AI-assisted software modelling

Viktor Ovchinnikov

COMENIUS UNIVERSITY IN BRATISLAVA

May 14, 2025

Motivation

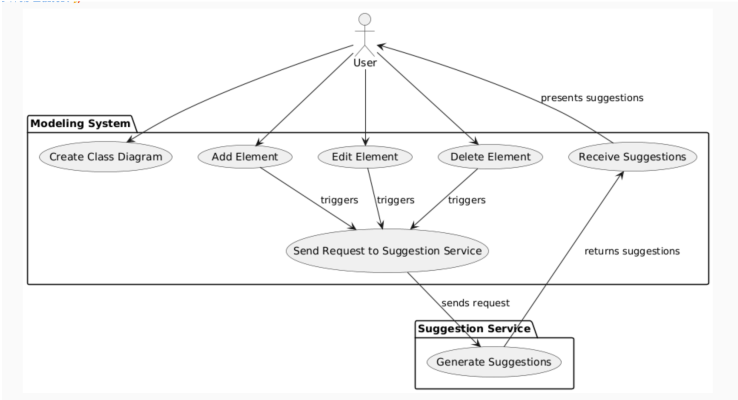


Figure 1: The following picture illustrates the use-case diagram of the AI-assisted component of AnimArch, a prototype software tool.

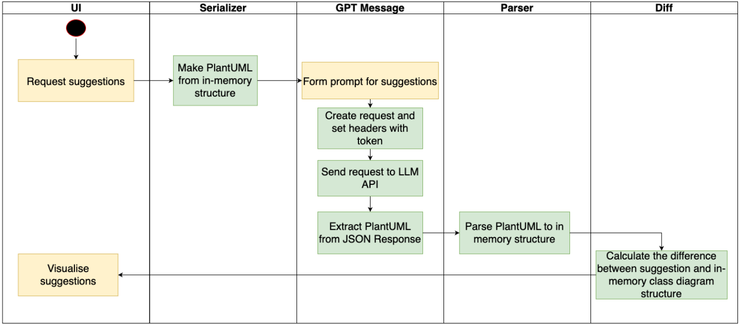
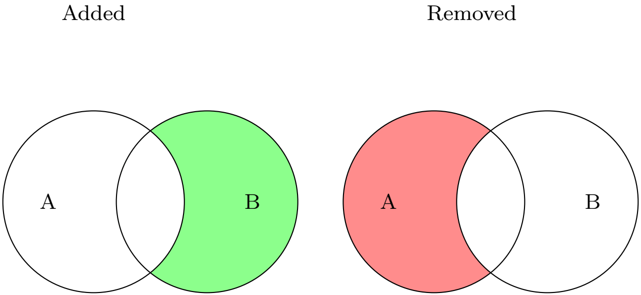

UI
Serializer
GPT Message
Parser
Diff
Request suggestions
Make PlantUML from in-memory structure
Form prompt for suggestions
Create request and set headers with token
Send request to LLM API
Extract PlantUML from JSON Response
Parse PlantUML to in memory structure
Calculate the difference between suggestion and in-memory class diagram structure
Visualise suggestions

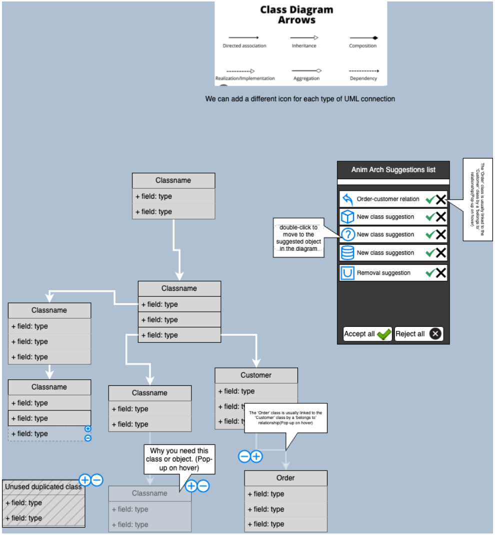
Figure 2: The following picture illustrates the activity diagram of the AI-assisted component of AnimArch, a prototype software tool.

Diff between A and B

- A in-memory structure
- B full UML diagram generated by LLM
- Added elements: B \ A
- Removed elements: A \ B



Where B is the complete UML diagram with suggestions.



We can add a different icon for each type of UML connection

The 'Order' class is usually linked to the 'Customer' class by a 'belongs to' relationship(Pop-up on hover)

double-click to move to the suggested object in the diagram

The 'Order' class is usually linked to the 'Customer' class by a 'belongs to' relationship(Pop-up on hover)

Why you need this class or object. (Pop-up on hover)

Figure 3: The provided image illustrates a conceptual prototype of the proposed user interface, accompanied by potential enhancements aimed at augmenting its functionality.

