Would you like to continue?

	Percent	Respondents
Yes	98.7%	680
No	1.3%	9
Total	100.0%	689

For how long have you used simulation tool(s) to simulate a robot and/or used URDF files?

	Percent	Respondents
Less than 1 year.	12.9%	78
Between 1 and 5 years.	58.6%	356
Between 6 and 10 years.	17.0%	103
More than 10 years.	8.6%	52
Never.	3.0%	18
Total	100.0%	607

In which of these types of environments have you used simulation tool(s) to simulate a robot and/or used URDF files?

	Percent	Respondents
Academia	81.4%	472
Industry	51.0%	296
Government	4.0%	23
Unaffiliated groups (e.g., Open Source Robotics Foundation, a drone hobbyist group, a school club)	14.3%	83
Individual	24.3%	141
Other	0.5%	3
Total	100.0%	580

In which of these types of environments have you used simulation tool(s) to simulate a robot and/or used URDF files? - Other

- Student team
- Competitions
- Research projects

If you have used simulation tool(s) to simulate a robot and/or used URDF files at any organization, how large was the most recent organization?

	Percent	Respondents
1-10	26.5%	149
11-50	23.8%	134
51-100	9.9%	56
100+	39.8%	224

	Percent	Respondents
I haven’t used simulation tool(s) to simulate a robot and/or used URDF files at an organization	0.0%	0
Total	100.0%	563

If you have simulated a robot before, when was the last time you did this?

	Percent	Respondents
In the last month.	71.2%	398
In the last year.	23.3%	130
In the last 5 years.	4.3%	24
More than 5 years ago.	0.5%	3
I have not simulated a robot before.	0.7%	4
Total	100.0%	559

Why not?

	Percent	Respondents
Not applicable	50.0%	2
Too difficult	25.0%	1
Other	25.0%	1
Total	100.0%	4

Why not? - Other

- to young ^^

If you have simulated a robot with 3D visualizations, which tools have you used to do this?

	Percent	Respondents
CoppeliaSim (formerly V-REP)	24.1%	133
FlexSim	0.5%	3
Gazebo	82.2%	454
MATLAB/MathWorks	29.3%	162
Peter Corke's Robotics Toolbox (for MATLAB or Python)	22.6%	125
RoboDK	7.4%	41
RViz	77.2%	426
Unity	18.3%	101
Unreal Engine	9.4%	52
Visual Components	3.4%	19
Webots	12.7%	70
Custom simulator	12.3%	68
Other	20.7%	114
I have not simulated a robot with 3D visualizations.	0.2%	1

Total	100.0%	552

If you have simulated a robot with 3D visualizations, which tools have you used to do this? - Other

- gazebo
- Pinocchio and Meshcat
- Pybullet, MuJoCo
- Blender
- pybullet (Assistive Gym)
- NVIDIA issac
- Own implementation
- robwork studio
- Morse
- Isaac Sim
- pyBUllet, Nvidia Omniverse
- ThreeJS
- Dymola
- Julia Robotics
- Pybullet
- Meshcat (Julia robotics)
- PyBullet, Mujoco
- Foxglove, 20-sim, Blender
- Klamp‘t Python package
- Labview and Robot driver functions
- ABB robot studio
- PyElastica
- Godot
- MuJoCo
- MeshCat.jl, RigidBodyDynamics.jl
- Foxglove, 20-sim, Blender
- Delmia IGRIP;
- OpenRAVE, Pybullet
- Process simulate, Delmia , Robcad ,
- MuJoCo, Drake
- MUJOCO
- Drake / Meshcat
- Drake
- Pybullet
- OpenRave, Mujuco
- Bullet
- Nvidia Isaac Gym
- PyBullet
- Drake
- Drake
- Isaac Sim
- Grasshopper3D
- Bullet
- Nvidia Isaac Simulator
- Mujoco
- PyBullet
- Stage
- PyBullet
- OpenRAVE, IsaacSim
- Mujoco
- Pybullet
- Pybullet
- Blender

- RoKi
- Isaac Sim, Isaac Gym, Raisim
- IGRIP
- Siemens PSR
- AGX Dynamics
- AGX Dynamics
- Isaac Sim
- Mujoco
- PyBullet
- Open Scene Graph
- WorkSpace
- MuJoCo
- ARGoS
- Pybullet
- Isaac Sim, Isaac Gym and PyBullet
- Chrono::Engine
- MuJoCo
- MuJoCo, Drake, IsaacGym
- Drake, Pybullet, Mujoco
- Mujoco
- Foxglove, NVIDIA Isaacsim
- Drake
- Blender, FreeCAD
- Mujoco
- MuJoCo
- OpenRave
- ABB Robotstudio
- MuJoCo
- URSim
- Robot studio abb
- NVidia Isaac Sim
- MuJoCo
- Isaac sim
- Isaac
- URSim
- Pybullet
- SOFA
- NVIDIA Isaac Sim
- Drake
- Drake
- Simscape Multibody
- MuJoCo, Modelica
- Pinocchio
- VeroSim
- Fanuc ROBOGUIDE
- iDynTree
- RaiSim
- Raisim, MSC Adams
- RaiSim, MSC adams
- PYBULLET
- Delmia
- bullet
- Seimens process simulate
- Mujoco
- Raisim
- Raisim

