# Automating Data Flow Diagram Generation from User Stories Using Large Language Models

Guntur Budi Herwanto[1,2]

[1]Faculty of Computer Science, University of Vienna
[2]Department of Computer Science and Electronics, Universitas Gadjah Mada

**Abstract**

Visual modeling, particularly Data Flow Diagrams (DFDs), plays an essential role in modern software development, aiding in the design, understanding, and communication of system structures and potential security and privacy threats. Despite their importance, the manual creation of visual models is time-consuming highlighting the need for automation in the generation of DFDs from user requirements. Automating the generation of DFDs presents a significant challenge, especially in accurately interpreting user requirements and abstracting them into correct and complete diagram elements. The complexity of this task is compounded by the need for semantic accuracy and the ability to facilitate visual editing for human intervention. This study explores the use of Large Language Models (LLMs) to automate DFD generation, utilizing GPT-3.5, GPT-4, Llama2, and Mixtral models. This study emphasizes human oversight and employs an open-source diagramming tool to ensure that diagrams are accurate, complete, and editable. The findings reveal GPT-4's superior capability in generating complete DFDs, with significant progress from open-source models like Mixtral, indicating a viable path toward automated visual modeling. This approach advances scalable automation in creating visual software models, with broader implications for automating other diagram types.

**Keywords**

Visual Modeling, Data Flow Diagrams, Software Development, Large Language Models

## 1. Introduction

Visual modeling is an essential part of modern software development. It facilitates the design and understanding of systems while improving communication and documentation [1, 2]. One application of visual modeling is identifying potential security [3, 4] and privacy threats [5, 6] in software systems. One of the prominent diagrams for this case is the Data Flow Diagram (DFD). DFD illustrates data movement within processes, stakeholders, and the data store. DFDs also provide an understanding of system operations at multiple levels of granularity. They have proven effective in characterizing the system in privacy threat analysis [5]. In addition, the standard syntax of DFD has been extended [4] and adapted to explicitly model security [3] and privacy concerns [6].

*In: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider. Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2024. Winterthur, Switzerland, April 8, 2024.*

gunturbudi@ugm.ac.id (G. B. Herwanto)

However, manually creating visual models such as DFD is a time-consuming process[7], which can be a challenge when conducting threat modeling[7]. To address this issue, attempts have been made to automate creating DFDs from user requirements[8]. However, accurately interpreting and abstracting user requirements into DFD elements remains challenging using standard NLP approaches such as POS tagging and dependency tree [9]. The emergence of Large Language Models (LLMs) provides new opportunities to address the challenges of understanding semantic nuances.

Novel LLMs have garnered increased interest within the software engineering domain, particularly in the automation of model creation[10]. Although LLMs have demonstrated potential in converting Unified Modeling Language (UML) diagrams into code-based diagrams like PlantUML[10], concerns persist regarding the semantic accuracy of the generated models[10]. Furthermore, code-based diagrams present limitations regarding accessibility for visual editing. This feature is particularly important for individuals without coding experience or when communicating complex models to clients.

This study explores the potential of Large Language Models (LLMs) in assisting software development teams with diagram generation, focusing on the creation of Data Flow Diagrams (DFDs) by leveraging the capabilities of widely used LLMs, including GPT-3.5 and GPT-4, alongside two of the most proficient open-source models currently available, Mixtral-8x7B and Llama2. The objective is to produce diagrams that are not only accurate and complete but also editable using the open-source diagramming tool[1]. Recognizing the limitations of semantic validity in previous studies[10], the importance of human oversight in the process is emphasized. This research addresses several questions related to the effectiveness of LLMs in producing usable diagrams for software development teams. Through empirical investigation, the *completeness* and *correctness* of the DFDs generated by these models are assessed. Specifically, this study aims to answer the following questions:

RQ1 How do the Large Language Models (GPT-3.5, GPT-4, Mixtral-8x7B, and Llama 2) compare in terms of generating syntactically correct Data Flow Diagrams (DFDs)?

RQ2 How well do the DFDs generated by these Large Language Models represent the system's functionalities when compared to each other?

