

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Rodriguez, Roberto; Syynimaa, Nestori

Title: Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-life Problems : Microsoft Entra ID Administration Agent (MEAN)

Year: 2024

Version: Published version

Copyright: © 2024 the Authors

Rights: CC BY-NC-ND 4.0

Rights url: <https://creativecommons.org/licenses/by-nc-nd/4.0/>

Please cite the original version:

Rodriguez, R., & Syynimaa, N. (2024). Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-life Problems : Microsoft Entra ID Administration Agent (MEAN). In J. Filipe, M. Śmiątek, A. Brodsky, & S. Hammoudi (Eds.), ICEIS 2024 : Proceedings of the 26th International Conference on Enterprise Information Systems, Volume 1 (pp. 881-887). SCITEPRESS - Science and Technology Publications. <https://doi.org/10.5220/0012735700003690>

Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-life Problems

MEAN: Microsoft Entra ID Administration Agent

Roberto Rodriguez¹ ^a, Nestori Syynimaa^{1,2} ^b

¹Microsoft Security Research (MSecR)

²Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
{rrodri, nsyynimaa}@microsoft.com

Keywords: LLM, autonomous agent, ChatGPT, PowerShell, Entra ID.

Abstract: Microsoft Entra ID is Microsoft’s identity and access management solution used by many public and private sector organisations globally. In March 2023, Microsoft retired two PowerShell modules which have enabled automation of administrative tasks, such as user management. The replacement module is based on Microsoft Graph API, and its effective usage would require administrators to learn software development skills. In this paper, we will report the results of work-in-progress research on exploring the applicability of LLM-powered autonomous agents to solve real-life problems. We describe the design and proof-of-concept implementation of MEAN, an agent that performs Entra ID administrative tasks using Microsoft Graph API based on natural language prompts. The results show that LLM-powered autonomous agents can perform at least simple Entra ID administrative tasks. This indicates that the agents could ease the administrative burden by removing the need to learn software development skills.

1 INTRODUCTION

1.1 Microsoft Entra ID


Microsoft Entra ID (MEID) is Microsoft’s cloud-based identity and access management (IAM) solution (Microsoft, 2023c), formerly known as Azure Active Directory (Azure AD). MEID is widely used globally in public and private sector organisations (see Syynimaa, 2022).


Administrators can administer Entra ID via web-based interfaces like the Entra ID portal. The portals are suitable for changing Entra ID-wide settings and performing manual tasks involving a small number of individual targets, such as users or groups. However, portals do not scale, which forces administrators to rely on automation solutions. For instance, administrators may need to quickly disable thousands of users during a cyber security attack to prevent further damage. This is not possible using the Entra ID portal.

PowerShell is a task automation solution consisting of a scripting language and a command-

line shell (Microsoft, 2023d). Microsoft has provided PowerShell modules for administering Entra ID. The first module, MSONline, was published in the early 2010s, and the next one, AzureAD, was published later in the 2010s. Both were built to help administrators perform their duties, like managing Entra ID users and licenses. However, these modules are now set to be deprecated in March 2024 (Bash, 2023). The reason for this is that the underlying Application Programming Interfaces (APIs) the modules are using are deprecated at the same time. The only supported API after that is Microsoft Graph (MSGraph) API (Microsoft, 2023b).

Microsoft offers a replacement module for administrators called “Microsoft Graph PowerShell SDK” (Microsoft, 2023a). The module is an API wrapper for the MSGraph API, created automatically from the MSGraph API schema (Microsoft, 2023a). This is problematic for administrators as the module is not built to help them perform their tasks but allows them to call the MSGraph API directly from PowerShell. Even the name of the module contains the acronym SDK, which refers to Software

^a  <https://orcid.org/0009-0004-0045-865X>

^b  <https://orcid.org/0000-0002-6848-094X>

Development Kit. According to the global skills and competency framework for the digital world (SFIA), software development skills are not part of the system administrator role skill set (SFIA, 2023). Mastering the new module would require learning software development, especially MSGraph API.