- ABB RobotStudio
- Manufacturer software (Robotguide, RobotStudio,...)
- Verosim
- OpenRave, SrLib (SNU Robotics Library)
- PyBullet
- Microsoft AirSim
- Bullet
- Pybullet, mujoco, isaacsim
- IsaacSim
- Robwork SDU
- Tecnomatix Process Simulate
- Drake, Mujoco
- Drake
- OpenRAVE, Simox
- Doosan + Kuka robotics
- Raisim, Isaac gym
- AGX Dynamics
- Abb robotstudio, fanuc roboguide
- Pybullet
- Nvidia issac sim
- O3DE
- Isaac sim
- Isaac sim
- Modelica
- Catia, Roboguide, Robot studio
- Bullet Physics SDK (C++)
- KUKA Sim Pro, RobotStudio

Why not?

	Percent	Respondents
Not applicable	100.0%	1
Too difficult	0.0%	0
Other	0.0%	0
Total	100.0%	1

Why not? - Other

Have you heard of (Unified Robot Description Format) URDF before?

	Percent	Respondents
Yes	96.2%	533
No	3.8%	21
Total	100.0%	554

If you have used URDFs before, for how long have you been using them?

	Percent	Respondents
Less than 1 year.	17.9%	95
Between 1 and 5 years.	60.9%	324
Between 6 and 10 years.	15.6%	83

More than 10 years.	4.7%	25
I have not used URDFs before.	0.9%	5
Total	100.0%	532

Why not?

	Percent	Respondents
Not applicable	20.0%	1
Too difficult	20.0%	1
Other	60.0%	3
Total	100.0%	5

When was the last time you used a URDF?

	Percent	Respondents
In the last month.	67.2%	340
In the last year.	26.7%	135
In the last 5 years.	5.3%	27
More than 5 years ago.	0.8%	4
Total	100.0%	506

How experienced do you feel you are with URDFs?

	Percent	Respondents
Beginner	22.4%	113
Intermediate	57.2%	289
Expert	20.4%	103
Total	100.0%	505

In which application domains have you used URDFs?

	Percent	Respondents
Agriculture	14.1%	71
Cleaning	8.1%	41
Defense	6.3%	32
Manufacturing	45.3%	229
Marine	5.9%	30
Medical	11.9%	60
Transportation	24.2%	122
Other	34.1%	172
Total	100.0%	505

In which application domains have you used URDFs? - Other

- Education
- Rover
- Human-robot interaction
- Academia
- Commercial aerial vehicles, and PhD research
- Food indusry
- Surveying and mapping
- heavy machinery vehicles
- Research
- Food & Agriculture
- Search&Rescue, Industrial applications
- Service
- Mining
- Robotics
- UAV and drones
- Healthcare, Game
- Research
- Academia
- Mining
- Edit to add multiple robots with some base blocks of some size in some pose, regarding to a generated .yaml file. Multi-Robot workcells
- R&D
- Academics
- hobby
- inspection
- Humanoids research
- Legged Locomotion, exoskeleton and manipulation research
- Social robotics
- Human Robot Interaction
- Inspection
- Robot-Assisted Care
- Educational differential drive Mobile robot
- Aerospace
- Intralogistics
- Manipulation
- Assistive Robotics and Industrial applications
- Construction
- Research
- None
- Controls
- Research
- Aerial robotics
- Robot
- Exploration
- Exploration
- Foundational Robotics Research
- Research
- Academia, Aerospace
- Robotics project
- Surveillance
- Research
- Aerospace
- Logistics, research
- Academia

- Robotics
- Competitive Robotics
- Research and food processing
- R&D
- Robot competition
- Pick-and-place and locomotion research
- Education
- Research mobile robots
- Logistics
- Social Robots
- Inspection, Rescue robotics
- learning
- Acadia
- Retail
- construction, legged robots
- Service
- Robotics
- General research
- generalized development
- Household
- Oil & Gas
- Research on general autonomy
- Experinentation
- Robotics
- Academic/Research
- Robotics
- Academia
- humanoid robotics
- Construction
- logistics
- Robotics and Automation
- Robotics
- Manipulation Research
- Mining
- Mining Industry
- Logistics
- Space research, mining
- humanoids and research
- Mining
- Inspection
- Construction
- Aerospace
- Underground mine robots, space, educational robots
- Food service, automotive, education
- General Engineering
- Manipulation research
- Robotic Manipulation
- Space
- Automated Driving
- Robotics
- Robotics
- Research
- research on robot grippers
- Autonomous Navigation
- Research
- construction