To address these research questions, the paper first outlines the syntactic rules specific to DFDs in Section 2. The proposed methodology is introduced in Section 3. The experiments conducted and the findings are described in Section 4. The threats to the validity of the approach are discussed in Section 5. Finally, the study concludes and summarizes the insights in Section 6.

## 2. Data Flow Diagram

Data Flow Diagrams (DFD) illustrate the movement of data between external entities, processes, and data stores within a system and serve as a tool to reveal relationships between system components and express functional requirements for complex systems [11]. The syntax and

[1]https://app.diagrams.net/

semantics rules that govern component connections and data transformations are fundamental to DFD correctness and ensure consistency across diagrams through specific guidelines [12]. These rules include:

1. External Entity: Each external entity must have at least one input or output data flow that facilitates interaction with the system.
2. Process: Each process within the DFD must have at least one input and/or output dataflow to ensure that processes are not isolated. Output flows typically have different names than input flows to distinguish the types of information being handled clearly.
3. Data flow direction: Data flows should move in one direction only, preventing cyclic or backward flows that might make the diagram difficult to interpret.
4. Connection to Process: Each data flow must connect to at least one process, providing a clear path for data movement within the system and ensuring that data does not exist in isolation.
5. Data Store Movement: Data cannot move directly from one data store to another; it must be processed by a process, emphasizing the role of processes in data transformation and movement.

There are also different levels of DFD; the higher the level, the more detail there is. In addition to syntactic rules, semantic rules ensure consistency between levels by requiring that the names of external entities and the data flow between processes and external entities remain the same across levels. [12]. In this study, the focus is solely on level one of the DFD.

## 3. Proposed Approach

This section presents the proposed approach of how to use large language models (LLMs) to transform a set of user stories into a DFD. Figure 1 summarizes the workflow of the proposed approach.

The process begins when a development team defines a user story set. Since the DFD is intended to represent the data flow clearly, overcrowding it with too many elements or connections can lead to confusion and reduce its readability. Therefore, an optional grouping of functionalities is recommended. The diagram shows an example from the ALFRED project, taken from the user story dataset [13]. It is a virtual assistant that helps elderly people to stay active. The stories can be grouped into several functional groups, such as health and safety features, communication and social interaction, etc. Each functional group, identified by its thematic or functional similarity among user stories, is represented by a distinct Data Flow Diagram (DFD).

These groups of user stories can then be used as input to a prompt as shown in Figure 2. The prompt is divided into four main parts: Task Description, Detailed Instruction, User Stories Input, and Few-Shot Prompt. This structure is designed to sequentially guide the LLM through the process of understanding the task, the method of execution, the input to be transformed, and the format in which the output should be structured.

The task description introduces the objective for the LLM, focusing on the correct representation of DFD elements without delving into the specific syntax rules of DFDs. Adding detailed

