# Towards LLM-driven Automated Verification of Best Practices in Digital Public Infrastructures

Milan Markovic[1,*], Somayajulu Sripada[1], Sujit Kumar Chakrabarti[2] and Raghuram Bharadwaj Diddigi[3]

[1]*Department of Computing Science, University of Aberdeen, United Kingdom*
[2]*Department of Computer Science, International Institute of Information Technology - Bangalore, India*
[3]*Department of Data Science and Artificial Intelligence, International Institute of Information Technology - Bangalore, India*

### Abstract

In this position paper, we discuss three distinct approaches for assessing risks associated with Digital Public Infrastructures (DPI) and how Large Language Models could provide automated knowledge extraction to support such analyses at scale. We further outline future research directions inspired by the domain of collective intelligence and formal verification methods.

### Keywords

Digital Public Infrastructures, Infrastructure Analysis, Large Language Model, Formal Verification

## 1. Introduction

Digital public infrastructures (DPIs) are increasingly being deployed by governments, for example, for the delivery of welfare benefits to citizens and to accelerate progress towards achieving the SDGs set by the United Nations [1]. Past work indicates that the transition to digital modes of service delivery can complicate access to crucial government services and raises concerns of data security and privacy [2]. For DPIs to facilitate democratic and equitable access, they must adhere to certain best practices. However, investigating the presence of such best practices in DPIs can be time- and labour-intensive. At present, it is difficult to appraise DPIs for their conformance to regulatory requirements and other best practices. In this paper, we discuss how the current advancements in the Large Language Model (LLM) technology such as ChatGPT [3] could help to automate the analysis of DPIs.

## 2. Digital Public Infrastructure Analysis

DPIs are complex systems that may be a subject of analysis from various perspectives and organisations. We will discuss three different analyses of DPIs that could benefit from usage of LLMs. These analyses share some common requirements such as the need for consistent and reproducible results and additional domain information either providing the context (e.g. information about the target deployment region) or restricting the knowledge space for question answering (e.g., code, system documentation, specific reports, etc.).

### 2.1. High-level Socio-Technical Analysis

DPIs are often reviewed and monitored by external organisations without access to detailed inner workings of these systems. In this "black-box" approach the main focus is to analyse whether the system meets the socio-technical requirements (e.g., fairness of access, security, etc.). The analysis may

*SICSA REALLM Workshop 2024, October 17, 2024, Aberdeen, UK*

*Corresponding author.

milan.markovic@abdn.ac.uk (M. Markovic); yaji.sripada@abdn.ac.uk (S. Sripada); sujitkc@iiitb.ac.in (S. K. Chakrabarti); raghuram.bharadwaj@iiitb.ac.in (R. B. Diddigi)
0000-0002-5477-287X (M. Markovic); 0000-0002-5428-8383 (S. Sripada); 0000-0001-8422-2900 (S. K. Chakrabarti); 0000-0003-0233-0698 (R. B. Diddigi)

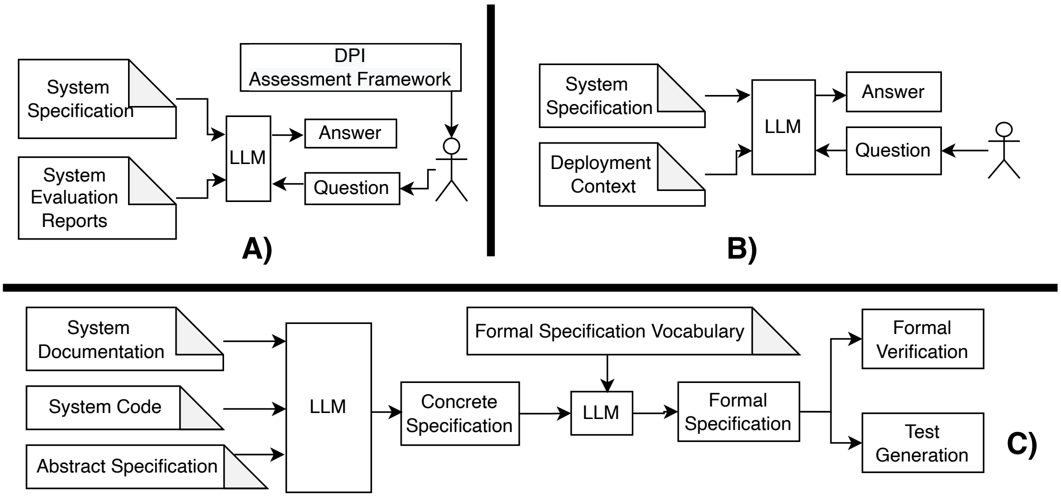


**Figure 1:** LLM-based pipelines for analysing DPI. A) A non-expert user interrogates external evidence base. B) A technical expert explores additional system requirements based on the deployment context. C) Formal verification process assesses a specific aspect of the system implementation.

