Student: Lars van Duijnhoven

Studentnummer: 2103948

Docent: Bram Knippenberg

Datum: 07-11-2024 om 10:00 tot 12:00

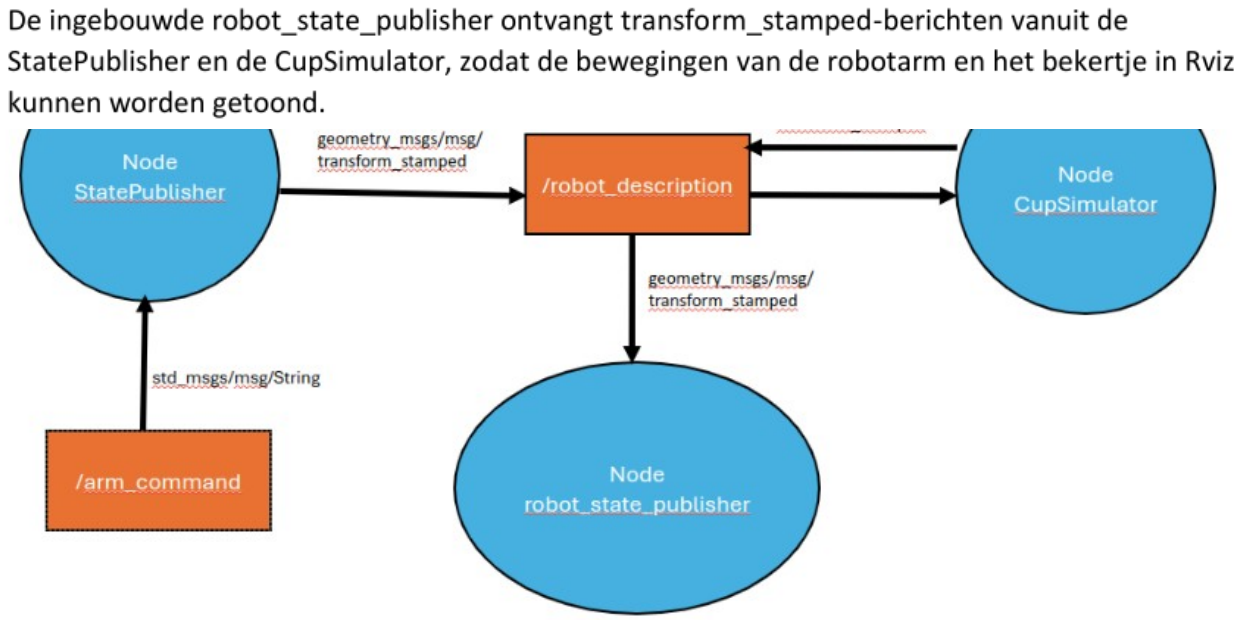
Course: World of Robots - World

Ik heb de inlevering gereviewed van Max Janssen, Studentnummer: 1658262.

1. Review op het ontwerp

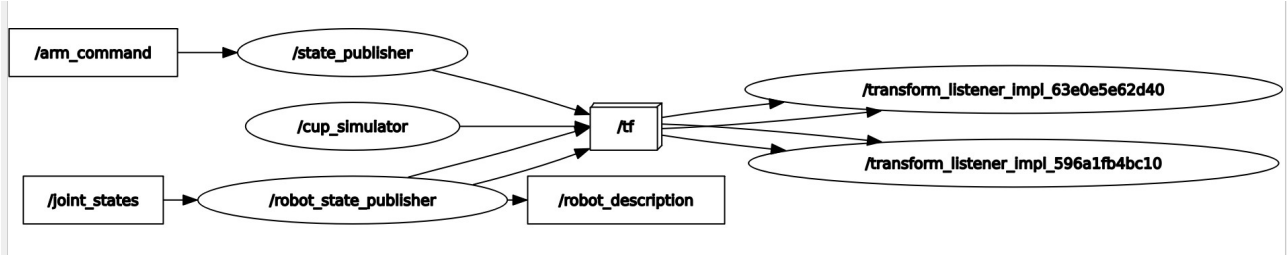
a. *Geef van één aspect een onderbouwde mening over de kwaliteit. Onderbouw dat met argumenten en een verwijzing naar voorbeelden.*

