retrieved documents, a response generator, and a component to conduct a web search if the initial retrieval is insufficient. Instead of a dynamic agent making decisions at each step, the solution adopts a graph-based approach with a pre-defined control flow to enhance reliability and conditional edges that ensure decisions are made based on the state of the data, such as whether to perform a web search or generate a response.

Agent augmentation with specialized, expert knowledge sources is essential to overcome the limitations imposed by reliance on the scope of an LLM's corpus of training data. RAG has been widely implemented as a very effective method of providing agents access to relevant data sources and improving agent responses. It provides the virtual environment agentic solution with significant enhancements and critically essential capability. It supports the curation of comprehensive specific knowledge sources drawn from multiple documents and information, providing precise, contextually relevant information. It offers expert-level responses, thorough topic coverage, and responses enriched with detailed information from authoritative sources. Facts can be cross-referenced to multiple sources, reducing the likelihood of incorrect or misleading information. Importantly, for digital twin solutions that require accurate, reliable, robust, high-fidelity virtual environments, RAG enables insights into retrieval decisions, attribution of references and sources, real-time access, integration, and retrieval of the latest information. Moreover, it provides essential explainability, transparency, and trust in the agentic reasoning, problem-solving, and response process.

Furthermore, RAG not only supports the needs of the requirement analyst but can be incorporated with any agent in the solution, such as the data processor, to augment it with specialized knowledge and expert information to support its role and successful task completion.

Agent Capabilities - Architect

The architect, an expert in VE design, plays a pivotal role in the task's success. It is responsible for generating a VE design specification that accurately reflects the requirements analyst's VE requirements and the data analysis results (captured in the data report). This specification, output in a standard format, such as a RIEDP or CDB-compliant data structure, guides the VE agent in planning any required data processing or content creation tasks.

Agent Capabilities - Data Analyst

The data analyst identifies the data that will satisfy the VE design specification and provides a data report indicating data sources, availability, data gaps, state, suitability, and any processing required, and provides this in a data report.

Agent Capabilities - Data Processor

The data processing agent conducts any data processing identified in the data report by selecting the appropriate tool from a set of geospatial processing tools. These tasks may vary from applying a particular data model or schema to changing the coordinate reference system or modifying the tile structure, but the agent will have the capability to perform any of the most likely data processing tasks required.

Agent Capabilities - Content Developer

The content developer will generate the required content. Tasks may include adapting existing content, such as 3D models, to conform to the design specification or generating 3D models from sources like point cloud data or imagery. This is a challenging agent role that will require tools that can direct applications like 3D authoring tools or even AI processes, such as NeRF-generated 3D objects. The agent will access these tools via APIs or code generated by the agent**.**

Agent Capabilities - Data Assembler

The assembler agent gathers and organizes the data (Figure 8) and content into a well-formed data set that provides the source and basis for environment generation for the target client systems. This data set may be a designated environment data standard like CDB or One World Terrain (OWT).

```python
import os
import shutil

def create_cdb_structure(base_path):
    # Define the base structure for CDB
    layers = {
        "ElevationTiles": "elevation_tiles",
        "ImageryTiles": "imagery_tiles",
        "FeatureData": {
            "CulturalFeatures": {
                "AirportBuilding": "cultural_features/airport_building",
                "Infrastructure": "cultural_features/infrastructure",
                "SurroundingStructures": "cultural_features/surrounding_structures"
            },
            "NaturalFeatures": {
                "Vegetation": "natural_features/vegetation",
                "Hydrography": "natural_features/hydrography",
                "Relief": "natural_features/relief"
```

Figure 8 Data Set Assembly Snippet

```python
import unreal

# Define file paths for your data sources
elevation_file_path = "/path/to/your/elevation_file.raw"  # High-resolution DEM file path
satellite_image_path = "/path/to/your/satellite_image.png"  # High-resolution satellite imagery file path

# Function to create a new landscape
def create_landscape(elevation_file, satellite_image):
    # Load the elevation data
    heightmap_import_data = unreal.LandscapeImportHeightmap()
    heightmap_import_data.file_path = elevation_file
    heightmap_import_data.width = 4033  # Adjust to your DEM resolution
    heightmap_import_data.height = 4033  # Adjust to your DEM resolution

    # Create landscape
    landscape_actor = unreal.EditorLevelLibrary.spawn_actor_from_class(unreal.Landscape, unreal.Vector(0, 0, 0))
    landscape = landscape_actor.get_component_by_class(unreal.LandscapeComponent)

    # Import heightmap to landscape
    landscape.import_heightmap(heightmap_import_data)

    # Apply satellite imagery as a texture
    texture_import_task = unreal.AssetImportTask()
    texture_import_task.filename = satellite_image
```

Figure 9 VE Construction Code Snippet

Agent Capabilities – Environment Developer

Finally, the environment developer constructs the environment for the client systems that will use it. Depending on the specific client system and its related processes for consuming the data and generating the environment, this stage may take different forms and capabilities. In the case of this prototype, the end target is Unreal Engine, and the agent creates Python code (Figure 9) that creates terrain in Unreal Engine from the data set provided by the assembler.

