Design of D3driver

Bhargavi Mohan

February 5, 2021

1 Introduction

This document is to review the high-level architecture of the bot — D3driver. The name stands for Design Decision Documentation and since the bot drives or in other words motivates this specific goal, it is named as D3driver in short. D3driver will be implemented to be added in Microsoft Teams. It is designed in way that can be added to a channel. This document gives the reader an idea of the architecture plan of D3driver and a short technology review has also been done in the end.

2 Prerequisites

This section describes all the terminologies needed for further understanding

- Microsoft Teams— is a well known IT collaboration tool used for communication purposes. It provides options for chat, group chats, audio-video calling, file storage and third-party apps integration. Official documentation can be referred here[1]

- Microsoft Azure— is a cloud computing platform owned by Microsoft. Users are offered a wide variety of cloud services like computing, developing, analytic, storage, networking, deploying their applications in the cloud. Official documentation can be found here[2]

- MS teams Adaptive cards—Microsoft introduced the concept of cards to exchange the UI content in a consistent way. It is a simple JSON object that can rendered within a bot. There are 3 types of cards of which adaptive cards are used for implementing D3driver. These cards can be used in bots that can either be part of bot communication or the users are also able to exchange cards with other members. There are abundant formatting options for styling the text inside the card to make it look impact-full. Please find more about adaptive cards here[3]

[1] https://www.microsoft.com/en/microsoft-teams/group-chat-software
[2] https://docs.microsoft.com/en-us/azure/?product=featured
[3] https://adaptivecards.io/

- V-model processes— It is the widely known and highly accepted model for developmental process. A team like Mechatronics that involves diverse engineering groups to build a system use this model as a guide in the different phases of development. It showcases the sequence of tasks to be carried out in each phase. The paper [1] explains the latest theories and concepts about the model.

3 D3driver logo

Presenting the logo for D3driver in figure 1. The bot has successfully gathers all the design decisions made during the conversation. Hence the logo exemplifies the documentation of all the decisions discussed.

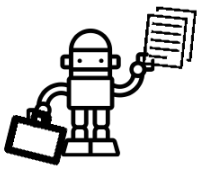

Figure 1: Logo of D3driver

4 Idea and Approach

In this section, the bot methodology is explained. The primary goal of the bot is to capture all the design related decisions during a conversation among the team members of an inter-disciplinary team. The main subject that is, documenting the most important conversations and segregating the messages in an asynchronous group chat. The idea has now been narrowed down to just document the design decisions. The users will invoke the bot in the simplest way possible only when they want to discuss design decisions during a conversation.

The design phase in any project may vary according to different V-model processes in an organization. The V-model that is used as a reference architecture to product life cycle management has multiple versions. The interpretations of V-models are well explained in [2]. This paper summarizes different systems using different standards that are well recognized and accepted globally. The 3 V-model processes has been chosen in this thesis implementation. They are the V-models used by 1) Domain specific systems, 2) INCOSE systems and 3)Model-based systems. The design phases in each of these standards change accordingly hence each phase would correspond to it's own design decisions.

The approach here is to sort the design decisions according to the respective design phases and are as follows

- Domain specific systems
  - Mechanical design phase hence Mechanical design decisions
  - Electrical design phase hence Electrical design decisions
  - Software design phase hence Software design decisions
- V-model for INCOSE systems
  - Design definition phase hence Design definition decisions
  - Design characteristics & Enablers phase hence Design characteristics & Enablers decisions
  - Design Alternatives phase hence Design Alternatives decisions
- V-model for Model-based systems
  - Functional design phase hence Functional design decisions
  - Logical design phase hence Logical design decisions
  - Physical design phase hence Physical design decisions