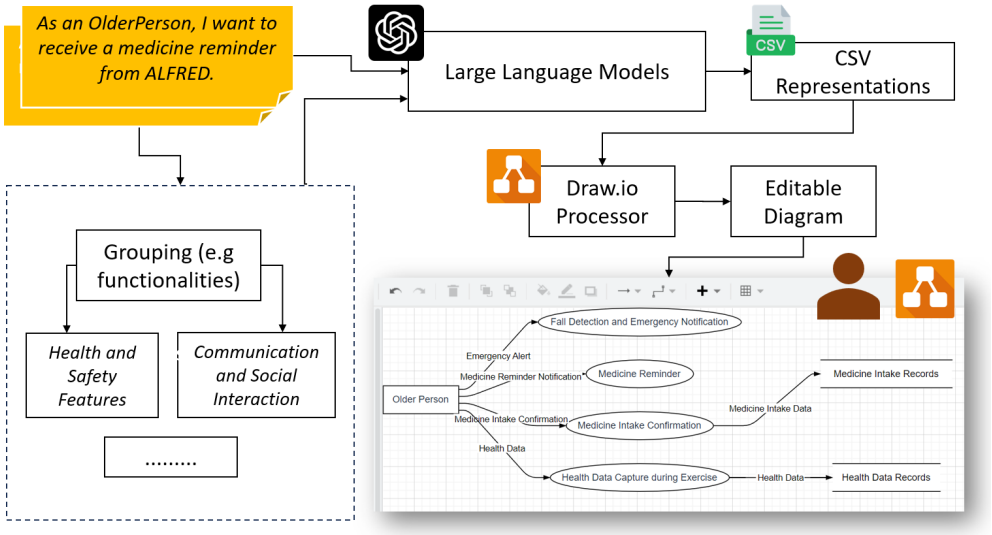



**Figure 1:** The method for creating a Data Flow Diagram (DFD) starts with user requirements. From there, it optionally groups functionality before using prompts in LLMs. The output of the prompts is in CSV format, which is then imported into draw.io. This process results in a generated, editable DFD.

instructions aims to enhance the LLM's ability to abstract concepts and ensure compliance with syntax requirements. Abstracting information effectively is vital to prevent the LLM from treating user stories as separate entities. The LLM must identify similarities among processes, actors, and data stores, capturing the essential flow of information accurately. The Few-Shot Prompt section introduces a CSV template that outlines the mandatory sequence the LLM must follow, aligning with the predefined syntax for use in draw.io. It is argued that generating the specific syntax without any initial template (zero-shot generation) is impractical in this context, considering the specific syntax required by the custom draw.io CSV import.

The output generated in response to this prompt includes the requested CSV format and additional textual information. Due to the format of this output, users are advised to interact through a chat interface under human oversight.

Finally, the Draw.io CSV import feature is utilized by first defining the style of the DFD elements and connections to create a standardized DFD representation. One advantage of using draw.io for diagrams, as opposed to code-based diagrams [10], is its editable nature. This allows for human visual input, which can be beneficial when communicating with customers.

## 4. Preliminary Evaluation

This preliminary evaluation aims to address the research question through empirical analysis. To ensure a thorough investigation, a variety of projects from an open dataset of user stories[13] were utilized. Following this, the LLMs and the evaluation metrics used to assess their output are detailed. Evaluators were then presented with DFDs generated by these models and asked to evaluate them. Lastly, the findings of the evaluation are outlined and discussed.

