

A picture containing text, sky, light, flock

Description automatically generated

Chart

Description automatically generated

After all model diagram has been made, we now must verify our model. Model verification is a methodology for developing computational models. This model can be used to make engineering predictions with quantifiable confidence. Model verification and validation are extremely important in today's industry. This model has influenced many aspects, including time consumption, cost, and many others. Verification and Validation of Simulation Models are performed using a variety of techniques. In this case, TAPAAL simulation is being used to validate our Precision farming system. Based on fig))))), we converted all the diagrams to the TAPAAL model. Before delving deeper into the flow of our Precision farming system, some key information about the TAPAAL model is provided. The circles are called places, and the rectangles are called transitions. They are linked together by either standard arcs between places and transitions or transport arcs. Tokens govern the net's behavior. Each location can hold a certain number of tokens, and each token has its own real-time age.

The concept of our model is divided into two major parts, detecting problems such as disease on plants and weeds and detecting the location of the drone. The drone is divided into 5 small components which are Receiver, Motor, Detector, Localizer and Transmitter as shown in fig)))))). It should be noted that this scenario takes place in the middle of the day and not when the system is idle. Consider how the drone detects a problem on a field. de Wait place which is located at Receiver component will fire in the to receive transition with one token. The translate place will then fire the token get path and get purpose transitions. Motor's component shares the get path transition. The token will fire to move to target location. Meanwhile, the get purpose transition is shared with the component of Detector. If the Detector detects a problem on the field, problem detected transition will fire the token report place. In other words , the Detector will report the problem to the transmitter, who will then send the data to the Problem handler. According to figPH)))))), the first place is a wait place, which consists of two tokens. Tokens must be fired separately to the problem accepted and location accepted transitions. As a result, after the Problem handler accepts the problem, it must wait for the Localizer component to send the coordinates to the Transmitter. The data will then be sent to the Problem handler by the Transmitter. After both tokens are fired into their respective transitions, it is important to the Problem handler to find a suitable solution and send the coordinate to the Navigator. When the Navigator receive the location, it will fire a token to calculate path place. Given that they have a shared transition, the calculated\_path transition is returned to the Problem handler. Both tokens from solution found transition and path calculated transition will fire to compile data place. This time, the total number of tokens will be two. Compile data will then fire both tokens to the accessing data transition, which is shared with the Data storage component. The token will be fired to storing data place during the accessing data transition. Furthermore, the Problem handler will verify the data and fire the token to the assigning\_actuator transition. The Actuator will solve the problem using the discovered solutions and calculated path given from Problem handler. In the Actuator component, when the problem is resolved, the token is sent to the task\_completed transition as proof that the actuator has taken care of the issue. Finally, both tokens will return to the wait state until the drone detects a new upcoming problem.

After modeling our Precision farming system with TAPAAL, we ran a few queries to ensure and verify the correctness of ur system.  First, we determine whether the deadlock is likely to occur during all processes. As shown in fig)))))) there is no deadlock, and the result is satisfied. The number of tokens in the Navigator component is also checked, with at least one token in each place. Next, the send location to navigator and wait for path places are checked and verified to ensure that no token is present at the same time. Furthermore, during program execution, the token in the Actuator component, assigning\_actuator place and the Data storage component, storing\_data place must not be equal to 1 or greater than 2. The outcome is also satisfactory.

Graphical user interface, application

Description automatically generated