be performed by non-technical experts who focus, for example, on social science aspects. We have worked with DVARA Research[1] who have developed an assessment framework consisting of a series of questions for which answers are sought through a comprehensive search of available evidence. This may include interviews with end users, external audit reports, documentation of the deployed system, news reports, etc. LLMs could help to speed up the review of the external documents against specific assessment frameworks (Fig. 1 A). Example questions inspired by the DVARA assessment framework include: *What financial details (e.g., bank details) does the system collect from citizens?*; *Where does the system obtain data about citizens?*; and *Is the system available in other languages?*. This type of analysis is especially challenging as the questions are posed by a non-technical expert who may lack the in-depth understanding of the generated answers. Quality and consistency of the results is therefore critical to elicit trust from the user.

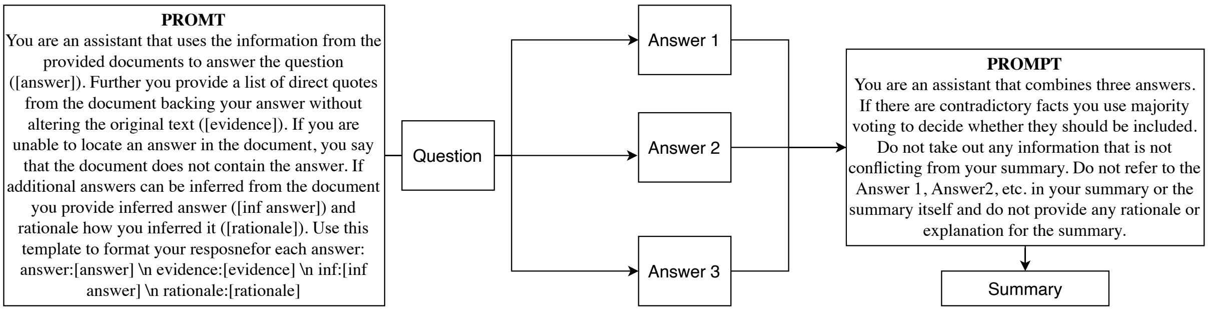
## 2.2. In-depth Technical Analysis & Testing

High-level socio-technical analysis can highlight areas that might require expanding the system design or verifying whether the system meets key criteria. These analyses demand a ”white box” approach and hence the LLM would need to deal with different forms of inputs such as code, diagrams, etc.

### 2.2.1. Expansion of System Requirements

System requirements may be formalised using variety of languages e.g. B-method [4] and notations such as SysML [5], UML [6] etc. These models may have missing details that do not take into account issues related to the environment where the system will be deployed. For example, a DPI that relies on live internet connections to validate identity of a person using large payloads may be unusable in areas with intermittent internet connection or insufficient bandwidth. Another example might involve people living in rural areas for whom DPI contact centres might be far away. If a DPI requires, for example, a multistage in-person registration process this might prove difficult, especially for the elderly and those who cannot afford to travel. An LLM module could be used to analyse the system design requirements within the context of the desired deployment scenario and suggest expansion or tweaks to system specification (Fig 1 B). However, to do so it is likely to require an up to date local information about the deployment area which is not present in the core model (e.g., local Internet speeds, planned locations of the contact centers, etc.).

[1] https://dvararesearch.com/



PROMT

You are an assistant that uses the information from the provided documents to answer the question ([answer]). Further you provide a list of direct quotes from the document backing your answer without altering the original text ([evidence]). If you are unable to locate an answer in the document, you say that the document does not contain the answer. If additional answers can be inferred from the document you provide inferred answer ([inf answer]) and rationale how you inferred it ([rationale]). Use this template to format your resposnefor each answer: answer:[answer] \n evidence:[evidence] \n inf:[inf answer] \n rationale:[rationale]

PROMPT

You are an assistant that combines three answers. If there are contradictory facts you use majority voting to decide whether they should be included. Do not take out any information that is not conflicting from your summary. Do not refer to the Answer 1, Answer2, etc. in your summary or the summary itself and do not provide any rationale or explanation for the summary.