MULTI-AGENT SYSTEM CHALLENGES & ISSUES

Multi-agent solutions have made very rapid and significant progress over the last two years, but this pace of evolution has created a dynamic, changing environment in which there remain many options, issues, and challenges to consider. In general, the emergence of several development frameworks, like Langchain, Langgraph, AutoGen studio, CrewAI, and others, has eased the development of multi-agent solutions and made their exploitation more feasible and accessible. However, designing a multi-agent solution remains a complex task requiring careful consideration of several critical design issues. The proposed VE architectural pattern must be aligned with these issues to ensure the system operates efficiently, accurately, and securely.

The LLM is the core of this agentic approach and is pivotal in the system's design and operation. Each model has its strengths, and the choice depends on specific use case needs regarding task specialization, performance, data privacy, security, scalability, cost, and application requirements. LLMs should be selected based on their specific strengths and the nature of the tasks they are optimized for (e.g., some LLMs might be better at summarization, while others excel at code generation). One of the critical benefits of multi-agent solutions is the flexibility they offer. Each agent can use a separate LLM for specific tasks, providing a tailored approach to problem-solving. This flexibility instils confidence in the adaptability of multi-agent solutions, as these models can be upgraded or replaced with alternatives if the chosen models have compatible model interfaces and use standard protocols. This flexibility extends to selecting models that run locally (on-premises) or over the internet (cloud-based). Local Models, such as LLaMA 3 by Meta, Falcon 180B by Technology Innovation Institute, or BLOOM by Hugging Face, are best for scenarios needing strict data control, privacy, low latency, and customization. Cloud Models are ideal for scalability, accessibility, and integration with other cloud services. Examples include GPT-4 by OpenAI, Claude 3 by Anthropic, and Google Gemini. Balancing the initial and ongoing infrastructure, resource costs, deployment, and management of local models compared to understanding the ongoing operational cost of complex pay-as-you-go pricing models, reduced capital expenditure, and managed services of cloud models is a further consideration. Performance is another potentially significant criterion that may drive LLM selection toward local models offering low latency, on-premises data processing, and control of computational resources compared to the accessible, scalable, and flexible features of cloud solutions. Finally, data privacy and security issues may be decisive if the security risks and data privacy compliance challenges of transmitting data to cloud models are prohibitive or if the complete control of sensitive data, models, and privacy from local LLMs is required.

Hallucinations remain a significant challenge in LLMs and even more so in multi-agent systems where their interconnected nature can propagate misinformation. It is crucial to detect and mitigate hallucinations by individual agents and manage the flow of information between agents to prevent the spread of inaccuracies. Multi-agent systems also face scalability challenges due to the computational complexity of individual LLM-based agents. As agents increase, resource requirements grow, and effective coordination and communication become more complex. Developing advanced agent orchestration methodologies, defining scaling laws, and finding innovative solutions to enable collective multi-agent learning is crucial for optimizing the potential collective intelligence of multi-agent systems.

Bringing all of these design and architectural considerations together into a coherent, effective multi-agent solution requires the application of an objective evaluation methodology that can verify performance. Evaluating multi-agent systems is challenging because research focuses on individual agent evaluation rather than broader behaviours. Additionally, the lack of comprehensive benchmarks across various research domains impedes accurate assessment and measurement of the full capabilities of multi-agent systems.

CONCLUSIONS

The proposed VE architecture is a starting point for further experimentation, evaluation, and refinement. It is not just sufficiently modular but highly versatile to support the adaptation of agent roles, chosen LLMs, tools, and overall system capability. Evaluating the performance of alternative LLMs over the wide variety of tasks the multi-agent system must accomplish, balanced against considerations of security, data privacy, cost, and scalability, is a critical area for future research. However, given the need to equip the system with access to a range of specific domain information and technical knowledge, which is highly likely to contain sensitive data, a local LLM RAG solution with access to private data combined with controlled access to web-based or other external sources emerges as one of the most suitable architectures to deliver this. Much multi-agent research focuses on optimizing inter-agent communication - the proposed VE architecture adopts a graph-based approach that offers a robust framework for managing and optimizing the interactions between multiple agents. One key benefit of multi-agent solutions is their divide-and-conquer approach, which focuses expertise on specific agents and allows them to collaborate to solve complex problems. Nevertheless, as the VE agent capabilities and system sophistication evolve, performance and scalability factors may dictate reconfiguring some aspects into separate multi-agent solutions.