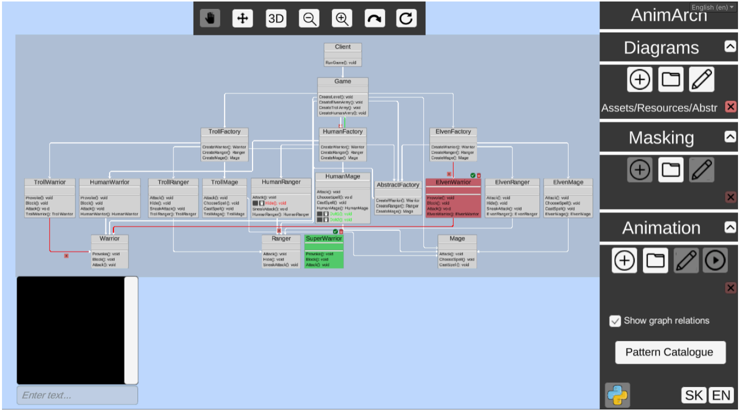


Figure 4: The image presented here illustrates the user interface of the system. Suggestions to add a new element—be it a class, method, attribute, or relation are indicated by a green. Suggestions to delete an element are indicated by a red.

Thank you for your attention!

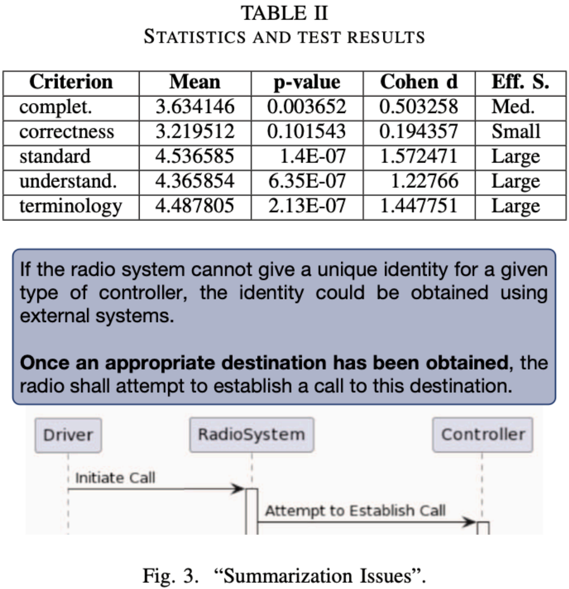


TABLE II
STATISTICS AND TEST RESULTS

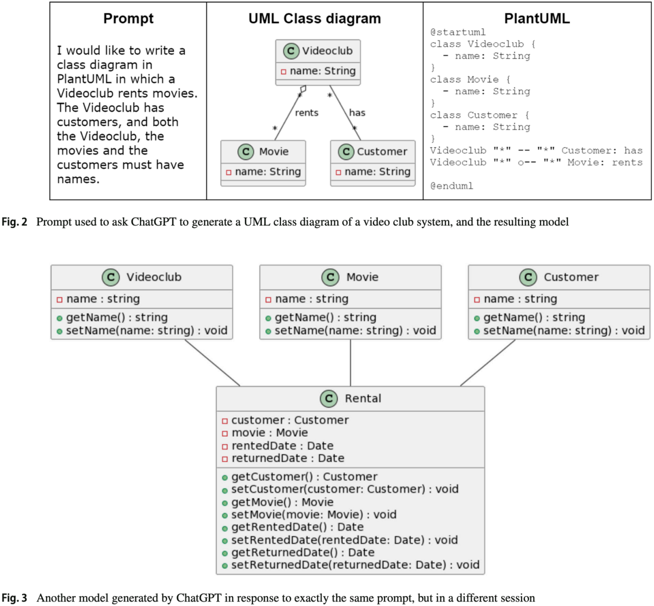
Criterion	Mean	p-value	Cohen d	Eff. S.
complet.	3.634146	0.003652	0.503258	Med.
correctness	3.219512	0.101543	0.194357	Small
standard	4.536585	1.4E-07	1.572471	Large
understand.	4.365854	6.35E-07	1.22766	Large
terminology	4.487805	2.13E-07	1.447751	Large

If the radio system cannot give a unique identity for a given type of controller, the identity could be obtained using external systems.

Once an appropriate destination has been obtained, the radio shall attempt to establish a call to this destination.

Fig. 3. “Summarization Issues”.

Figure 5: Model Generation with LLMs: From Requirements to UML Sequence Diagrams [2]



Prompt	UML Class diagram	PlantUML
I would like to write a class diagram in PlantUML in which a Videoclub rents movies. The Videoclub has customers, and both the Videoclub, the movies and the customers must have names.		