(één aspect is vaag :/) Als aspect heb ik gekozen om de nodes en topics te beoordelen. Mijn mening over de kwaliteit is hier dat het beter had gekund, omdat de namen nu niet kloppen bij de functionaliteit.
Neem als voorbeeld de message die van CupSimulator naar robot_state_publisher gaat:



Waarom wordt de locatie en de transform van de CupSimulator doorgestuurd naar de robot_state_publisher? Robot_state_publisher geeft aan mij aan dat het de state van de robot publisht, maar in dit geval publisht het de state van alle objecten in RVIZ, wat apart is. Dit zou beter kunnen door direct vanuit CupSimulator naar de TF te publishen, want dat is wat robot_state_publisher ook doet.
Ik vind het dan ook raar dat er niet weergeven wordt dat er naar TF gepublisht wordt, omdat dit wel belangrijke informatie kan zijn voor de lezer.

Na het schrijven van de bovenstaande stuk tekst heb ik rqt_graph geopend, maar dit gaf een heel ander resultaat, namelijk:



Eigenlijk wordt de robot_state_publisher dus helemaal niet gebruikt om de transforms af te handelen, maar dit heb ik niet uit de nodes van de documentatie kunnen halen.

b. *Geef aan in hoeverre jij dit ontwerp bruikbaar acht en waarom.*

Dit ontwerp is zeker wel bruikbaar en zou best kunnen werken, maar het is niet de meest handige manier en er zijn dus wel meerdere flaws. Verder is de onnodig hoger dan dat het had kunnen zijn, omdat alles dus via de robot_state_publisher gaat ipv apart naar TF.

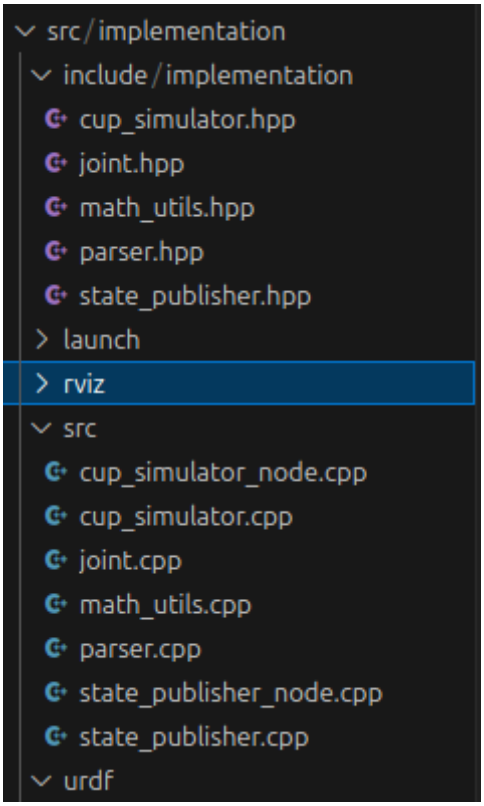
Na het erachter komen dat de nodes en topics heel anders met elkaar communiceerde vind ik het wel logischer gedaan en alsnog bruikbaar. Wel had ik state_publisher niet zelf toegevoegd en gewoon joint_states gebruikt om via de robot_state_publisher te publishen naar TF.