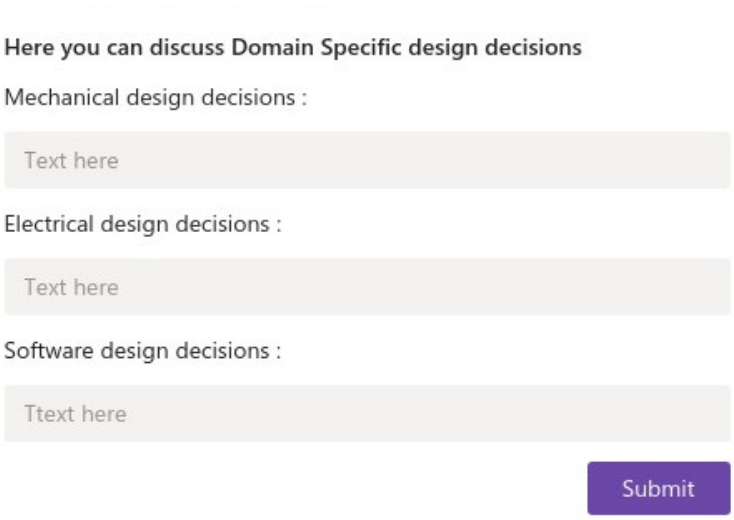

Figure 2: Example adaptive card

After submitting the decision cards in the group, all these are stored in the database for later extraction. The idea is to implement a D3driver tab. This tab will have a display of all the design decisions, the member who took the decision, type of decision and the date of decision. This way, any new member of the group can have an access to all the older decision that was taken in their absence.

Apart from this, the evaluation of the existing bots has shown that the bot is invoked every time any team member wants to make any decision. This means that the user either is required to type a long and complicated command that is a combination of many ASCII characters, or the users need to make around 2 mouse clicks. These are the steps users need to perform in order to invoke a bot i.e. a way to let the bot know that it needs to now start documenting the discussions(decisions). The long commands and many mouse clicks could become tiresome and hence have to be made tireless. Since the policies of Microsoft teams would not allow any changes pertaining to it's user interface, it's not yet possible to make it absolutely tireless to invoke the bot. Therefore, an attempt is made to call the bot with just one mouse click. The idea here is to make the routine more effortless yet impact-full. Therefore, Microsoft teams offer messaging extensions. This can be added as a static button along with the other default buttons.

Furthermore, on click of this button, an adaptive card will pop up as shown in the figure 2. There are dedicated text-boxes to discuss design related decisions.

5 High-level architecture

The figure 3 represents the high-level architecture of D3driver. The components of the architecture diagram are as follows

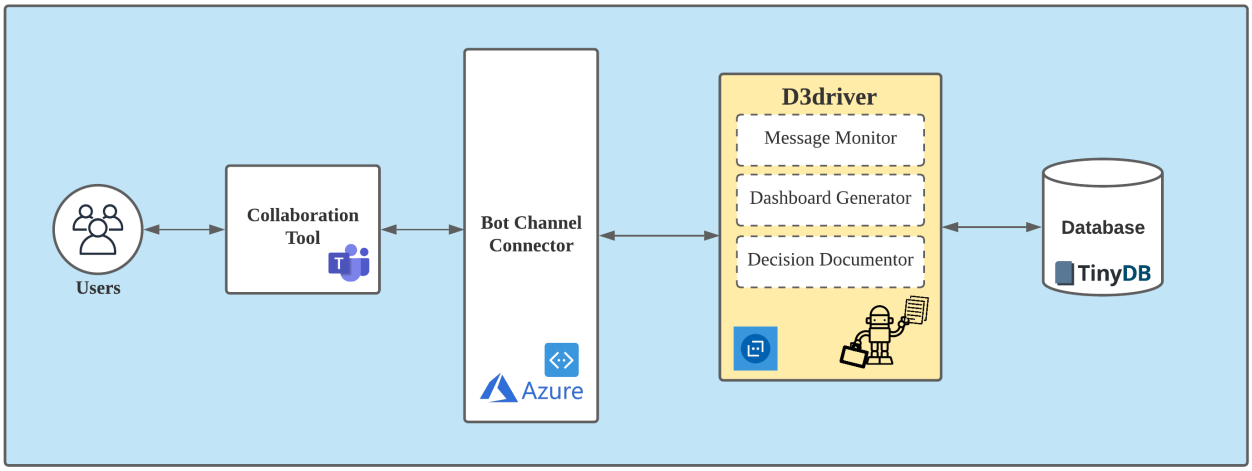

Figure 3: High-level bot architecture

- **Users**— They symbolize the group members of a collaboration tool. The users converse within the group members via the collaboration tool.
- **Collaboration tool**— Microsoft Teams is picked as the collaboration tool for this implementation. This tool helps users to communicate with each other by means of allowing them to form teams and channels. This tool also has a rich set of