This early-stage proof-of-concept multi-agent solution initially focuses on the critical requirements and design phases as the foundation of the entire VE lifecycle. Still, the architecture demonstrates the potential capabilities agentic systems' complex reasoning and problem-solving abilities can deliver for a virtual environment development use case. More importantly, it underscores the transformative impact of scaling up agentic solutions on the digital twin composition task and the broader engineering lifecycle. The rapid evolution and deployment of agentic solutions are underway in software development, content creation, and other fields. They are already enhancing creativity, productivity, and innovation, fundamentally changing how AI integrates into our lives and work.

REFERENCES

Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2023). SELF-RAG: Learning to retrieve, generate, and critique through self-reflection. Retrieved from: https://arxiv.org/abs/2310.11511v1

Chase, H. (2023). Harrison Chase - Agents Masterclass from LangChain Founder (LLM Bootcamp) Retreived From: https://youtu.be/DWUdGhRrv2c?si=ZyKyRbmWxXCJGyRy

Capabilities Periodic Table - Digital Twin Consortium. Retrieved from: https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/

DeepLeaning.AI, (2024). Four AI agent strategies that improve GPT-4 and GPT-3.5 performance. Retrieved From: https://www.deeplearning.ai/the-batch/how-agents-can-improve-llm-performance/

Fajardo, A, (2024) A Cheat Sheet and Some Recipes For Building Advanced RAG. Retrieved From: https://medium.com/llamaindex-blog/a-cheat-sheet-and-some-recipes-for-building-advanced-rag-803a9d94c41b

Fatehkia, M., Lucas, J. K., & Chawla, S. (2024). T-RAG: Lessons from the LLM trenches. Retrieved from: https://arxiv.org/abs/2402.07483v2

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., & Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. Retrieved from:. https://arxiv.org/abs/2402.01680

He, J., Treude, C., & Lo, D. (2024). LLM-based multi-agent systems for software engineering: Vision and the road ahead. Retrieved from:. https://arxiv.org/abs/2404.04834v1

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., & Schmidhuber, J. (2023). METAGPT: Meta programming for a multi-agent collaborative framework. . Retrieved from: https://arxiv.org/abs/2308.00352v5

Jeong, S., Baek, J., Cho, S., Hwang, S. J., & Park, J. C. (2024). Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. Retrieved from: https://arxiv.org/abs/2403.14403v2

Karpas E, Abend O, Belinkov Y, Lenz B, Lieber O,Ratner N, Shoham Y, Bata H, Levine Y, Leyton-Brown K, Muhlgay D, Noam, Schwartz R, Gal, Shai S, Shwartz S, Amnon Shashua A, Tenenholtz M. (2022). MRKL Systems. Retrieved from: https://arxiv.org/abs/2205.00445v1

Langchain (2024). Local RAG agent with LLaMA3. Retrieved from: https://github.com/langchain-ai/langgraph/blob/main/examples/rag/langgraph_rag_agent_llama3_local.ipynb

Llamaindex (2024). RAG in 2024:advancing to agents ..Retrieved from: https://slides.com/seldo/rag-and-agents

Li, G., Hammoud, H. A. A. K., Itani, H., & Khizbullin, D. (2023). CAMEL: Communicative agents for "mind" exploration of large language model society. Retrieved from: . https://arxiv.org/abs/2303.17760v2

Masterman, T., Besen, S., Sawtell, M., & Chao, A. (2024). The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey. Retrieved from: https://arxiv.org/abs/2404.11584v1

Ng, A. (2024) Andrew Ng's Vision for AI's Future: Unlocking Agentic Workflows. Retrieved From: https://nextbrain.ai/blog/andrew-ngs-vision-for-ais-future-unlocking-agentic-workflows

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., & Sun, M. (2024). ChatDev: Communicative agents for software development. Retrieved from: https://arxiv.org/abs/2307.07924v5

Skinner, S. G. (2022). Taking simulation interoperability standards to the next level with digital twins. Retrieved from: https://www.sto.nato.int/publications/STO%20Meeting%20Proceedings/STO-MP-MSG-197

Weng, L. (2023, June). LLM-powered autonomous agents. Lil'Log. Retrieved from: https://lilianweng.github.io/posts/2023-06-23-agent/

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A., White, R. W., Burger, D., & Wang, C. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. ArXiv. Retrieved from: https://arxiv.org/abs/2308.08155v2

Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective retrieval augmented generation. Retrieved from:. https://arxiv.org/abs/2401.15884v2

Yang Z, Li L, Wang J, Lin K, Azarnasab E, Ahmed F, Liu Z, Liu C, Zeng M, Wang L. (2023). MM-REACT : Prompting ChatGPT for Multimodal Reasoning and Action. Retrieved from: https://arxiv.org/abs/2303.11381v1

Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y. (2023). REACT: Synergizing Reasoning And Acting In Language Models. Retrieved from: https://arxiv.org/abs/2210.03629v3

Zala, A., Cho, J., Lin, H., Yoon, J., & Bansal, M. (2024). EnvGen: Generating and adapting environments via LLMs for training embodied agents. Retrieved from: https://arxiv.org/abs/2403.12014v1