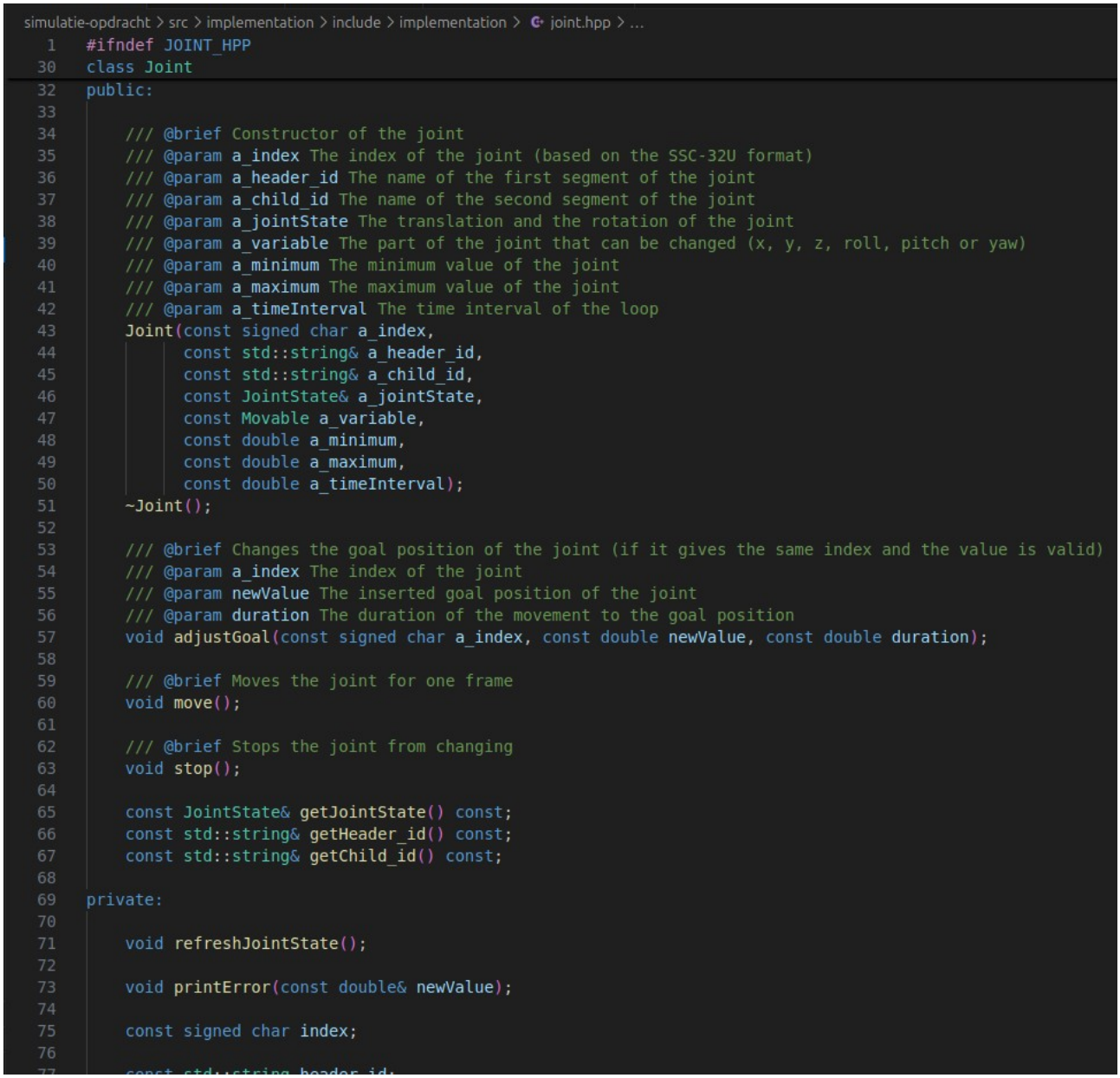
2. Review op de code

a. Geef van enkele aspecten van de code aan wat je heel goed vindt en waarom. Onderbouw dat met argumenten en een verwijzing naar een voorbeeld (bestandsnaam en regelnum- mers). Suggestie: de course World of Robots is een verdiepend semester, met een oordeel over alleen zaken als codeopmaak, camel case, hardcoded values, et cetera laat je niet zien dat je een verdiepend oordeel kunt geven.

Ik vind het goed dat alle code gerunt kan worden vanuit een python launch file, zodat ik niet op meerdere terminals iets moet openen. Nu is het gewoon `ros2 launch implementation display.launch.py`
en dan wordt de arm én de cup ingeladen in Rviz.