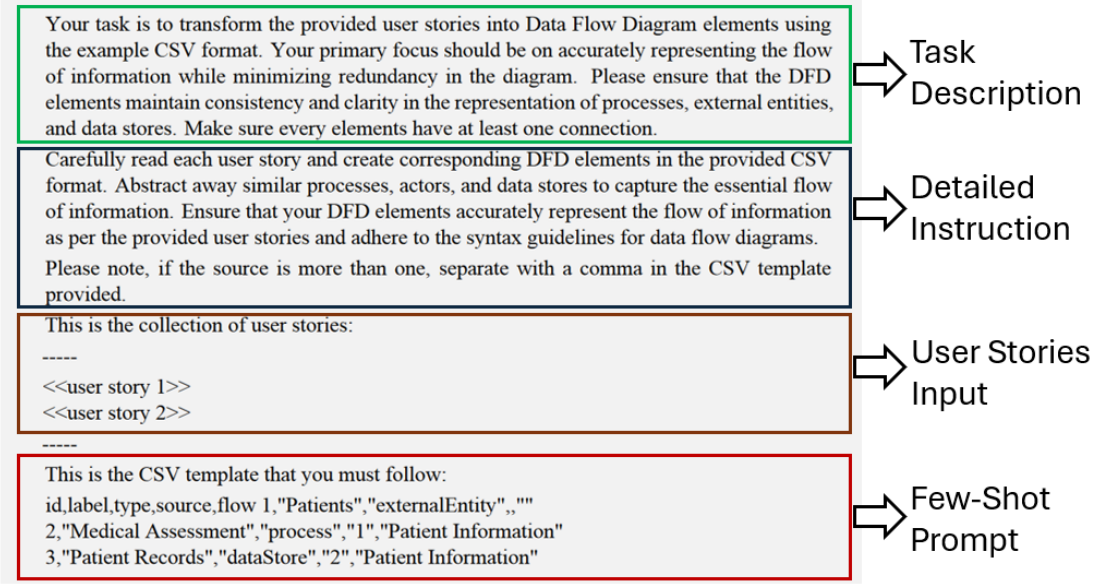



**Figure 2:** Prompt to Generate the CSV syntax of the Data Flow Diagram.

### 4.1. Experiment Setup

The study involved experimenting with four Large Language Models (LLMs). Among these, two were provided by OpenAI: GPT-3.5 Turbo and GPT-4 Turbo, accessible through the ChatGPT interface[2]. The other two open-source models, Llama2-70B[3] by Meta AI and Mixtral-8x7B[4] by Mistral AI, were accessed via the Together.AI[5] chat playground.

For the empirical analysis, three software engineering instructors with experience teaching DFDs participated. They evaluated the DFDs generated by each method based on completeness and correctness on a scale from 1 to 5, where 5 represented the highest score. The LLMs used are pseudonymized to eliminate any bias resulting from the instructor's familiarity with the LLM's capabilities. Prior to the evaluation, a meeting was held to standardize the scoring methodology among the three evaluators and the author.

The evaluation criteria are as follows:

- **Completeness**: This metric assessed whether the DFD included all essential elements (processes, data stores, external entities, and data flows) for a comprehensive system description. Each component must be connected to at least one other component to avoid isolated elements.
- **Correctness**: This involved verifying that the DFD accurately reflected the system's requirements, including logical data flow and consistent representation of external entities

[2]https://chat.openai.com
[3]https://huggingface.co/meta-llama/Llama-2-70b-chat-hf
[4]https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1
[5]https://api.together.xyz

and data stores following the system's external interactions. The goal was to ensure that the DFD faithfully represented the semantic content of the user stories.

From the open dataset[13], 17 projects were chosen, each featuring a variety of user stories. To keep the complexity of the DFDs at a manageable level for evaluation, five representative user stories from each project were selected. These stories were chosen to illustrate the system's primary functions and demonstrate the effectiveness of LLMs in delineating data flows among various stakeholders, with a specific focus on stories involving multiple stakeholders.

Five out of the 17 projects were assessed collectively by the three evaluators to maintain a manageable workload for the evaluators. The evaluators then individually assessed three projects, while the author evaluated the remaining three. The projects selected for collective evaluation included Alfred, CamperPlus, Recycling, NSF, and Datahub. To assess the completeness and correctness of the generated DFDs, a 1-to-5 Likert scale was used, enabling evaluators to quantify their judgments regarding the completeness and correctness of each DFD.

Generating DFDs began with manually entering the user stories into a chat interface, following a specific prompt described in Figure 2. The responses containing the desired CSV format were then integrated into predefined Draw.io CSV templates and uploaded to Draw.io for editing. The platform's auto-layout feature was used to organize the diagram elements, eliminating the need for manual adjustments. All materials produced, including CSV files, Draw.io configurations, and final DFD images, were documented and made available in a dedicated public repository[6].

## 4.2. Performance Results

The DFD example produced by this approach is presented in Appendix A. Table 1 displays the median completeness and correctness scores for each LLM method from the five collectively selected projects. Among these, GPT-4 is highlighted as the most accurate in generating DFDs, with the highest scores in completeness and correctness. This underscores GPT-4's superior capability in interpreting and converting user stories into DFDs. Mixtral, an open-source model, demonstrates commendable performance by producing useful DFDs with a respectable degree of accuracy and completeness. It outperforms closed-source models like GPT-3.5, which display moderate to lower effectiveness. Llama2 received the lowest scores, especially in correctness, suggesting that it faces challenges in understanding the semantics of the user stories. These results can also answer the two research questions:

> Answer to **RQ1**: GPT-4 outperformed other LLM for generating syntactically correct DFDs, scoring a median of 4.0 on a 1 to 5 Likert scale.

> Answer to **RQ2**: GPT-4 outperformed other LLM for accurately representing system functionalities in DFDs, scoring a median of 4.0 on a 1 to 5 Likert scale.

