stepDP: A Step Towards Expressive and Pervasive Dialogue Platforms

Julian Wolter[0000−0003−3272−7290], Niko Kleer[0000−0003−4288−4724], and Michael Feld[0000−0001−6755−5287]

DFKI, Saarland Informatics Campus, Saarland University, 66123 Saarbrücken, Germany
{julian.wolter;niko.kleer;michael.feld}@dfki.de

Abstract. The advancement of human-machine interactions and the expectation to interact with devices in a natural way necessitates the development of multi-modal dialogue systems that can process and respond to various forms of input modalities. Despite effective recognition methods of the different modalities, their integration into reusable frameworks often falls short, impeding rapid development and the long-term maintenance of systems that incorporate multiple modalities. This paper introduces stepDP, a novel and open-source dialogue platform designed to overcome these challenges by facilitating the quick and efficient implementation of multi-modal dialogue systems following a model-driven design paradigm. Well-researched concepts and algorithms were integrated into a framework and fine-tuned to work together seamlessly. One core concept is that the dialogue logic is abstracted from actual input modalities, allowing for the generalisation of dialogue behavior and seamless integration across domains. Emphasizing modularity and flexibility, stepDP allows for the easy integration of new features, for example, the combination of traditional NLU techniques with innovative LLMs, without requiring extensive system modifications. Our platform not only accelerates the development process but also promotes the exploration of new concepts and techniques in human-machine interaction.

Keywords: Dialogue Systems · Multimodal Techniques and Modelling · Fusion · Blackboard.

1 Introduction

This paper introduces stepDP, an open-source and freely available[1] dialogue framework for multi-modal interactions. stepDP is a platform that is not tied to a specific domain or modality and can be used universally for the rapid development of a research prototype. A key objective is to make it easy to reuse already implemented components. In many cases, the modalities can be exchanged and the components still work. This makes it possible to offer a set of default components often required for dialogue systems, thereby enabling fast implementation

[1] https://stepdp.dfki.de/

of the prototype or application. By using established standards in software development such as JAVA, Maven, MkDocs, OpenAPI, and Spring, integration into existing and larger projects is straightforward.

stepDP is designed in such a way that it imposes as few rules as possible on the developer, allowing them to implement their interactions with maximum flexibility. Therefore, the use of most functions is optional. A blackboard is used as the central exchange point, which allows the various components to interact with each other. For this purpose, abstract tokens are exchanged, which are either written to the blackboard or read from there by the component. To know what the tokens look like, a knowledge base is built in which the types are represented inside a semantic tree. Additional knowledge can also be stored in the knowledge base as concrete instances of a specific type.

To carry out an interaction, rules that react to certain tokens on the blackboard can be defined. What the rules do is up to the user: creating new tokens and changing an entry in the knowledge base for triggering a hardware actuator, anything is possible. There is a set of default rules, e.g. for fusion, to quickly recreate standard dialogue behavior. A state chart is also included, which allows the status of the current dialogue to be represented and certain rules to be activated or deactivated accordingly, for example.

Furthermore, stepDP was designed with maximum flexibility in mind. All functionalities can be implemented via JAVA program code, but it is often also possible to model the behavior with JSON files only (or SCXML for the state chart). The knowledge base can either be initialized via JSON files, application code in JAVA, or external interfaces. Furthermore, custom components for stepDP written in JAVA can be integrated directly into the program code, included as components, or distributed as a separate Maven package.

This paper explains each component[2]. The main contribution compared to existing solutions is integrating all components into one framework:

- **Modality-independent**: Components can be implemented and reused independently of the modality.
- **Strict semantic**: Central and standardized semantics are used in all parts and every token has a type.
- **Multi-modality**: Through abstraction, stepDP is prepared at every level for multi-modality.
- **Fast prototyping**: The points above allow standard dialogue behavior to be provided to the developer.
- **Integration**: Easy integration with other components through the use of established software standards.
- **Flexibility**: Wide range of integrated dialogue concepts that can but do not have to be used.
- **Domain knowledge**: The integrated knowledge base with flexible data format and input allows domain-specific knowledge to be easily integrated.