1.2 Large Language Model

The large Language Model (LLM) is a special case of the Pre-trained Language Model (PLM). The challenge with PLMs is that they are unable to perform unseen tasks without task-specific training (Kalyan, 2024). Generative Pre-trained Transformers (GPTs) are using Natural Language Processing (NLP) for task-agnostic pre-training (Brown et al., 2020). Still, task-specific examples or datasets are required on top of general training to perform the desired task (Brown et al., 2020).

The public and scientific interest in LLMs has significantly increased after OpenAI published ChatGPT (OpenAI, 2024) for the general public in November 2022. For instance, ChatGPT articles published in arXiv have risen from zero in December 2022 to over 300 by May 2023 (Liu et al., 2023).

ChatGPT has been used successfully to generate software code (Liu et al., 2023) and to teach software development skills (Hellas et al., 2023). However, there have also been some challenges. For instance, the quality of the generated code cannot be guaranteed (Liu et al., 2023), and the judgements made may not be grounded in reality (Faggioli et al., 2023). This is often referred to as *hallucination* (Händler, 2023). Moreover, as an LLM, ChatGPT still lacks domain-specific knowledge (Faggioli et al., 2023).

1.3 Autonomous Agents

LLMs have been found to struggle with maintaining consistent logic across chains of reasoning (Händler, 2023). Autonomous LLM-powered agents can break user-prompted goals into smaller tasks (Händler, 2023). These tasks can be assigned to specialised agents utilising contextual resources such as tools (Händler, 2023). The balance between the autonomy of LLM-powered agents and the alignment of human users is illustrated in Figure 1. In other words, the agent should be self-organising if the user needs to perform tasks that demand active participation (L2).

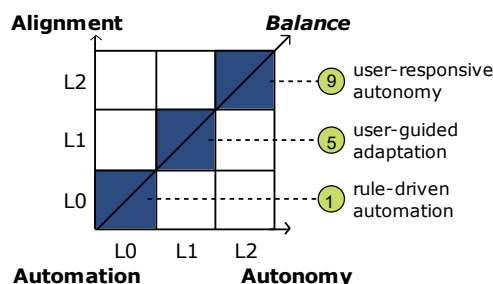


Figure 1. The interplay between agent autonomy and human users alignment (Händler, 2023, p. 90).

LLM-powered agents have demonstrated their performance in language understanding and interactive decision-making. However, usually, they have only been able to reason or generate an action plan based on human interaction. To overcome this, Yao *et al.* (2023) introduced a ReAct paradigm that combines reasoning and action within a closed-loop system that interacts with the external world. The autonomous LLM-powered agent and its collaboration with the user and outside tool is illustrated in Figure 2 using BPMN 2.0 notation. As the agent’s reasoning is based on access to external real-life entities, its problem-solving trajectory is “more grounded, fact-driven, and trustworthy” (Yao et al., 2023, p. 6). In other words, the agent is less prone to hallucinate.

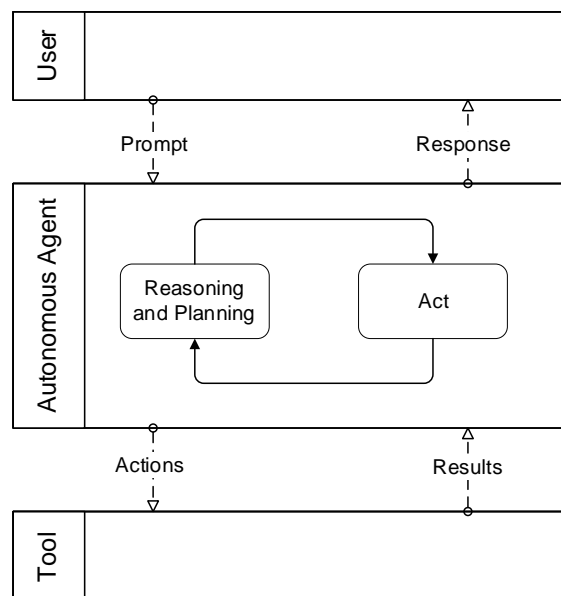


Figure 2. Autonomous LLM-powered Agent

1.4 Research Questions

Our research sought to answer the following research question:

How can we implement an autonomous LLM-powered agent to help administrators perform Entra ID administrative tasks?