Note that there are no special syntax rules in the prompt. This demonstrates the LLM's internal knowledge of standard DFD conventions. Even if it is not a perfect score in terms of completeness and correctness, including human involvement in the further processing of the

[6] https://github.com/gunturbudi/drawio_llm

DFD could improve the applicability of the generated DFDs. In addition, the chat modality of these models offers the potential for iterative refinement of DFDs. Future studies could explore how iterative prompting could guide LLMs in adapting and improving DFDs. The threshold between what humans would accept the prompt and later edit in editable diagramming tools opens up an interesting empirical observation.

**Table 1**
The Median Scores of Completeness (Cm) and Correctness (Cr)

| Method | Cm (5) | Cr (5) | Cm (17) | Cr (17) |
|---|---|---|---|---|
| GPT-3.5 | 2.0 | 1.5 | 2.5 | 2.0 |
| **GPT-4** | **4.0** | **4.0** | **4.0** | **4.0** |
| Llama2 | 1.0 | 1.0 | 2.0 | 1.0 |
| Mixtral | 3.5 | 3.0 | 3.5 | 3.0 |

**Table 2**
Correlation Coefficients between Evaluator

| Evaluator Pair | Spearman | Kendall |
|---|---|---|
| 1 & 2 | .594 (<.001) | .532 (<.001) |
| 1 & 3 | .666 (<.001) | .549 (<.001) |
| 2 & 3 | .594 (<.001) | .505 (<.001) |
| Average | 0.618 | 0.529 |

Table 2 presents the correlation coefficients among pairs of evaluators, aiming to assess the level of concordance between them to ensure reliability in their assessments. To achieve uniform coding, scores are converted into rankings among LLMs, even if an evaluator assigns identical scores to multiple LLMs within a project. The ranking is determined based on the lowest rank within each group, using a dense ranking method that increments by one across different groups for consistency and clarity.

All pairs of evaluators demonstrate statistically significant positive correlations in their scoring, with Spearman's correlation coefficients ranging from moderate to strong (0.618 on average). While slightly lower, Kendall's tau values also indicate positive relationships and are significant (0.529 in average). These findings suggest a consistent agreement in the scoring among the evaluators, with higher scoring by one evaluator associated with higher scoring by another. The statistical significance of these correlations (p-values at <0.001) strongly supports the reliability of these observed associations, indicating that such agreement in scoring is not due to random chance but reflects a genuine pattern of concordance among the evaluators.

### 4.3. Discussion

Completeness scores are typically higher than correctness scores for all models, indicating that while LLMs can accurately identify necessary elements of DFDs with the correct syntax, they struggle to grasp the underlying semantics or logical connections between processes. This finding is consistent with previous research [10] and highlights the importance of a collaborative approach between humans and AI in the early stages of system design, particularly in diagram modeling. Nevertheless, the ability of LLMs to understand the syntax of DFD without explicitly mentioning the syntax and convention is a potential way to adapt to other open and widely used models, such as Business Process Model and Notation (BPMN) and UML.

The evaluator also found that data flow in the DFD tends to move exclusively to the left in horizontal and downward in vertical orientations, as shown in Appendix A. This observation suggests that data typically flows from external entities to processes and from processes to data stores, but rarely in the opposite direction. This limitation is likely due to the examples used

in the prompts, which demonstrated data flow in only one direction, highlighting the critical impact that the choice of examples in few-shot prompts can have on the model's performance.

This research employed a single prompt, which hinders the LLM potential of prompt engineering. For example, explicitly specifying the desired criteria for the DFD within the prompt might have been more advantageous than relying on the inherent knowledge of the LLM. Additionally, exploring the possibility of incorporating various aspects of DFD into a few-shot prompt represents another viable strategy. Despite LLM's understanding of the DFD syntax, it's critical to refine the few-shot examples to ensure they can accurately represent all potential directions of data flow.

Finally, it's important to recognize the inherent risks and limitations of relying solely on AI-generated models, such as the potential for confusing them with reality, losing information, or applying models that are inappropriate or incomplete [1].