[2] The website contains detailed and technical documentation of all components.

2 Related Work

The development of dialogue platforms is not a new discipline. Therefore, there are many existing concepts for certain application scenarios. However, these scientific platforms are usually aimed at a specific problem. Gropp et al. [6] have created a multilingual dialogue system, Chia-Yu Li et al. [14] presented ADVISER which also incorporated social factors into the dialogue, and Körber et al. [12] focused on cyber-physical environments.

There are not only different objectives for the dialogue systems but also different approaches. Lager et al. [13] showed how to use SCXML and Neßelrath et al. [17,18] presented a model-based approach for dialogue management. However, there is also a statistical approach by Jurčíček et al. [8] whereas Valenta et al. [21] presented a system based on a knowledge base and grammar. There is a large number of other systems, some of which have their niche [7,15,19], and there are some comprehensive surveys of these systems, for example by Kath et al. [9].

A central idea of stepDP is to distribute the dialogue management over several components and to use a central synchronization via a blackboard, as suggested by Kerminen et al. [10] and used by Fitrianie et al. [5] in their system for flexibility and expandability of our approach.

Rules can be used so that interactions take place from the blackboard and the current status can be represented via a state chart, which was partially used in the previously mentioned papers and is well-researched. We use SCXML for the state chart, which is very powerful in itself (e.g., Brusk et al. [2]).

We also looked at a variety of other systems to learn from their experiences and requirements when implementing a dialogue system with a specific goal so that our framework could cover these use cases [1,3,4,16,20].

2.1 Commercial Platforms

Several commercial products are available for building dialogues, each embedded within a major cloud service. Google Cloud Platform offers Dialogflow, a tool for creating conversational interfaces for websites and messaging platforms. Similarly, Amazon Web Services provides Amazon Lex, which powers the conversational interfaces using the same deep learning technologies as Alexa. Microsoft's Azure includes Language Understanding (LUIS), a service designed to apply custom machine-learning intelligence to a user's conversational, natural language text to predict overall meaning and extract relevant information. Other competitors include Cerence, which specializes in dialogue systems in the automotive sector. However, while these platforms are robust for commercial use, they present limitations for academic research which hampers the exploration of new dialogue system capabilities and the adaptation of these systems to novel research-focused applications.

3 stepDP

The dialogue platform stepDP presented here is aimed at anyone conducting research in the field of human-machine interaction. By using JAVA, our dialogue platform runs on all relevant systems and by relying on common libraries such as Spring Boot, an extension is easy to implement. All basic functionalities for dialogue interaction are provided so that a first simple dialogue is possible with just a few lines of code. Many important concepts for modeling an interaction, such as rules, knowledge base, and state charts, are already implemented and can be controlled directly via code or JSON files. Input and output are independent of the modality due to the concept of abstract tokens and the blackboard. Additional tools are available for processing the inputs and outputs, allowing a high degree of flexibility and customization so that almost any research setup can be implemented.

3.1 Conceptual Overview

The framework is based on four central concepts: the tokens, the blackboard with the rules, and the knowledge base (see Fig. 1). In principle, all inputs, outputs, and intermediate states are represented as tokens. These tokens are stored on a central blackboard and therefore visible to all components, both stepDP internal and external ones. The tokens on the backboard are processed by rules that can trigger an interaction. The framework and the technical documentation are available on the Internet under the MIT license. stepDP is aimed at researchers and is not designed with commercial use in mind.