conversational, productive bots that aid the users in their routine activities. This tool serves as a medium for sending and receiving messages for the users.

Note: MS teams will come into picture once again after the bot has been hosted by Microsoft Azure. Teams is responsible for creating an important file along with logo files to finally use the bot. This details will be discussed in the implementation details section.

- **Channel connector**— A connection between the Microsoft teams and D3driver is nothing but the channel connector. To be able to deploy D3driver in the bot studio of MS teams, one must register it with Microsoft Azure bot service. Microsoft Azure has a set of approved channels. The developer must set the bot up with the appropriate channel available in Microsoft Azure. In this case, the bot should be configured with Microsoft Teams. This means the D3driver will be able to talk to users via Microsoft Teams. This connector is responsible for forwarding user's incoming messages to the bot's endpoint appended with the standard endpoint of MS Teams i.e, **/api/messages**.

  Here is an official documentation[4] on how to structure bots with channels with the help of Microsoft Azure

- **Bot Framework v4**— Azure Bot Service offers a complete toolkit required to build bots, for example the popular Bot framework SDK. This framework is the main element for developing a bot. The framework provides many schemas and this implementation will use cards schema. Card schema is the representation of interactive adaptive cards in the application-level, that can be used inside a group chat. After the bot development, the developer is required to create a new bot service by providing all the desired data. Information like name of the bot,developer location, Bot template(v4), Bot ID and password etc should all be mentioned during the registration. This will help Azure to create the bot service and deploy the bot to the cloud. After successful deployement, the developer will prepare all the necessary code files to deploy. All the files should be zipped up i.e, the python files(app.py), dependency files(requirement.txt) etc. The deployment and the channel connector configurations should make the bot available to MS teams bot studio.

  Once again the official documentation for this can be seen here[5]

- **Database**— This is the storage component in the architecture. This serves as the persistency layers for D3driver. The incoming decisions should be stored for later retrieval and hence it would require a simple database for that purpose. TinyDB is written in python and can be easily extended to make use of custom storage. The main function of the database is to store the details like channel name, sender's name, decision name and date etc. It can be devised to store any schema that a

[4]https://docs.microsoft.com/en-us/azure/bot-service/bot-service-manage-channels?view=azure-bot-service-4.0

[5]https://docs.microsoft.com/en-us/azure/bot-service/abs-quickstart?view=azure-bot-service-4.0

developer would design. After storing, it can be queried to display any data the developer wants to display on the D3driver tab that is discussed in section 4