Het is heel leuk gedaan dat er specifiek in de demo een paar keer de beker wordt gedraaid over een as. Dit ziet er gaaf uit en laat zien dat er overnagedacht is van wat er in de demo zou komen.
(review op de demo code :))





```
simulatie-opdracht > src > implementation > include > implementation > joint.hpp > ...
#ifndef JOINT_HPP
class Joint
public:

    /// @brief Constructor of the joint
    /// @param a_index The index of the joint (based on the SSC-32U format)
    /// @param a_header_id The name of the first segment of the joint
    /// @param a_child_id The name of the second segment of the joint
    /// @param a_jointState The translation and the rotation of the joint
    /// @param a_variable The part of the joint that can be changed (x, y, z, roll, pitch or yaw)
    /// @param a_minimum The minimum value of the joint
    /// @param a_maximum The maximum value of the joint
    /// @param a_timeInterval The time interval of the loop
    Joint(const signed char a_index,
          const std::string& a_header_id,
          const std::string& a_child_id,
          const JointState& a_jointState,
          const Movable a_variable,
          const double a_minimum,
          const double a_maximum,
          const double a_timeInterval);
    ~Joint();

    /// @brief Changes the goal position of the joint (if it gives the same index and the value is valid)
    /// @param a_index The index of the joint
    /// @param newValue The inserted goal position of the joint
    /// @param duration The duration of the movement to the goal position
    void adjustGoal(const signed char a_index, const double newValue, const double duration);

    /// @brief Moves the joint for one frame
    void move();

    /// @brief Stops the joint from changing
    void stop();

    const JointState& getJointState() const;
    const std::string& getHeader_id() const;
    const std::string& getChild_id() const;

private:

    void refreshJointState();

    void printError(const double& newValue);

    const signed char index;

    const std::string header_id;
```

De code is goed gescheiden door header files apart te houden in een include, dit is netjes!
Zoals hier op de afbeelding te zien is zijn alle header files voorzien van Doxygen, wat een pluspunt is voor de kwaliteit, maar komt nog bovenop dat er specifiek aandacht is besteed aan het private maken van functies die niet buiten de klasse gebruikt worden. Dit zorgt voor betere encapsulatie van de code, iets wat bij alle bestanden zo is. Zo zijn bijvoorbeeld in CupSimulator alleen de contstructor en deconstructor public, terwijl de rest van de functies allemaal private zijn, wat dus een goed voorbeeld is van encapsulatie.

Verder kreeg ik geen valide warnings of errors van cppcheck met dit commando:
`cppcheck --enable=all --inconclusive --force --inline-suppr --std=c++17 --suppress=missingIncludeSystem ./src/`
Wat dus ook goed is, omdat dit betekent dat cppcheck geen errors kan vinden in de code en dus de code hierdoor valideerd.

```
 Robot initialized
[rviz2-4] QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-j
enkins'
[rviz2-4] [INFO] [1730973424.693559183] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-4] [INFO] [1730973424.693693053] [rviz2]: OpenGl version: 4.6 (GLSL 4.6)
[rviz2-4] [INFO] [1730973424.707567283] [rviz2]: Stereo is NOT SUPPORTED
[state_publisher-2] The pwm for joint number 0 (base_link2turret) is out of rang
e. It has to be between 500 and 2500, so 0 is invalid.
[state_publisher-2] '-10' is not a number.
[state_publisher-2] '-10' is not a number.
[state_publisher-2] The pwm for joint number 4 (hand2gripper_left) is out of ran
ge. It has to be between 500 and 2500, so 10 is invalid.
[state_publisher-2] The pwm for joint number 4 (hand2gripper_right) is out of ra
nge. It has to be between 500 and 2500, so 10 is invalid.
[state_publisher-2] You should have one 'P' after each hashtag.
[state_publisher-2] You should have one 'P' after each hashtag.
[state_publisher-2] The part after the 'T' is either empty, or it isn't numeric.
```