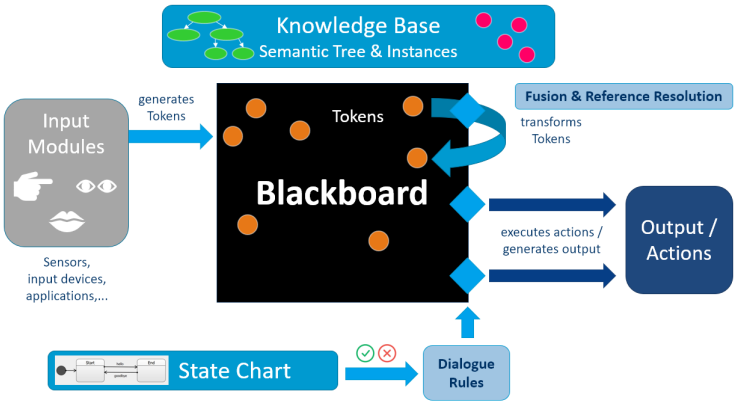

Fig. 1. Schematic representation of the core components of stepDP . All components communicate via the blackboard with abstract tokens whose semantics are stored in the knowledge base.

3.2 Abstraction

The main goal of stepDP is to be as flexible as possible to map all research scenarios and still support researchers in implementing their scenarios. To achieve this balancing act, a key element is a high level of abstraction. In addition, there are no fixed paths that have to be followed except for the blackboard, as is the case with other systems. Although there are a few recommendations and helper classes that favor a certain processing path, there is no obligation to use these.

To integrate a large number of different components, whether hardware or software, all of these communicate via tokens in stepDP that are placed on the blackboard. Tokens need to have a type that is specified in the knowledge base with the according properties, but additional fields with data can be added. Therefore, users have full flexibility as these tokens can contain almost anything and can still be processed by the functions already provided in stepDP .

3.3 Tokens and Blackboard

The important feature of stepDP is the central blackboard on which all tokens are stored. Tokens on the blackboard are immutable once they have been placed on it. A token is initially just a piece of information assigned to a type. This type can define further property fields of the token if necessary. A token is normally represented in JSON, e.g., whenever added to or read from the blackboard via an external API endpoint. An example token is shown in Fig. 2. Any fields with information can be added so that all relevant information can be stored in a token. A property can also contain a reference to an entry in the knowledge base. Tokens are not only used for input and output or interaction with external components but may contain intermediate steps or serve as a memory for the dialogue. The knowledge base is also represented via tokens so that knowledge from it can be utilized easily to interact with the blackboard.

```
{ type : "PointingEvent",
  objectPointed : "BottleC",
  certainty : 0.97,
  source : "DeviceA",
  customPayload : "abc" }
```

Fig. 2. JSON representation of a token of type "PointingEvent".

The Blackboard represents a core concept within the architecture of stepDP , serving as the initial repository for all tokens. Once placed on the Blackboard, they are either processed further or act as catalysts for subsequent actions. In particular, a specific token may initiate a dialogue interaction or multiple tokens may be required to form a new token.

There are two categories of tokens: active tokens and archived tokens. Active tokens are in the pipeline to be actively processed, whereas archived tokens, despite being inactive in triggering mechanisms, remain retrievable. By standard practice, tokens transition to the archived status following a duration of five minutes, though this parameter is adjustable both globally and on an individual token basis. For instance, a token associated with a pointing device may prefer a shorter active period, given the diminishing relevance of its processing post a brief interval. After their archival, tokens are irrevocably removed from the blackboard 60 minutes later, rendering them inaccessible.

3.4 Rules & Conditions

Within the Blackboard system, tokens are subject to evaluation and manipulation based on a set of rules and conditions. Conditions are tasked with identifying the circumstances under which tokens are to be processed, whereas rules define what kind of processing is happening with one or multiple tokens. For a rule to be invoked, it must be associated with a condition that specifies the tokens that should be processed by the rule (see Fig. 3). These conditions can be arbitrarily complex, ranging from simple checks on the type to firmly defined properties and time intervals to each other if several tokens are required. Active tokens are continuously streamed to the relevant condition, which in turn filters out those tokens that fulfill its criteria. These filtered, or matched, tokens are subsequently directed to the associated rule for processing, potentially triggering specific actions defined by the token's properties.