5.1 Work flow of the D3driver

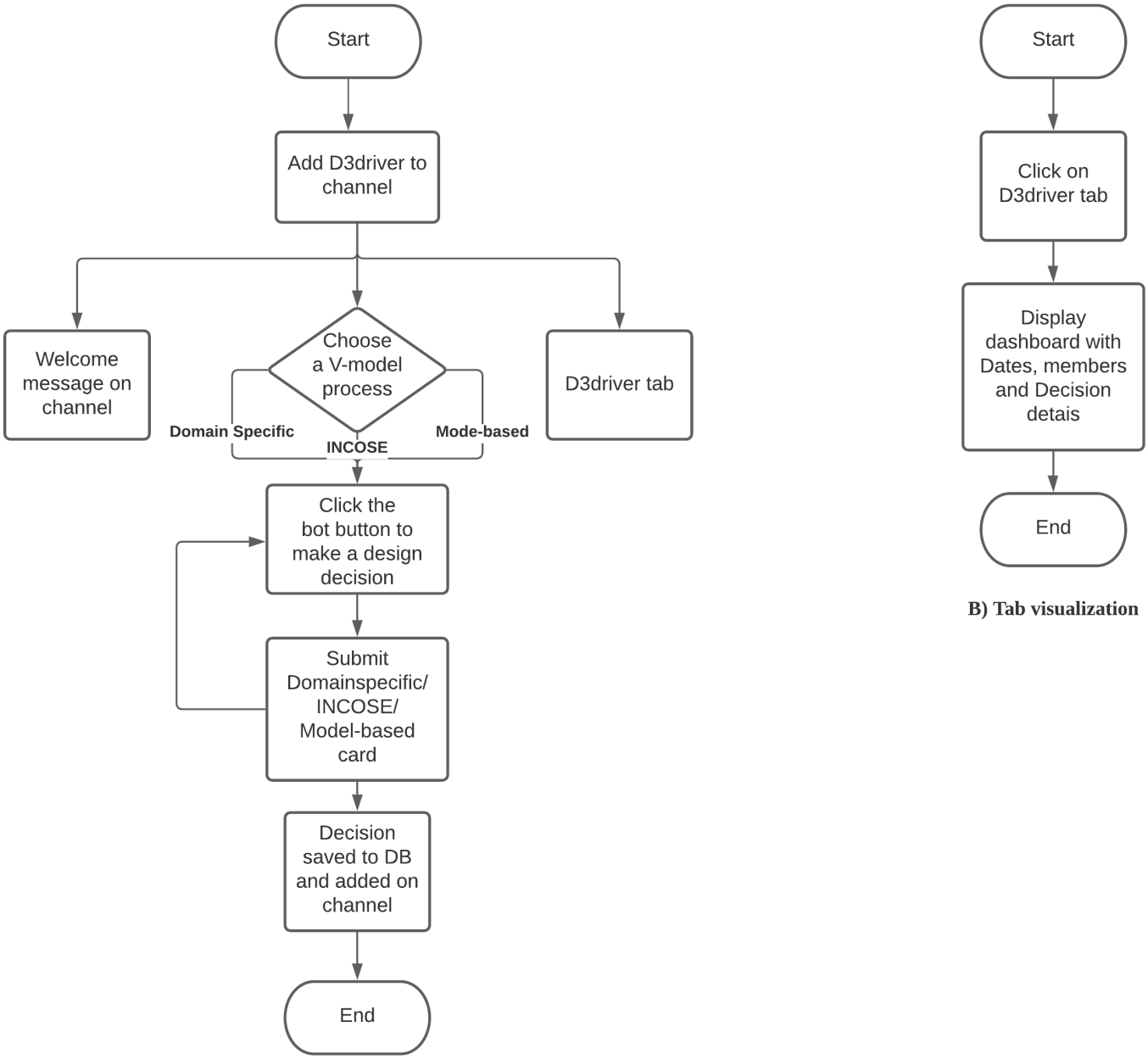

Figure 4: Workflow of the D3driver

This section describes the step-by-step working of the D3driver by means of a flowchart as shown in figure 4. There are 2 different flowcharts depicted. The left flowchart denotes the main working procedure of the D3driver. The bot after it's implementation will be made available in Microsoft teams' bot studio. When a user clicks on D3driver to install, it gives an option to install it to a desired channel. Once the bot is installed, three activities will take place. 1) the D3driver message extension will be installed, D3driver tab will be installed and 3) a welcome card is sent only to that user who installs it. The D3driver will then offer V-model choices to that particular user. The user selects the suitable V-model process required by their project. This is a one-to-one conversation since only one user is allowed to choose the appropriate V-model standard. Although, the initial configuration takes place in a chat, the other activities will happen in the intended team.

After a V-model process is selected, the D3driver's message extension(button) can be found along with other default buttons below the message text-box. The button will always be present in the same location there unless the bot is uninstalled. Any user who wants to start a thread or wants to inform something related to design decisions, they can simply click on D3driver's button. Upon clicking, an adaptive card pops up with three other input fields and their names are dependent on the V-model process chosen for the team/project. The user can make use of those fields to discuss design decision and click on the submit button within the card. This way, all design decisions are sorted from the other conversation materials in the group chat.Once the submit button is hit, the card appears on the group. It will have the sender's name and the decision name. All these cards are saved into the database.Similarly, the D3driver's button can be clicked as and when design decisions are to be discussed.