1.5 Structure of the Paper

The rest of the paper is structured as follows. The building process of the *Microsoft Entra id Administration ageNt* (MEAN) is described in the next section. The last section, discussion, concludes the paper.

2 BUILDING THE MEAN

This paper reports a Design Science Research (DSR) study (see vom Brocke, Hevner, & Maedche, 2020), and we'll follow the DSR process guidelines by Peffers *et al.* (2007) in this section. Following Onggo (2012), the resulting agent is illustrated using BPMN 2.0 notation.

2.1 Motivation

The problem and motivation for the research were explained in the Introduction section. To sum up, Microsoft is deprecating existing Entra ID PowerShell modules, and the replacement module requires learning new skills that are not part of administrators' skillset.

2.2 Objectives

The solution's objective is to build an agent that helps administrators perform their tasks without needing to learn software development skills.

2.3 Design and Development – Round 1

Jupyter Notebooks (Jupyter, 2024) was selected as a development platform for the agent because it supports Python, giving easy access to ChatGPT-4 API. ChatGPT-4 supports *function calling*, which allows agents to call external APIs (OpenAI, 2023). The available functions and parameters can be sent to ChatGPT with the user prompt. The functions and parameters should be in the format that ChatGPT can interpret. The MSGraph API documentation (see Microsoft, 2024a) is not structured enough for ChatGPT. However, ChatGPT supports OpenAPI

specification (see OpenAPI, 2021). Microsoft publishes MSGraph API documentation in OpenAPI format (see Microsoft, 2024a), allowing it to be used with ChatGPT as-is.

The agent design is illustrated in Figure 3. The user sends the OpenAPI specification along with a prompt to the agent. Agent interprets (reasons) the input and plans the next steps based on the user prompt and the given OpenAPI specification. The agent starts to execute the plan by calling MSGraph API. The response from the API call is returned to the reasoning and planning process. When the plan is completed, the response is returned to the user.

The source code of the proof-of-concept (PoC) agent that implements the design is available on GitHub (see Rodriguez, 2023a). The PoC uses the MSGraph API *users* endpoint, which provides functionality for searching, creating, updating, and deleting users.

2.4 Demonstration – Round 1

The output of running the PoC agent is also available on GitHub (see Rodriguez, 2023a). The output shows the result of a simple query to find a username based on a given object ID.

After receiving the input from the user, the agent starts building a plan. First, it makes an *observation* of which API it could use to get the requested information. This leads to a *thought* that the agent is ready to execute the API call. Next, the agent starts acting on the plan and executes the *request_get* tool to make the actual API call. From the response, the agent makes an *observation* that the name of the user is "Example User". This leads to a *thought* that user information is successfully retrieved and the plan is successfully executed. The agent makes a final *observation* that the name of the user whose id was given is "Example User". This leads to a *thought* that the plan is finished and the information the user asked for is retrieved.

However, the agent was not able to perform more complex tasks. When asked to list all users whose password has been reset over six months ago, the agent answered *I'm sorry, but this API does not provide the functionality to filter users based on the time their password was reset*. The API returns only a subset of properties for users by default (see Table 1). All other attributes, such as *lastPasswordChangeDateTime*, would require using the *\$select* query parameter. To us, it seemed that the agent was only aware of the common properties, not all available properties.

Table 1. Common properties of users (Microsoft, 2024b)

Property	Description
id	The unique identifier for the user.
businessPhones	The user's phone numbers.
displayName	The name displayed in the address book for the user.
givenName	The first name of the user.
jobTitle	The user's job title.
mail	The user's email address.
mobilePhone	The user's cellphone number.
officeLocation	The user's physical office location.
preferredLanguage	The user's language of preference.
surname	The last name of the user.
userPrincipalName	The user's principal name.