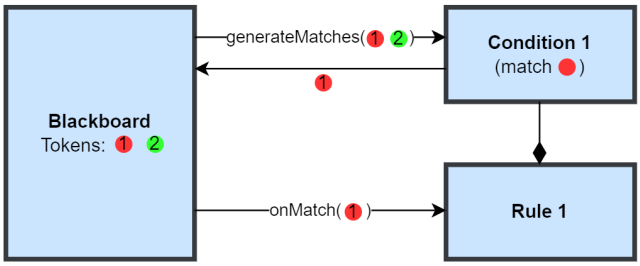

Fig. 3. Blackboard with two tokens of type red and green. The rule has a condition that only accepts tokens of type red. This condition filters the red token from the blackboard and forwards it to the rule for processing.

The execution order of rules is governed by a system of prioritization, with priority levels numerically assigned. The most urgent priority is denoted by a zero. In instances where multiple rules share identical priority levels, precedence is given to the rule that was first integrated into the Blackboard system. There is also a state chart system that can set active and inactive rules.

One type of condition is already integrated into stepDP , known as a pattern condition, which is designed to identify tokens that conform to a predefined structure or pattern. This pattern may specify various aspects of a token, including its type, the characteristics of its properties, and the presence (or absence) of certain properties, ensuring they are not null.

Despite the prioritization system, the potential for conflicts among rules and conditions remains. This scenario arises when multiple rules, each defined by distinct patterns aimed at matching varying configurations of a token, exhibit overlapping criteria. Consequently, a single token might satisfy the conditions for more than one rule. To mitigate such overlaps and ensure that tokens are exclusively processed by the most pertinent rule, the system incorporates the use of exclusion tags. These tags facilitate the selective engagement of tokens with the most specific rules, effectively sidelining less relevant ones. However, the application of exclusion tags is discretionary, allowing for their omission should there be scientific or experimental reasons to explore the interactions between concurrently applicable rules.

stepDP comes with a set of predefined rules that can be used as a template. One of the more powerful predefined rules is the fusion rule (see Fig. 4). This allows several tokens to be merged into a new token. The behavior of the rule, or more precisely the condition, can be described semantically. This makes it very easy to add another modality later or to quickly adapt the behavior of the rule without major code changes. It is further possible to reuse fusion rules from a previous project for a new project.

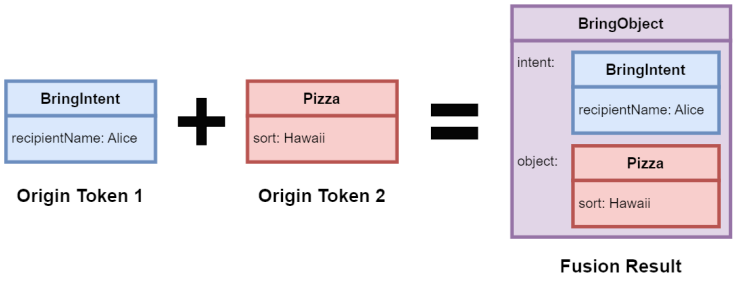

Fig. 4. Working principle of a fusion rule: In this example, the fusion rule takes two tokens of the type "BringIntent" and "Pizza" and merges them into a new token of the type "BringObject".

3.5 Statechart

stepDP extends its versatility by including the option to model dialogue flows using state charts built upon its rule system. This feature is particularly helpful for managing larger dialogue systems, offering an additional layer of organization

and structure. In this framework, each state within the state chart is linked to a specific collection of active rules. Consequently, changing the current state of the state chart results in the activation and deactivation of rules in alignment with the configuration. The mechanism for transitioning between states is designed to be responsive to triggers from either tokens or the rules themselves.

3.6 Knowledge Base

In the initial phase of developing your dialogue application, you should think about the semantic tree and implement it accordingly. This tree serves as the foundation for defining all necessary types within the dialogue system. Initially, stepDP provides a generic type, Object, from which all other types are derived. Once the semantic tree has been established, encompassing all requisite types, these types can then be instantiated into concrete instances (see Fig. 5). These instances are persistently stored within the knowledge base, contrasting with the transient nature of tokens on the blackboard, which are archived after a predetermined duration and subsequently deleted.