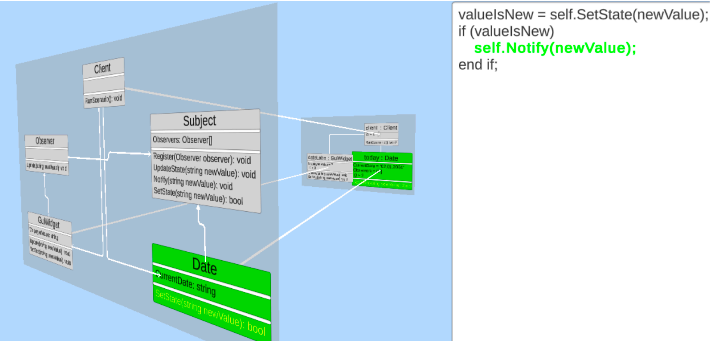
```
@startuml
class Videoclub {
  - name: String
}
class Movie {
  - name: String
}
class Customer {
  - name: String
}
Videoclub "*" -- "*" Customer: has
Videoclub "*" o-- "*" Movie: rents

@enduml
```

Fig. 2 Prompt used to ask ChatGPT to generate a UML class diagram of a video club system, and the resulting model

Fig. 3 Another model generated by ChatGPT in response to exactly the same prompt, but in a different session

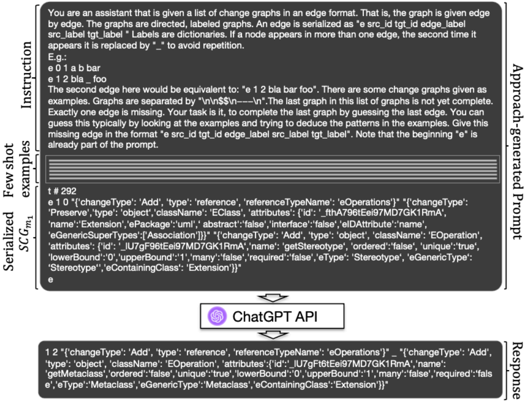
Figure 6: On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML [1]


Client
RunScenario(): void
Subject
Observers: Observer[]
Register(Observer observer): void
UpdateState(string newValue): void
Notify(string newValue): void
SetState(string newValue): bool
Observer
GuiWidget
Date
CurrentDate: string
SetState(string newValue): bool
client : Client
dateLabel : GuiWidget
today : Date

```
valueIsNew = self.SetState(newValue);
if (valueIsNew)
   self.Notify(newValue);
end if;
```

Figure 2: Overview of graphical user interface of our prototype.

Figure 7: Executable Multi-Layered Software Models [4]



```
You are an assistant that is given a list of change graphs in an edge format. That is, the graph is given edge
by edge. The graphs are directed, labeled graphs. An edge is serialized as "e src_id tgt_id edge_label
src_label tgt_label " Labels are dictionaries. If a node appears in more than one edge, the second time it
appears it is replaced by "_" to avoid repetition.
E.g.:
e 0 1 a b bar
e 1 2 bla _ foo
The second edge here would be equivalent to: "e 1 2 bla bar foo". There are some change graphs given as
examples. Graphs are separated by "\n\n$$\n–––\n".The last graph in this list of graphs is not yet complete.
Exactly one edge is missing. Your task is it, to complete the last graph by guessing the last edge. You can
guess this typically by looking at the examples and trying to deduce the patterns in the examples. Give this
missing edge in the format "e src_id tgt_id edge_label src_label tgt_label". Note that the beginning "e" is
already part of the prompt.
```

```
t # 292
e 1 0 "{'changeType': 'Add', 'type': 'reference', 'referenceTypeName': 'eOperations'}" "{'changeType':
'Preserve','type': 'object‘,'className': 'EClass', 'attributes': {'id': '_fthA796tEei97MD7GK1RmA‘,
'name':'Extension','ePackage':'uml',' abstract':'false','interface':'false‘,'eIDAttribute':'name',
'eGenericSuperTypes':['Association']}}" "{'changeType': 'Add', 'type': 'object', 'className': 'EOperation',
'attributes': {'id': '_lU7gF96tEei97MD7GK1RmA‘,'name‘: 'getStereotype‘, 'ordered':'false‘, 'unique':'true‘,
'lowerBound':'0','upperBound':'1','many':'false‘,'required':'false','eType‘: 'Stereotype‘, 'eGenericType‘:
‘Stereotype‘','eContainingClass‘: 'Extension'}}"
e
```

```
1 2 "{'changeType': 'Add', 'type': 'reference', 'referenceTypeName': 'eOperations'}" _ "{'changeType': 'Add',
'type': 'object', 'className': 'EOperation', 'attributes':{'id‘:'_lU7gFt6tEei97MD7GK1RmA‘,'name‘:
'getMetaclass‘,'ordered':'false‘,'unique‘:'true‘,'lowerBound':'0','upperBound':'1','many':'false','required':‘fals
e','eType':'Metaclass','eGenericType':'Metaclass','eContainingClass':'Extension'}}"
```

Figure 2: Detailed prompt and simple change graph serialization of the RAMC approach corresponding to the example given in Figure 1, exact few-shot examples are provided in the appendix.