Moreover, the agent seemed to be capable of showing only the 16 first users (sorted by *userPrincipalName* property). However, the API returns by default 100 users and allows returning up to 1000 users per request using *\$top* query parameter. Again, it seemed that the agent was not aware of this parameter.

2.5 Design and Development – Round 2

After the first design and development round, the agent seemed to work only partly as intended. It was able to perform simple tasks but didn't seem to understand the API correctly.

The original MS Graph API Open API specification YAML file had 28021 rows in total. Validating the file with the OpenAPI online validator (see APIDevTools, 2022) crashed the browser. The first lines of the file (see Figure 4) show that the file

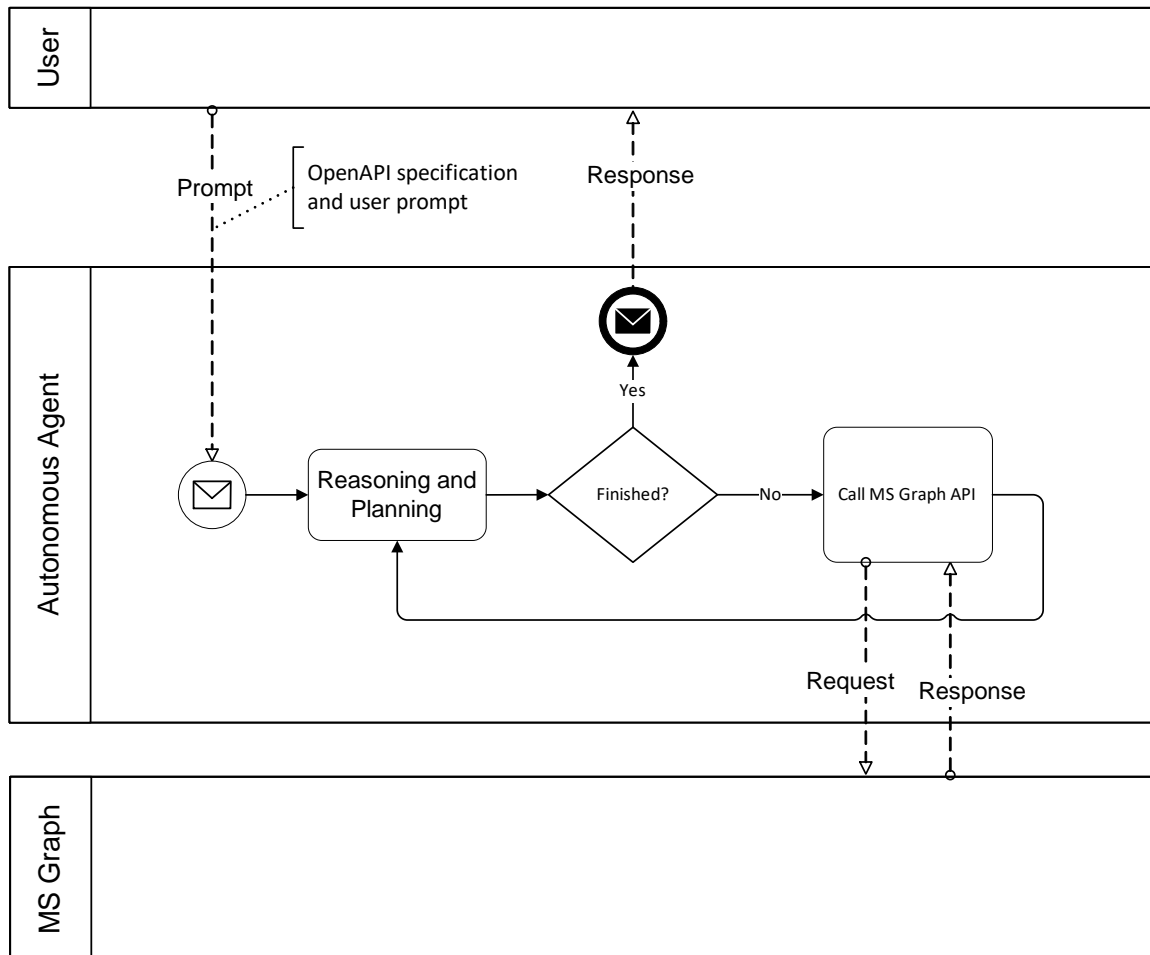


Figure 3. MEAN Design

is using a lot of internal *\$ref* clauses. Changing the setting of the online validator to skip circular references threw an error message “Circular \$ref pointer found”. For the same reason, in the first development round, we had to use *dereference=False* option when calling *reduce_openapi_spec* function (see Rodriguez, 2023b). That effectively did not follow any *\$ref* clauses and thus removed crucial information about the API. Moreover, *reduce_openapi_spec* ignored all query parameters that were not marked as required (see Langchain, 2023, step 3).

To fix the aforementioned issues, the MS Graph Open API specification was manually modified to remove unnecessary circular references and to change the type of all query parameters to required (see Syynimaa, 2024).

2.6 Demonstration – Round 2

The output of running the PoC agent to list *lastPasswordChangeDateTime* and *jobTitle* properties of users is available on GitHub (see Rodriguez & Syynimaa, 2024b). The output shows that the agent now understands the API better, i.e., how to use *\$select* query parameter. This time, the API threw an error indicating that *\$select* query parameter was specified more than once, which the

agent *observed* correctly. The resulting thought was that it should use just one *\$select* parameter and comma to separate the values. Next *observation* was the returned user information and the resulting *thought* that the data was successfully retrieved, and it could now be returned.

The process seems the same as you would expect from a natural person to execute a similar task. Instead of a natural person, the user could ask the autonomous LLM-powered agent to retrieve the needed information.