For a wide array of applications, it proves beneficial to integrate instances from the knowledge base into the blackboard. This integration allows for the dynamic processing of these instances through the established rules and conditions of the system. It can also be used to create a long-term memory that does not forget things as quickly as if you just use tokens. Such integration can be accomplished by importing these instances from JSON files or by directly utilizing JAVA code within your dialogue application. This approach not only enriches the dialogue system with a robust knowledge base but also leverages the flexibility of stepDP in processing complex information and using this information for interaction with the user.

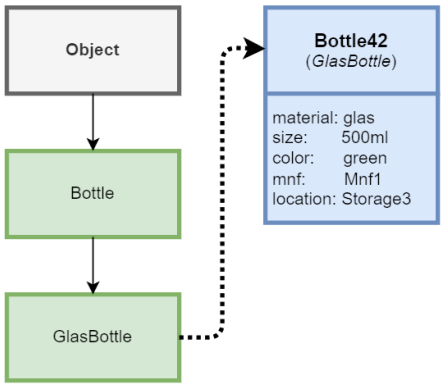

Fig. 5. On the left is a section of the semantic tree (properties are not visualized) and on the right is a concrete instance of an object of type "GlasBottle".

4 Implementation

The dialogue framework is implemented in JAVA as a maven package, which enables simple resolution of dependencies and easy integration into a new project. stepDP itself uses the Spring Boot Framework as a basis. Extensions can easily be added by including the Maven-specific *pom.xml* file. Some extensions are provided directly by us through our Maven repository or Github page, but third parties may provide extensions that can be added with just a few lines to the *pom.xml* file. The use of Spring Boot makes it easy to extend the API and create your API endpoints. In addition, stepDP runs on all platforms where the JVM (Java Virtual Machine) is available.

To implement a dialogue based on stepDP , one has to create a Maven project and add stepDP as a dependency. All other dependencies are then resolved automatically. Additional external components can be added directly to the project configuration. The "Hello World" dialogue can be created in just a few lines using templates provided by stepDP . Since only abstract tokens are processed, additional components are normally required to act as an interface between the tokens and the input and output modalities.

4.1 External Components

As stepDP has already been used for many projects, there is a large pool of external components available.

Since stepDP itself only works with abstract tokens, the processing of language (be it written or spoken language) into these tokens is particularly important. For this reason, there is the "stepAM (Audio Manager)"[3], which manages audio streams. This tool can be used, for example, to stream the microphone of a Hololens to stepDP . Voice activity recognition can then be performed on the audio stream and the sections with speech can be sent to a service that generates the required tokens from the speech. There are currently two extensions that connect to speech-to-text services and an intent recognizer: an extension for Cerence Mix and another one for Microsoft Luis. Both extensions accept written text. An integration with RASA is currently under development. Furthermore, there are external components for connecting the knowledge base to an MQTT broker for exchanging information or to connection to the Asset Administration Shell. Various robots have been connected to stepDP as external components.

4.2 Debugging

Especially more complex dialogues require suitable debugging tools to find and correct bugs efficiently. For this purpose, stepDP was equipped with a web interface in which the current status can be tracked and manipulated. A browser can be used to investigate the blackboard to see which tokens are currently active. It is further possible to view the properties of these tokens. In addition, all rules are

[3] https://stepdp.dfki.de/tools/audiomanager/

listed and their activation status is displayed. A state chart where the current status is highlighted can also be selected. If required, there is an input mask for writing tokens directly to the blackboard or changing the knowledge base.

The interface includes a primitive speech output for an initial simple dialogue by using the text-to-speech module built into the browser to generate an output. Finally, there is a text field that can be used for debugging purposes. It supports several concurrent sessions and may function as an extra modality for your dialogue system.

5 Applications

In principle, stepDP is suitable for any use case in which a dialogue platform or a platform for managing interactions is required. Internally, we have already used stepDP for many different scenarios in various domains.