Na het testen van de limits ben ik erachter gekomen dat dit goed afgehandeld wordt, op T0 na, maar dit komt hieronder nog.
Ik heb negatieve-, te lage- en te hoge getallen geprobeerd en alles werd afgehandeld voor P(osition).
Zelfs duplicate inputs werden afgehandeld! Dus 2 keer servo 2 aansturen bijvoorbeeld. Dit is goed gedaan, omdat er anders unexpected behaviour voor kon komen, bijvoorbeeld dat de grippers extreem ver uit elkaar zouden kunnen gaan.

b. Geef op eenzelfde wijze enkele aspecten van de code aan die je niet goed vindt en waarom
niet. Onderbouw dat met argumenten en een verwijzing naar een voorbeeld (bestandsnaam en regelnummers)

```
Vanuit deze directory kun je in de folder genaamd 'rviz' een bestand vinden die 'urdf.rviz' heet.
Verder kun je in de folder genaamd 'urdf' de urdf-files vinden van de cup en de lynxmotion_arm.

In regels 40 en 152 van urdf.rviz zijn de volgende regels te vinden:

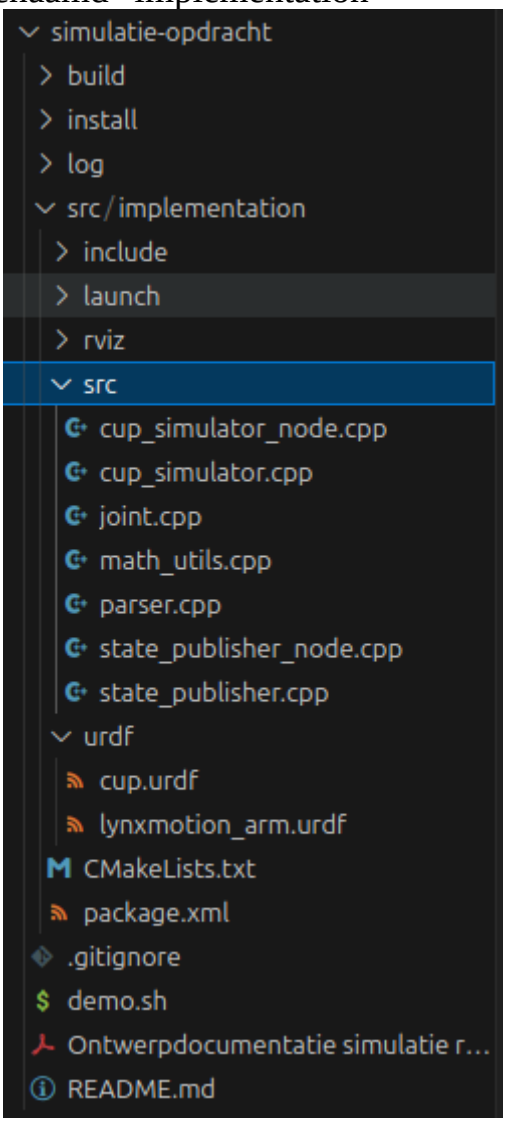
```bash
Regel 40:
Description File: /home/max/Documents/WoR/simulatie/simulatie-opdracht/src/implementation/urdf/lynxmotion_arm.urdf

Regel 152
Description File: /home/max/Documents/WoR/simulatie/simulatie-opdracht/src/implementation/urdf/cup.urdf
```
```

Dit is het eerste wat in de readme.md staat op regels 20 en 23, de bestanden konden veel beter met ~/ beginnen zodat je niet alles van de home directory zou moeten pakken. Verder zijn er ook betere manieren voor om dit vanuit de hoofd directory te pakken, namelijk:
`./src/implementation/urdf/lynxmotion_arm.urdf`
en
`./src/implementation/urdf/cup.urdf`

Dit zou altijd werken en zo hoeft de gebruiker het niet handmatig aan te passen.