2.7 Evaluation

The conventional way to perform administrative tasks, such as finding a user with the given ID, would be to use an MSGraph PowerShell module. This would require the administrator using the module to know which API calls and corresponding PowerShell cmdlets they should use.

The designed agent allowed a user to complete a task that used MSGraph API without a need to learn software development skills. As such, the agent fulfils the set objectives. However, due to limitations of the current LangChain implementation, especially the *reduce_openapi_spec* function, do not allow the agent to release the full potential of ChatGPT. Even the source code of the function says “This is a quick

```

1  openapi: 3.0.1
2  info:
5  servers:
8  paths:
9  /users:
10  get:
11  tags:
12  - users.user
13  summary: List users
14  description: Retrieve a list of user objects.
15  externalDocs:
16  description: Find more info here
17  url: https://learn.microsoft.com/graph/api/user-list?view=graph-rest-1.0
18  operationId: user_ListUser
19  parameters:
20  - name: ConsistencyLevel
21  in: header
22  description: 'Indicates the requested consistency level. Documentation U
23  style: simple
24  schema:
25  type: string
26  examples:
27  example-1:
28  description: $search and $count queries require the client to set th
29  value: eventual
30  - $ref: '#/components/parameters/top'
31  - $ref: '#/components/parameters/search'
32  - $ref: '#/components/parameters/filter'
33  - $ref: '#/components/parameters/count'

```

Figure 4. Excerpt of OpenAPI-MSGraph-v1.0-Users.yml file

and dirty representation of OpenAPI specs” (Langchain, 2023).

2.8 Communication

The process and output of building the agent are reported in this research paper. The source code and Jupyter notebook are published on GitHub (see Rodriguez & Syynimaa, 2024a) and are freely available for anybody to download.

3 DISCUSSION

3.1 Implications to practice

We demonstrated that an LLM-powered autonomous agent can help administrators perform tasks that would otherwise require software development skills. This allows administrators to focus on daily activities instead of learning software development skills. Moreover, this greatly improves administrators’ efficiency in performing administrative tasks, especially in stressful situations like during cyber-attacks. However, due to limitations in the components used in the MEAN implementation, it is not mature enough to perform the day-to-day tasks expected from administrators.