**Figure 2:** An overview of the prompts used in our experiment where different GPT4 instances produced three answers to the same question and another GPT4 instance summarised these answers.

### 2.2.2. Formal Verification

DPIs pose a novel challenge to system designers due to their population scale deployment. Many non-functional requirements (NFRs) such as inclusivity, accessibility, availability, security, privacy and trustworthiness assume primary importance in case of DPIs. Such properties are hard to specify in simple natural language primarily due to their dependence on environmental factors, and their violation may be exceptionally hard to detect, as it may be caused due to the interaction of a multitude of components and agents. We believe that formal methods[2] may prove useful in ensuring that best practices are indeed followed in the design and implementation of DPIs. LLMs may play role in supporting the generation of formal specifications of best practices (Fig. 1 C) - i.e, in the form of properties or predicates using a mathematically rigorous notation with formal semantics describing an unambiguous interpretation of these best practices. This would then allow application of formal methods which have proved invaluable in domains of engineering where the cost of errors is high (e.g. safety critical and business critical software). We believe that formal verification techniques can be applied to automatically analyse design models and software implementations to discover bugs.

## 3. Discussion & Future Work

Below we will discuss two potential avenues of our future research focus stemming from a series of early experiments.

### 3.1. Application of Collective Intelligence Techniques to Control Quality of Results

The research community has previously spent a significant effort on exploring hybrid man-machine systems operating on the principles of collective intelligence - i.e. "*groups of individuals doing things collectively that seem intelligent*" [7]. Diverse concepts such as crowdsourcing [8], human computation [9], social machines [10], and social computation [11] have emerged and with them a range of system designs and techniques for managing the quality of results of such computations [12]. The issues related to the use of LLMs such as the influence of the task design (i.e., prompts), the characteristics of entities performing the task (i.e., different LLM models have different capabilities), and varying outputs generated by different workers are shared with the collective intelligence systems.

In our early experiments, we were inspired by the task decomposition and role separation [12] implemented by, for example, the *Create-Verify* pattern [13] which splits the workers' tasks into two groups. The first group of tasks is used to create some new data and these are then validated by the second group of worker tasks based on majority voting. We have reversed the task order by performing multiple create steps followed by a summarisation step where the LLM was asked to summarise the results of the create steps and use majority voting in case of conflicting facts (Fig 2). In our experiments, we asked the LLM to answer a question from the DVARA assessment framework (see Section 2.1) based

[2] By *formal methods*, we mean both *formal verification* and *automated software testing* under formal methods

on information from a pdf report on the Samagra system [14]. The question was answered in three separate GPT 4 conversations and the summarisation was also performed in a separate conversation.

For example, for a question "*Can citizens, government officials and last mile delivery agents login to the portal*?" the LLM produced two correct answers, however, one of the answers stated that the document did not contain the required information. The summarisation step resulted in a correct output listing the identified functionalities. Other examples include all three answers agreeing but also cases where the answers provided a wide spread information. This is similar to the parameters such as subjectivity and difficulty of a crowdsourcing task which may be correlated with the high variation in the produced answers [15]. We aim to investigate how other techniques from the collective intelligence domain may be applied or inspire quality assurance mechanisms in the LLM-based pipelines.

## 3.2. Using LLMs to Enable Formal Method Analysis at Scale

Specifications entailing non-functional requirements (NFRs) often appear as *abstract specifications*. For example, a security requiremenment might be expressed as "*Resources once deleted should not be available for viewing anymore*". In order to be applicable for specific application, abstract specifications must be translated into *concrete specifications*. For example, when a candidate revokes or cancels an application in an academic admission system, it should no longer be accessible for viewing by anyone.

Finally, such concrete, but informal, specifications can be translated into formal specifications. For example, consider a formal specification, written in a notation that borrows from Z [16] and design by contract [17] notations, of an API functionality of the *deleteApplication* and *viewApplication* methods:

**DeleteApplicationOK**

| | |
|---|---|
| Precondition | $(t \mapsto u) \in T, u \in Adm \vee a \in A[u]$ |
| API | `deleteApplication`$(t, a) \rightarrow HttpOK$ |
| Postcondition | $\nexists c \in C : a \in A[c]$ |

**ViewApplicationErr1**

| | |
|---|---|
| Precondition | $\nexists c \in C : a \in A[c]$ |
| API | `viewApplication`$(\_, a) \rightarrow HttpNotFound$ |
| Postcondition | _ |

The delete application API action is associated with precondition of presenting an active session token $t$ belonging to user $u$ with admin privileges ($u \in Adm$) and an application ID belonging to the same user ($a \in A[u]$). The postcondition of the delete action states that there should be no candidates in the database such that $a$ is among its application IDs ($\nexists c \in C : a \in A[c]$). The action of attempting to view previously deleted application is described with the precondition stating if ID $a$ does not belong to any current candidate $c$ ($\nexists c \in C : a \in A[c]$), the API call should return error *HttpNotFound*. Postcondition specifies there is no important change in the application state (shown as _). While such specifications are difficult to read by humans, they are ideal for formal verification and test generation. We are currently working on a platform that can generate automated tests from such specifications.