Alles staat in 1 package genaamd “implementation”



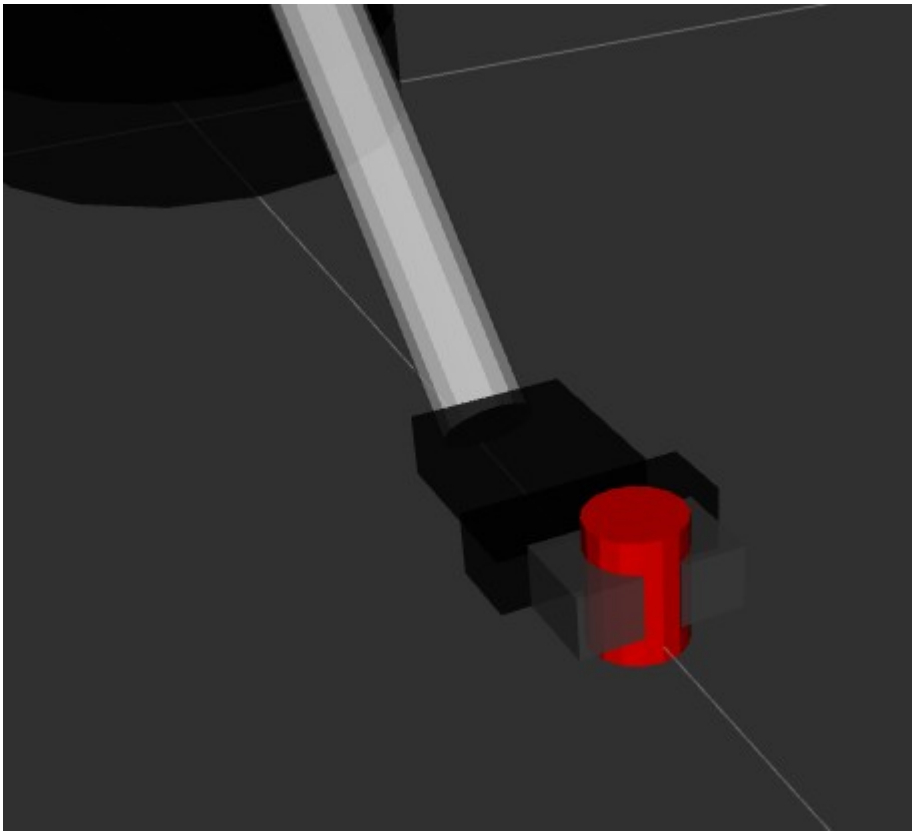

```
Starting >>> implementation
Finished <<< implementation [12.6s]

Summary: 1 package finished [12.7s]
```

Dit had beter in verschillende packages gekund, zodat je beter aan de ROS-directory structuur voldoet. Denk hierbij aan een Cup package en een robot package, dat had dit al beter gemaakt. Helaas is het ook onduidelijk dat de robotarm code in state_publisher zit, omdat de naam niet overeenkomt met wat er in staat.

```
jenkins@0ddcba0ddc99:/home/lars/Downloads/materiaal/WORLDA05_TOETS-02_Max Jansse
n_03-04-2024/simulatie-opdracht/src$ . install/setup.bash
bash: install/setup.bash: No such file or directory
jenkins@0ddcba0ddc99:/home/lars/Downloads/materiaal/WORLDA05_TOETS-02_Max Jansse
n_03-04-2024/simulatie-opdracht/src$ ros2 launch implementation display.launch.p
y
Package 'implementation' not found: "package 'implementation' not found, searchi
ng: ['/opt/ros/jazzy']"
jenkins@0ddcba0ddc99:/home/lars/Downloads/materiaal/WORLDA05_TOETS-02_Max Jansse
n_03-04-2024/simulatie-opdracht/src$
```

Verder staat er ook in de readme op regel 35 dat je . install/setup.bash moet runnen in elke terminal, maar niet waar in de mapstructuur je dit moet doen. Hierboven, zoals je ziet, zit ik in de src map en werkt het niet, omdat dit verkeerd aangegeven staat.



Hiernaast is het niet zichtbaar wanneer de beker vastgepakt wordt, zoals in de foto. Hij is hier wel vastgepakt, maar dit is dus niet zichtbaar :/. Dit had voorkomen kunnen worden door bijvoorbeeld de kleur van de URDF aan te passen als de beker opgepakt is.