## 5. Threats to Validity

This study faces threats to validity from two major fronts: the subjective nature of evaluating Data Flow Diagrams (DFDs) generated by Large Language Models (LLMs) and the selection process of user stories. Subjective evaluations may not accurately capture the constructs of completeness and correctness due to evaluators' variability, oversight of diagram details, and biases, challenging the construct validity of the findings. Concurrently, the potential bias introduced by selecting a limited, possibly simpler set of user stories poses a risk to the external validity of this study. This selection might skew the results towards more favorable outcomes by not fully representing the complexity and diversity necessary for a comprehensive assessment of LLM performance in generating DFDs.

To mitigate the first threat, a meeting is conducted between the author and three raters, and a clear expectation of what the Likert scale should mean for scoring is set. As for the second threat, the author set selection criteria for user stories and ensured that only these criteria were followed. However, there is still much scope to improve the validity of future research. Developing a structured assessment framework with clear, objective criteria for evaluating charts and automated comparison methods with predefined "gold standard" charts would be beneficial. These approaches aim to standardize the assessment process and reduce subjectivity to provide a more objective and reliable measure of the completeness and correctness of the diagrams. In addition, training evaluators and incorporating multiple perspectives into the evaluation process can further align judgments and capture a broader range of insights into the quality of the diagrams.

## 6. Conclusion and Future Work

Visual modeling has long been a key tool for uncovering the complexity of software requirements. Recent advances in large language models (LLMs) offer promising ways to augment human efforts in this domain. This study investigated the capacity of LLMs for generating Data Flow Diagrams (DFDs) within software development processes. The preliminary empirical evaluation focused on assessing the completeness and correctness of DFDs produced by GPT-3.5, GPT-4,

Llama2, and Mixtral-8x7B. The results show the leading performance of GPT-4 in achieving a particularly high score in the generation of syntactically correct DFD and the accurate representation of system functionalities in DFD. In addition, the efficacy of the Mixtral-8x7B model emphasizes the value of open-source models in broadening access to AI technologies in software engineering.

This research identified a common drawback of LLMs which is failing to accurately capture the semantics of the user requirements [10]. This highlights the imperative for human oversight in AI-assisted design processes. By synergizing the capabilities of LLMs with human expertise, there is potential to initiate diagrammatic representations of software systems effectively. Developing interactive tools that incorporate user input could significantly improve the fidelity of diagrams and promote a more collaborative design methodology. Consequently, a concentrated effort to refine LLMs' understanding of system semantics is essential. Such improvements would enable these models to more accurately encapsulate the technical nuances and domain-specific details that are essential for complete and accurate visual representations.

## Acknowledgments

The author thanks the anonymous reviewers for their valuable feedback on this paper. Appreciation is extended to Gerald Quirchmayr, Annisa Maulida Ningtyas, Diyah Utami Kusumaning Putri, and Dinar Nugroho Pratomo for their insightful evaluation and contributions to the paper. Furthermore, the author acknowledges the financial support received from the Indonesia Endowment Fund for Education (IEFE/LPDP), Ministry of Finance, Republic of Indonesia, and the assistance provided by the University of Vienna, Faculty of Computer Science.

## References

[1] J. Ludewig, Models in software engineering–an introduction, Software and Systems Modeling 2 (2003) 5–14.

[2] P. Caserta, O. Zendra, Visualization of the static aspects of software: A survey, IEEE Transactions on Visualization and Computer Graphics 17 (2010) 913–933.

[3] B. J. Berger, K. Sohr, R. Koschke, Automatically extracting threats from extended data flow diagrams, in: Engineering Secure Software and Systems: 8th International Symposium, ESSoS 2016, London, UK, April 6–8, 2016. Proceedings 8, Springer, 2016, pp. 56–71.

[4] L. Sion, K. Yskout, D. Van Landuyt, W. Joosen, Solution-aware data flow diagrams for security threat modeling, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, 2018, pp. 1425–1432.

[5] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, W. Joosen, A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements, Requirements Engineering 16 (2011) 3–32.