For example, consider a test to check whether a deleted application can be viewed:

```
r1 := deleteApplication('t1', 'a1')
assert(r1 = HttpOK)
r2 := viewApplication('t1', 'a1')
assert(r1 = HttpNotFound)
```

We argue that an LLM pipeline with access to contextual information about the application (e.g., source code, technical documentation) could help with creation of both concrete and formal specifications. This could be achieved through the utilization of Retrieval-Augmented Generation (RAG) mechanism [18] which allows LLMs to intelligently retrieve information from external databases to generate contextually relevant output. Furthermore, we also aim to investigate how the logical properties of the target specification language could be exploited to verify the validity of the generated LLM results.

## 4. Acknowledgments

This work was funded by the seed funding award made by the Royal Academy of Engineering (RaENG) as part of their Frontiers programme. We thank Aishwarya Narayan (Dvara Research) for her support.

## References

[1] United Nations Development Programme (UNDP), Accelerating the sdgs through digital public infrastructure, 2023. URL: https://www.undp.org/sites/g/files/zskgke326/files/2023-08/undp-g20-accelerating-the_sdgs-through-digital-public-infrastructure.pdf, accessed: 2024-08-28.

[2] A. Gupta, A. Narayan, B. Chugh, I. Ghosh, L. Narang, State of open digital ecosystems for social protection (sp-odes) in india, 2023. URL: https://dvararesearch.com/state-of-open-digital-ecosystems-for-social-protection-sp-odes-in-india/.

[3] L. Floridi, M. Chiriatti, Gpt-3: Its nature, scope, limits, and consequences, Minds and Machines 30 (2020) 681–694.

[4] K. Lano, The B Language and Method: A Guide to Practical Formal Development, 1st ed., Springer-Verlag, Berlin, Heidelberg, 1996.

[5] S. Friedenthal, A. Moore, R. Steiner, A Practical Guide to SysML, Third Edition: The Systems Modeling Language, 3rd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[6] J. Rumbaugh, I. Jacobson, G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition), Pearson Higher Education, 2004.

[7] T. W. Malone, R. Laubacher, C. N. Dellarocas, Harnessing crowds: Mapping the genome of collective intelligence, 2009. MIT Sloan Research Paper No. 4732-09. Available at SSRN: http://ssrn.com/abstract=1381502 or http://dx.doi.org/10.2139/ssrn.1381502.

[8] J. Howe, The rise of crowdsourcing, Wired Magazine 14 (2004).

[9] L. von Ahn, Human Computation, Ph.D. thesis, Carnegie Mellon University, 2005.

[10] T. Berners-Lee, M. Fischetti, Weaving the Web: The original design and ultimate destiny of the World Wide Web, Harper Collins, NY, 1999.

[11] D. Robertson, F. Giunchiglia, Programming the social computer, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371 (2013).

[12] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, M. Allahbakhsh, Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions, ACM Computing Surveys (CSUR) 51 (2018) 1–40.

[13] C. Callison-Burch, Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2009, pp. 286–295.

[14] A. Sharma, J. Copestake, M. James, The samagra anti-poverty programme in madhya pradesh: integrating household data, overcoming silo-problems and leaving nobody behind., Development Policy Review 39 (2021) 435–449. doi:10.1111/dpr.12502.

[15] Y. Jin, M. Carman, Y. Zhu, W. Buntine, Distinguishing question subjectivity from difficulty for improved crowdsourcing, in: J. Zhu, I. Takeuchi (Eds.), Proceedings of The 10th Asian Conference on Machine Learning, volume 95 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 192–207. URL: https://proceedings.mlr.press/v95/jin18a.html.

[16] J. M. Spivey, Understanding Z: a specification language and its formal semantics, Cambridge University Press, USA, 1988.

[17] R. Mitchell, J. McKim, B. Meyer, Design by contract, by example, Addison Wesley Longman Publishing Co., Inc., USA, 2001.

[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.