Hier komt ook nog bij dat er geen minimale tijd is ingesteld in de code, dus als er een commando met T0 komt teleporteerd de arm naar die plek, wat geen realistische simulatie is. Dit had voorkomen kunnen worden door een minimale waarde aan tijd te geven, zodat het in ieder geval zichtbaar is.

Hiernaast staat in de documentatie op de laatste pagina dat de aansturing volgens het SSC-32U formaat kan worden gestuurd. Zie dit hieronder.

> **Multiple servo command (a.k.a. "Command Group")**
> The SSC-32U allows for multiple commands to be received in the same string. For servo position, this may look like:
>
> **# <ch> P <pw> S <*spd*> ... # <ch> P <pw> S <*spd*> T <time> <cr>**
>
> Commands of different types cannot be mixed in the same command group. Note that the <time> function should be placed at the end of the line and is associated with the entire move, whereas the <speed> command can be associated with each servo.

Helaas toen ik S100 probeerde toe te voegen kreeg ik dit:

```
[state_publisher-2] '2200S100' is not a number.
[state_publisher-2] '2200S100' is not a number.
[state_publisher-2] '2200S100' is not a number.
```

Wat dus betekent dat de strings gewoon samengevoegd worden ipv dat ze daadwerkelijk geparsed worden en er gekeken wordt of het wel echt een P is of juist iets anders. Dit klopte toen ik in de code ging kijken, nadat ik S2200 ingevuld had en ik onderstaande error kreeg:

```
// Error if there is less or more than one 'P' after the hashtag
if (std::count_if(command.begin(), command.end(), [](unsigned char c) { return c == 'P'; }) != 1)
{
    std::cout << "You should have one 'P' after each hashtag." << std::endl;
    return false;
}
```

(Dit stond in parser.cpp vanaf regel 80)

Dit is erg gevaarlijk om de strings zo te controleren, omdat, zoals hierboven het geval was, alles wat er na die P komt gewoon in de string gegooid wordt.
Graag had ik ook gezien dat ik geen servo 100 aan kon sturen, omdat deze niet bestond. Nu lijkt het namelijk alsof er een servo 100 is die ik aan kan sturen, als ik niet beter zou weten. Dit had voorkomen kunnen worden met een simpele check.

Daarnaast is er een aparte URDF gemaakt voor de beker, wat niet logisch is. URDF staat namelijk voor "Unified Robot Description Format" en de laatste keer dat ik gekeken heb, was mijn beker geen robot. Verder is URDF vooral handig voor meerdere joints en hoeken, waar de beker er geen van heeft. Ik had dit met bijvoorbeeld een marker gedaan, omdat dat logischer is qua indeling.

Als laatste ga ik toch maar even zeiken over de includes. In state_publisher.cpp staan sstream en algorithm geïnclude, terwijl deze gewoon weg kunnen en alles nog steeds goed werkt.
Zo zijn in de state_publisher.hpp ook de chrono, functional, memory, array en vector includes onnodig.
Ook in cup_simulator.hpp zijn de chrono, functional en memory includes onnodig.

3. Samenhang tussen ontwerp en code

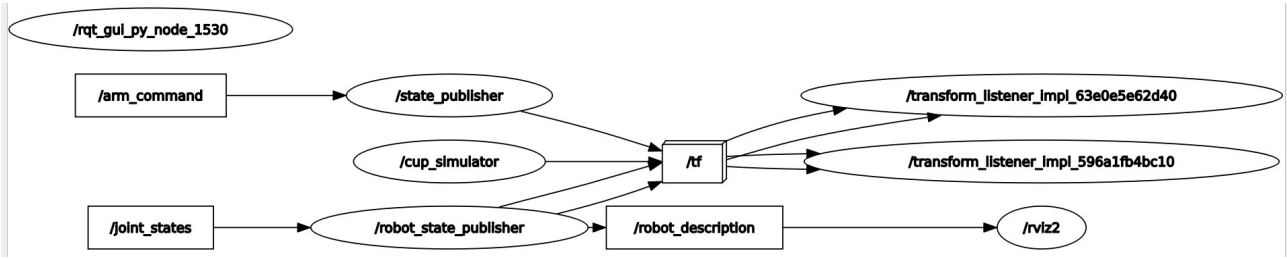
Geef aan in hoeverre de ontwerpen en code samenhagen en wat je daarvan vindt.