The right most flowchart denotes the visualization of all the saved decisions. The purpose of having a D3driver tab is to display all the decisions in one place. After all the card are saved to the database, the details can be rendered as a html page on the tab. The work flow of this is also pretty simple. Any user who wished to take a look at all the decisions that has been discussed can click on the designed D3driver tab. In this tab, a dashboard can be viewed. The preliminary design is to just generate the date, decision name associated with the sender's name and either a search box or filter button or both depending on the time left for it's implementation.

6 Technologies and Applications

The choice of technologies and applications that will be used for the implementation are listed in this section:

- Microsoft Teams— The D3driver app package is designed and developed in such a way that it can be integrated to any collaboration tool but this thesis implementation focuses on hosting the bot in Microsoft teams.

- Microsoft Azure— This cloud service seems to be the best to work with Microsoft teams and also the platform ensures a great collection of bot framework features.

The open-source bot framework SDK v4 is an ideal development toolkit for this implementation.

- ngrok[6]— is a tunneling tool that helps users to expose their local-host server to the internet. This is a free software that can be downloaded on the system. ngrok uses either the default HTTP port or an explicit port number can be mentioned such that ngrok exposes that local port.

- Python SDK version 3.7[7]— One of the popular programming languages that makes use of an interpreter making it an interpreted, high-level language. This is used as the main developing language for D3driver because it has simpler syntax, easy usage, can be executed faster and works well for prototyping software.

- TinyDB[8]— As discussed in the architecture plan, this database will be used as decision database for D3driver.

- HTML[9] & JavaScript[10]— D3driver tab used these two technologies to render the card summary and other details. HTML to display the card information and JavaScript to make the page interactive.

- Visual studio code[11]— This is the best code editor created by Microsoft. Excellent features for code debugging, code review integrated with version control system(github). The biggest advantage is the availability of "Microsoft UI toolkit" extension with visual studio code specially used for bot development and management. It has some internal configurations with Azure that is done automatically for users and makes the process easier and faster.

7 Conclusion

This document is the brief description of how the D3driver is designed. This design is supposed to be the solution to poor or no documentation in a multi-disciplinary team. The design of D3driver aims at motivating team members to distinguish the most important design decisions from the typical conversations that they would have among their team. Designed in the best possible way to balance the simplicity, usability and functionality of the bot. The design also exhibits flexibility to choose the type of V-model process standard that the team wants to document. The design in ease of using the card with just one click is also the highlight and finally a distinct tab to perceive all the decisions together shows that the design is meant to decrease any effort of going through the long threads of decisions in a channel.

[6]https://ngrok.com/docs
[7]https://www.python.org/
[8]https://tinydb.readthedocs.io/en/latest/
[9]https://www.w3.org/TR/html52/
[10]https://devdocs.io/javascript/
[11]https://code.visualstudio.com/

Additionally, there are also a couple of ideas to make the design much more user friendly. This will be decided on the time left for implementation.They are

- Giving an option of ”custom” along with the other V-model process and introducing ”custom” sub-options.
- Designing graphs and statistics in the D3driver tab to see the number of decisions, highest number of decisions taken by a particular member and a few more useful facts and figures.

Any modifications in the existing or upcoming design will be mentioned in the final thesis document.

Next action item in the pipeline is straightforward, diving into the most important phase of the thesis that is Implementation, details of which will be provided in the implementation document

References

[1] Iris Graessler and Julian Hentze. The new v-model of vdi 2206 and its validation. *at-Automatisierungstechnik*, 68(5):312–324, 2020.

[2] Iris Gräßler, Julian Hentze, Tobias Bruckmann, et al. V-models for interdisciplinary systems engineering. In *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, pages 747–756, 2018.