3.2 Implications to science

In this paper, we designed and implemented a PoC of an autonomous agent capable of completing tasks by invoking MSGraph API based on the user’s natural language input. This proves that, despite the abovementioned challenges, LLM-powered autonomous agents can perform simple Entra ID administrative tasks. As such, the study supports the findings of previous studies (Nascimento, Alencar, & Cowan, 2023; Topsakal & Akinci, 2023).

3.3 Limitations

This study focused on performing tasks using a specific external tool, MSGraph API. Other APIs may yield different results.

3.4 Directions for Future Research

An interesting avenue for future research would be to study could agents be used to call PowerShell commands instead of back-end APIs. PowerShell module metadata, e.g., function descriptions and parameter details, can be extracted programmatically.

This could lead to a more generalised solution which is not limited to cloud services and could also allow using agents to perform tasks on local computers.

REFERENCES

- APIDevTools. (2022). Swagger/OpenAPI online validator. Retrieved from <https://apitools.dev/swagger-parser/online/>
- Bash, Kristopher. (2023). Important: Azure AD Graph Retirement and Powershell Module Deprecation. Retrieved from <https://techcommunity.microsoft.com/t5/microsoft-entra-blog/important-azure-ad-graph-retirement-and-powershell-module/ba-p/3848270>
- Brown, Tom B., Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, . . . Amodei, Dario. (2020). *Language models are few-shot learners*. Paper presented at the Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada.
- Faggioli, Guglielmo, Dietz, Laura, Clarke, Charles L. A., Demartini, Gianluca, Hagen, Matthias, Hauff, Claudia, . . . Wachsmuth, Henning. (2023). *Perspectives on Large Language Models for Relevance Judgment*. Paper presented at the Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval, Taipei, Taiwan.
- Hellas, Arto, Leinonen, Juho, Sarsa, Sami, Koutchme, Charles, Kujanpää, Lilja, & Sorva, Juha. (2023). *Exploring the Responses of Large Language Models to Beginner Programmers’ Help Requests*. Paper presented at the Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1, Chicago, IL, USA.
- Händler, Thorsten. (2023). *A Taxonomy for Autonomous LLM-Powered Multi-Agent Architectures*. Paper presented at the Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KMIS.
- Jupyter. (2024). Project Jupyter's origins and governance. Retrieved from <https://jupyter.org/about>
- Kalyan, Katikapalli Subramanyam. (2024). A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*, 6, 100048. doi:10.1016/j.nlp.2023.100048
- Langchain. (2023). Source code for langchain_community.agent_toolkits.openapi.spec. Retrieved from https://api.python.langchain.com/en/latest/modules/langchain_community/agent_toolkits/openapi/spec.html
- Liu, Yiheng, Han, Tianle, Ma, Siyuan, Zhang, Jiayue, Yang, Yuanyuan, Tian, Jiaming, . . . Ge, Bao. (2023). Summary of ChatGPT-Related research and perspective towards the future of large language models. *Meta-Radiology*, 1(2), 100017. doi:10.1016/j.metrad.2023.100017