Zoals in hoofdstuk 1 al verteld is, komen het ontwerp en de code niet altijd overeen. De diagrammen kloppen wel, alleen de tekst die er soms bij staat ook niet, bijvoorbeeld bij het component diagram.
Er staat dat CupSimulator een interface biedt aan robot_state_publisher met de cup_transform, maar zoals in rqt_graph stond bij hoofdstuk 1 zijn deze onafhankelijk van elkaar.
Verder haal ik uit het component diagram dat de State_Publisher een interface biedt aan CupSimulator, maar dit komt in de daadwerkelijke code nergens voor.
Wel wordt er een listener aangemaakt, maar deze wordt nooit gebruikt. Zie cup_simulator.cpp op regel 10, de tf_listener. Als je deze CTRL + F zoekt dan wordt hij niet nogmaals gebruikt.

Daarnaast wordt er ook nog verteld dat de posities van de beker en de arm naar rviz2 gepublished worden, zie hier H1.1 van de documentatie:

> De StatePublisher en de CupSimulator geven respectievelijk hun posities mee aan de robot_state_publisher, om deze vervolgens mee te geven aan Rviz2, die de bewegingen van de robotarm en het bekertje toont.

Terwijl Rviz2 alleen maar luistert naar robot_description, waar alleen robot_state_publisher gebruik van maakt :/, hier had de topic best bij vermeld kunnen worden. Verder worden deze posities nooit naar robot_state_publisher gepublished en dus ook niet naar rviz2, dus de bewegingen worden ergens anders door getoond.



Kortom, de samenhang tussen het ontwerp en de code is minimaal.

4. Advies bruikbaarheid voor vervolgproject

Geef een beargumenteerd advies of dit werk bruikbaar is om in een ander project te gebruiken.
Bijvoorbeeld voor een project waar deze robotarm ingezet gaat worden, maar er eerst middels simulatie getest moet worden.

Ja, deze simulatie is bruikbaar voor een volgend project om alleen te testen, omdat de daadwerkelijke implementatie voldoet aan de eisen en redelijk werkt. Mijn advies zou wel zijn om de maximale speed nog te implementeren, zodat de simulatie realistischer wordt en er uitgebreider gesimuleerd kan worden. Ik ga er dan wel vanuit dat de documentatie minder van belang is en het volgende projectteam dit nog wel kan ontcijferen met rqt_graph.
Al zou ik het een cijfer moeten geven van hoe sterk ik dit zou aanraden voor het genoemde voorbeeld (alleen om te testen) zou dit een 6 zijn. Het heeft zijn dingetjes en is niet handig om te gebruiken, maar alleen om te testen werkt het goed genoeg, mits er geen gebruik gemaakt hoeft te worden van individuele servo speeds.

Zou ik deze simulatie aanraden als eindproduct? Absoluut niet, er zitten nog veel fouten in niet functionele dingen, zoals bijvoorbeeld een ontwerp dat niet overeenkomt met de code. Wel zou het een redelijke basis zijn om op te starten, alleen zullen er veel dingen aangepast moeten worden, zoals geen beker URDF, geen aparte state_publisher als er al een is, bestandslocaties die niet dynamisch gemaakt zijn, etc.
Al zou ik het een cijfer moeten geven van hoe sterk ik dit zou aanraden voor het genoemde voorbeeld (als daadwerkelijk product) dit een 3 zijn, omdat er nog erg veel aanpassingen gedaan moeten worden, maar de basis is er wel.

Kortom, het doet wat het moet doen, maar het is heel onhandig ingericht, niet gebruiksvriendelijk en niet efficiënt gemaakt.