For example, stepDP has already been used as a dialogue platform to control an underwater robot as part of the Mare-IT project[4]. The user controlled the robot by wearing a Hololens and using their voice and gestures. The visualisation in the Hololens was triggered via stepDP and inputs in the Hololens were sent back to stepDP. The robot was also connected to stepDP. Kleer et al. [11] used stepDP to teach a robot a task via multi-modal teach-in. Apart from robotic scenarios, stepDP was used to test and evaluate algorithms for the reference resolution of speech input. Furthermore, stepDP is utilized in the "Software Campus"[5] research project AquaChat[6], which aims to design an adaptive chatbot, requiring the integration of an LLM.

The documentation on the website contains many other examples of how stepDP can be used for a wide range of scenarios, from simple examples with just a few lines of code or JSON files to complex dialogues with complex behaviour.

Achieving identical objectives with alternative systems could pose numerous challenges or demand considerably more effort. For instance, using RASA for the robot teach-in project by Kleer et al. would greatly complicate the task of collecting and confirming parameters through multimodality, due to the absence of built-in mechanisms for multimodal fusion. The same limitations apply to DialogFlow from Google. While it offers advanced functionalities for productive operation, these features are not easily customizable or extendable, as required for research projects such as AquaChat. A more comprehensive comparison is accessible on the website[7].

6 Conclusions

We have described stepDP, a new multi-modal dialogue platform. By focusing on modality independence, strong semantics, rapid development, and a high degree

[4] https://robotik.dfki-bremen.de/de/forschung/projekte/mare-it
[5] https://softwarecampus.de/certificates/18560/
[6] https://stepdp.dfki.de/research/aquachat/
[7] https://stepdp.dfki.de/stepdp/comparison/

of reusability, the platform has proven itself very useful for us several times. The high degree of integration of various dialogue management models ensures that the required methods are available for almost every application. The central interface of the blackboard with the abstract tokens that are strictly typed allows previously implemented components to be reused, which has resulted in a large repertoire of useful components over time. Thanks to the use of JAVA, Maven, and Spring Boot, countless libraries are available to easily integrate additional software (e.g., audio libraries, interface libraries for MQTT, databases, or file formats). By using it for several internal projects, we were able to demonstrate that stepDP can be used successfully and quickly as a dialogue platform and meets expectations regarding sustainability.

Acknowledgements. This work is supported by the German Federal Ministry of Education and Research in the projects MADMACS (grant no. 01IW14003), TRACTAT (grant no. 01IW17004), CAMELOT (grant no. 01IW20008), and FedWell (grant no. 01IW23004).

References

1. Ábalos, N., Espejo, G., López-Cózar, R., Callejas, Z., Griol, D.: A multimodal dialogue system for an ambient intelligent application in home environments. In: Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic, September 6-10, 2010. Proceedings 13. pp. 491–498. Springer (2010)
2. Brusk, J., Lager, T., Hjalmarsson, A., Wik, P.: Deal: dialogue management in scxml for believable game characters. In: Proceedings of the 2007 conference on Future Play. pp. 137–144 (2007)
3. Bui, T.H., Rajman, M., Melichar, M.: Rapid dialogue prototyping methodology. In: Sojka, P., Kopeček, I., Pala, K. (eds.) Text, Speech and Dialogue. pp. 579–586. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
4. Cenek, P.: A flexible framework for evaluation of new algorithms for dialogue systems. In: Sojka, P., Kopeček, I., Pala, K. (eds.) Text, Speech and Dialogue. pp. 437–440. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
5. Fitrianie, S., Wiggers, P., Rothkrantz, L.J.: A multi-modal eliza using natural language processing and emotion recognition. In: Text, Speech and Dialogue: 6th International Conference, TSD 2003, České Budéjovice, Czech Republic, September 8-12, 2003. Proceedings 6. pp. 394–399. Springer (2003)
6. Gropp, M., Schmidt, A., Kleinbauer, T., Klakow, D.: Platon: Dialog management and rapid prototyping for multilingual multi-user dialog systems. In: Text, Speech, and Dialogue: 19th International Conference, TSD 2016, Brno, Czech Republic, September 12-16, 2016, Proceedings 19. pp. 478–485. Springer (2016)
7. Jiang, R., Tan, Y.K., Limbu, D.K., Tung, T.A., Li, H.: A configurable dialogue platform for asoro robots. Proc of APSIPA ASC (2011)
8. Jurčíček, F., Dušek, O., Plátek, O., Žilka, L.: Alex: A statistical dialogue systems framework. In: Text, Speech and Dialogue: 17th International Conference, TSD 2014, Brno, Czech Republic, September 8-12, 2014. Proceedings 17. pp. 587–594. Springer (2014)