- Microsoft. (2023a). Microsoft Graph PowerShell overview. Retrieved from <https://learn.microsoft.com/en-us/powershell/microsoftgraph/overview>
- Microsoft. (2023b). Use the Microsoft Graph API. Retrieved from <https://learn.microsoft.com/en-us/graph/use-the-api>
- Microsoft. (2023c). What is Microsoft Entra ID? Retrieved from <https://learn.microsoft.com/en-us/entra/fundamentals/whatis>
- Microsoft. (2023d). What is PowerShell? Retrieved from <https://learn.microsoft.com/en-us/powershell/scripting/overview>
- Microsoft. (2024a). Microsoft Graph REST API v1.0 endpoint reference. Retrieved from <https://learn.microsoft.com/en-us/graph/api/overview?view=graph-rest-1.0>
- Microsoft. (2024b). Working with users in Microsoft Graph. Retrieved from <https://learn.microsoft.com/en-us/graph/api/resources/users?view=graph-rest-1.0>
- Nascimento, N., Alencar, P., & Cowan, D. (2023, 25-29 Sept. 2023). *Self-Adaptive Large Language Model (LLM)-Based Multiagent Systems*. Paper presented at the 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C).
- Onggo, Bhakti S. S. (2012). *BPMN pattern for agent-based simulation model representation*. Paper presented at the Proceedings of the Winter Simulation Conference, Berlin, Germany.
- OpenAI. (2023). Function calling. Retrieved from <https://platform.openai.com/docs/guides/function-calling>
- OpenAI. (2024). Introduction to ChatGPT. Retrieved from <https://openai.com/blog/chatgpt>
- OpenAPI. (2021). OpenAPI Specification v3.1.0. Retrieved from <https://spec.openapis.org/oas/v3.1.0>
- Peffers, Ken, Tuunanen, Tuure, Rothenberger, Marcus A., & Chatterjee, Samir. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of management information systems*, 45-77.
- Rodriguez, Roberto. (2023a). A MS Graph API (Users) Agent. Retrieved from <https://github.com/OTRF/MEAN/blob/c9fb26610abf6be759ca3ce2abd82a4882349d9d/notebooks/OAI-MSGraph-Users-Agent.ipynb>
- Rodriguez, Roberto. (2023b). OpenAPI Spec - reduce_openapi_spec - maximum recursion depth exceeded. Retrieved from <https://github.com/langchain-ai/langchain/issues/12163>
- Rodriguez, Roberto, & Syynimaa, Nestori. (2024a). Github. MEAN: Microsoft Entra ID Administration LLM-based Autonomous Agent. Retrieved from <https://aka.ms/MEAN>
- Rodriguez, Roberto, & Syynimaa, Nestori. (2024b). Microsoft Entra ID Administration Agent (MEAN). Retrieved from <https://github.com/OTRF/MEAN/blob/ecf7c4c2f462d9c5f4dad974774c5499e5bf0c5d/notebooks/OAI-MSGraph-Users-Agent.ipynb>
- SFIA. (2023). SFIA 8 - illustrative skills profiles. Retrieved from <https://sfia-online.org/en/tools-and-resources/standard-industry-skills-profiles/sfia-8-skills-for-role-families-job-titles>
- Syynimaa, Nestori. (2022). *Exploring Azure Active Directory Attack Surface: Enumerating Authentication Methods with Open-Source Intelligence Tools*. Paper presented at the 24th International Conference on Enterprise Information Systems.
- Syynimaa, Nestori. (2024). OpenAPI-MSGraph-v1.0-Users_dereferenced.yml. Retrieved from https://github.com/OTRF/MEAN/blob/ecf7c4c2f462d9c5f4dad974774c5499e5bf0c5d/notebooks/openapi-specs/OpenAPI-MSGraph-v1.0-Users_dereferenced.yml
- Topsakal, Oguzhan, & Akinci, Tahir Cetin. (2023). *Creating large language model applications utilizing langchain: A primer on developing llm apps fast*. Paper presented at the Proceedings of the International Conference on Applied Engineering and Natural Sciences, Konya, Turkey.
- vom Brocke, Jan, Hevner, Alan, & Maedche, Alexander. (2020). Introduction to Design Science Research. In J. vom Brocke, A. Hevner, & A. Maedche (Eds.), *Design Science Research. Cases* (pp. 1-13). Cham: Springer International Publishing.
- Yao, Shunyu, Zhao, Jeffrey, Yu, Dian, Du, Nan, Shafran, Izhak, Narasimhan, Karthik, & Cao, Yuan. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models*. Paper presented at the The Eleventh International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda. https://openreview.net/pdf?id=WE_vluYUL-X