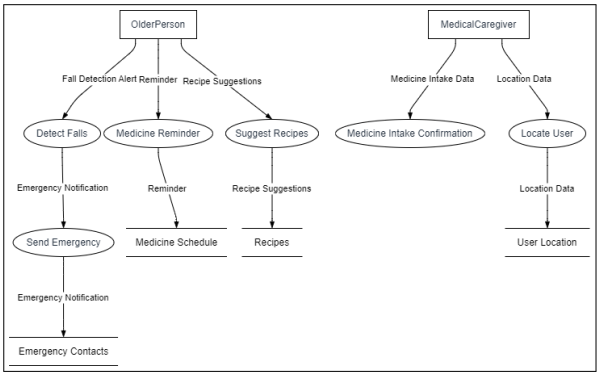
[6] H. Alshareef, S. Stucki, G. Schneider, Transforming data flow diagrams for privacy compliance., MODELSWARD 21 (2021) 207–215.

[7] K. Wuyts, L. Sion, W. Joosen, Linddun go: A lightweight approach to privacy threat

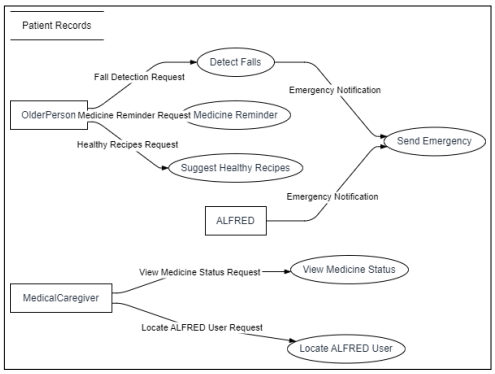
modeling, in: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, 2020, pp. 302–309.
[8] G. B. Herwanto, G. Quirchmayr, A. M. Tjoa, From user stories to data flow diagrams for privacy awareness: A research preview, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2022, pp. 148–155.
[9] G. B. Herwanto, G. Quirchmayr, A. M. Tjoa, Leveraging nlp techniques for privacy requirements engineering in user stories, IEEE Access 12 (2024) 22167–22189. doi:10.1109/ACCESS.2024.3364533.
[10] J. Cámara, J. Troya, L. Burgueño, A. Vallecillo, On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml, Software and Systems Modeling (2023) 1–13.
[11] E. Yourdon, L. L. Constantine, Structured design. fundamentals of a discipline of computer program and systems design, Englewood Cliffs: Yourdon Press (1979).
[12] R. Ibrahim, et al., Formalization of the data flow diagram rules for consistency check, arXiv preprint arXiv:1011.0278 (2010).
[13] F. Dalpiaz, Requirements data sets (user stories), Mendeley Data, V1 (2018).

## A. Sample DFD Output

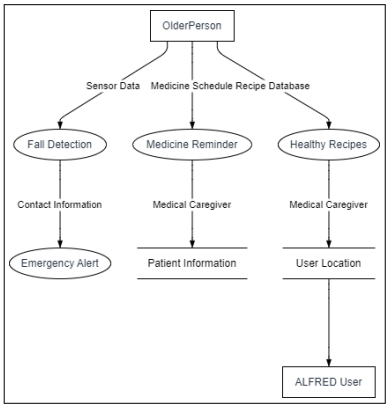
Figure 3 illustrates the data flow diagrams (DFDs) created by each of the LLMs methods for Alfred user stories.



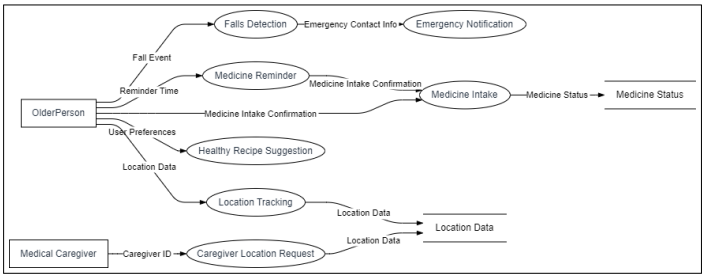

(a) DFD generated from GPT-4




(b) DFD generated from GPT-3.5




(c) DFD generated from Llama2




(d) DFD generated from Mixtral

**Figure 3:** DFDs Generated from Large Language Models