Figure 8: Software Model Evolution with Large Language Models: Experiments on Simulated, Public, and Industrial Datasets. [6]

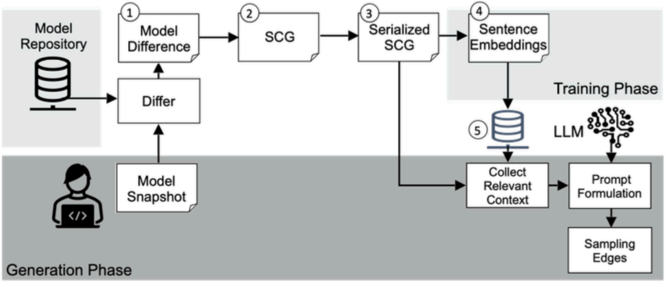


Figure 3: Overview of RaMc.

Figure 9: Software Model Evolution with Large Language Models: Experiments on Simulated, Public, and Industrial Datasets. [6]

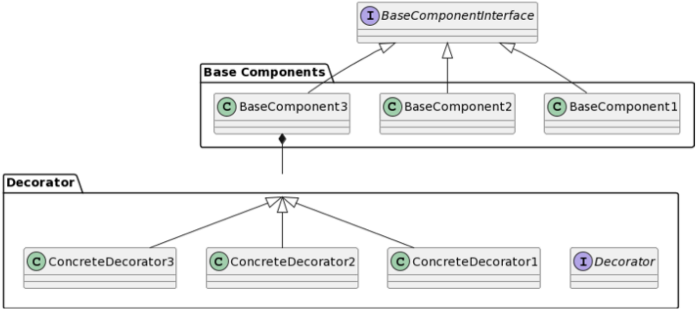


Fig. 1: Diagram Generated by ChatGPT

Figure 10: Creating UML Class Diagrams with General-Purpose LLMs [5]

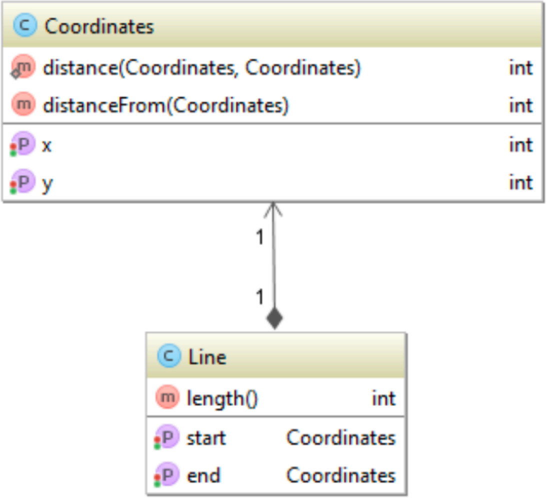


Figure 2. Class model with renamed class (Diagram 2).

Figure 11: An approach to compare UML class diagrams based on semantical features of their element. [3]

References I

[1] Javier Cámara et al. "On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML". In: Software and Systems Modeling 22.3 (2023), pp. 781–793.

[2] Alessio Ferrari, Sallam Abualhaijal, and Chetan Arora. "Model generation with LLMs: From requirements to UML sequence diagrams". In: 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW). IEEE. 2024, pp. 291–300.

[3] Oksana Nikiforova et al. "An approach to compare UML class diagrams based on semantical features of their elements". In: The tenth international conference on software engineering advances. 2015, pp. 147–152.

[4] Lukas Radosky and Ivan Polasek. "Executable Multi-Layered Software Models". In: Proceedings of the 1st International Workshop on Designing Software. Designing '24. Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 4651. ISBN: 9798400705632. DOI: 10.1145/3643660.3643938. URL: https://doi.org/10.1145/3643660.3643938.

[5] Mina Shehata et al. "Creating UML Class Diagrams with General-Purpose LLMs". In: 2024 IEEE Working Conference on Software Visualization (VISSOFT). 2024, pp. 157–158. DOI: 10.1109/VISSOFT64034.2024.00031.

[6] Christof Tinnes, Alisa Welter, and Sven Apel. "Software Model Evolution with Large Language Models: Experiments on Simulated, Public, and Industrial Datasets". In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE). IEEE Computer Society. 2025, pp. 649–649.