9. Kath, H., Lüers, B., Gouvêa, T.S., Sonntag, D.: Lost in dialogue: A review and categorisation of current dialogue system approaches and technical solutions. In: German Conference on Artificial Intelligence. pp. 98–113. Springer (2023)
10. Kerminen, A., Jokinen, K.: Distributed dialogue management in a blackboard architecture. In: Proceedings of the 2003 EACL Workshop on Dialogue Systems: interaction, adaptation and styes of management (2003)
11. Kleer, N., Rekrut, M., Wolter, J., Schwartz, T., Feld, M.: A multimodal teach-in approach to the pick-and-place problem in human-robot collaboration. In: Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. p. 81–85. HRI '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3568294.3580047
12. Körber, Y., Hahn, V., Moniri, M.M., Schwartz, T., Feld, M.: Madmacs-multiadaptive dialogue management in cyber-physical environments. In: 2017 International Conference on Intelligent Environments (IE). pp. 184–187. IEEE (2017). https://doi.org/10.1109/IE.2017.33
13. Lager, T.: Statecharts and scxml for dialogue management. In: Habernal, I., Matoušek, V. (eds.) Text, Speech, and Dialogue. pp. 35–35. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
14. Li, C.Y., Ortega, D., Väth, D., Lux, F., Vanderlyn, L., Schmidt, M., Neumann, M., Völkel, M., Denisov, P., Jenne, S., Karacevic, Z., Vu, N.T.: Adviser: A toolkit for developing multi-modal, multi-domain and socially-engaged conversational agents. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2020)
15. Mouromtsev, D., Kovriguina, L., Emelyanov, Y., Pavlov, D., Shipilo, A.: From spoken language to ontology-driven dialogue management. In: Král, P., Matoušek, V. (eds.) Text, Speech, and Dialogue. pp. 542–550. Springer International Publishing, Cham (2015)
16. Mouromtsev, D., Kovriguina, L., Emelyanov, Y., Pavlov, D., Shipilo, A.: From spoken language to ontology-driven dialogue management. In: Text, Speech, and Dialogue: 18th International Conference, TSD 2015, Pilsen, Czech Republic, September 14-17, 2015, Proceedings 18. pp. 542–550. Springer (2015)
17. Neßelrath, R.: SiAM-dp: An open development platform for massively multimodal dialogue systems in cyber-physical environments. Ph.D. thesis, Saarland Informatics Campus, Saarland University (2015)
18. Neßelrath, R., Feld, M.: Siam-dp: A platform for the model-based development of context-aware multimodal dialogue applications. In: 2014 International Conference on Intelligent Environments. pp. 162–169. IEEE (2014). https://doi.org/10.1109/IE.2014.31
19. Prange, A., Chikobava, M., Poller, P., Barz, M., Sonntag, D.: A multimodal dialogue system for medical decision support inside virtual reality. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue. pp. 23–26 (2017). https://doi.org/10.18653/v1/W17-5504
20. Rothkrantz, L.J., Wiggers, P., Flippo, F., Woei-A-Jin, D., van Vark, R.J.: Multimodal dialogue management. In: Text, Speech and Dialogue: 7th International Conference, TSD 2004, Brno, Czech Republic, September 8-11, 2004. Proceedings 7. pp. 621–628. Springer (2004)
21. Valenta, T., Švec, J., Šmídl, L.: Spoken dialogue system design in 3 weeks. In: Text, Speech and Dialogue: 15th International Conference, TSD 2012, Brno, Czech Republic, September 3-7, 2012. Proceedings 15. pp. 624–631. Springer (2012)