- AI
- RnD
- Aviation
- Warehouse, Mobile Robotics
- Legged robots
- Modular robots
- Robotics
- home robotics, research
- R&D
- Assistive Robotics
- Psychological studies
- Academia
- robotics
- Retail
- Research
- General robotic AI applications
- Academia
- Logistics
- Logistics
- Mining
- Logistics/Warehouse management
- Mobile manipulation research
- Aircraft
- Logistics
- Construction
- Acedemic research
- Utility
- Construction machinery
- Service Robots
- Competition
- Construction Equipment
- Consumer (Domestic)
- Construction
- Just displaying my personal robot in RVIZ.
- Academia
- Research
- School Club
- Metrology

Which manufacturers of robots or end effectors have you used URDFs of?

	Percent	Respondents
ABB	20.3%	102
Boston Dynamics	9.3%	47
Clearpath Robotics	21.7%	109
Fanuc	9.9%	50
Franka Emika	25.8%	130
Kinova Robotics	14.7%	74
Kuka	35.6%	179
Robotiq	22.7%	114
Universal Robots	52.5%	264

Willow Garage	14.1%	71
Yaskawa Motoman Robotics	5.6%	28
Other	40.2%	202
Total	100.0%	503

Which manufacturers of robots or end effectors have you used URDFs of? - Other

- Reach robotics
- Pal Robotics
- Custom
- None
- Hello Robot; Rethink Robotics
- ResGreen Group
- Custom
- Rethink, Barrett, Aldeberan
- Custom URDFs
- Denso, Pilz, Schunk
- DENSO
- Techman Robots
- Self-made
- no manufacturer
- Custom Robot
- none
- Cutom
- Blue Robitics
- Custom
- Bespoke
- Schneider Electric
- Custom made
- Mainly robotics vehicles, UAVs and tractor platforms
- Magazino
- Custom
- Omron, Custom
- custom
- just my own
- antbotics
- Wandercraft, Enchanted Tools
- ANYbotics, NASA
- Softbank
- Rethink Robotics
- iRobot, Dexter Industries
- Schunk, OnRobot
- Red one
- Mainly robotics vehicles, UAVs and tractor platforms
- anybotics
- many...
- PAL Tiago
- custom made robot
- Sensor vendors
- Custom robot arn
- Custom robots
- Custom robots as well
- None
- custom design
- Epson

- Custom built hardware
- None
- Custom robot
- ANYbotics
- Robotnik
- Custom
- Kawada Nextage
- HyQ, Unitree, MIT Cheetah, Baxter
- Elfin
- Fetch
- Toyota HSR
- Custom
- Specifically designed robots
- Schunk, and other in house robot designs
- xArm
- Self-made
- Custom Robots
- N/A
- Tractor
- Unitree
- Comau
- Schneider
- Self built robots
- Neura Robotics
- Team CAD
- Trossen Robotics
- Rethink Robotics
- Custom Cartesians
- Dynamixel
- Custom robots
- DIY robot arm
- custom
- Nothing
- GummiArm
- Comau NM 45-2.0
- anybotics, menzi muck
- Own design
- none
- Custom
- Our own
- Customize
- Boston Dynamics, PAL Robotics
- Agility Robotics
- Custom URDFs
- Barrett, Custom built robot
- MiR100, OnRobot
- Pal robotics, iRobot
- Heavy machineries
- Robotis
- Dobot
- ROBOTIS
- None
- Unitree, ANYBotics, ROBOTIS
- Own developed robot
- custom
- Self made by the company

- Kawasaki
- Rethink Robotics
- Sensodrive
- Schunk
- Custom made by us
- ANYbotics
- Mining equipment from various manufacturers
- Custom solution
- custom
- Custom
- AgroIntelli
- Custom
- Custom
- Own
- Hyundai
- Dorna, Mecademic
- Made my own
- Robotis
- toyota
- Openbotv v1
- ANYbotics, Unitree
- Custom made robots
- Robot designed in-house
- IIT (iCub robot)
- Anymal, aliengo, my own design of wheeled robot, our own design of quadruped robot
- Designed my own Exoskeleton
- Anymal, my own design wheeled robot, our own design quadruped robot
- Self built robot
- Reis Robotics
- ETH Zurich's Hector Quadrotor
- Custom
- Staubli
- Robotis
- TurtleBot2
- Rethink Robotics, AMR, Quanser, Techman, Robotis, Mecademic
- baxter
- thorvald
- Neobotix (custom URDF)
- MiR
- Omron Techman Robotics, Rethink Robotics
- Franka Emika
- Neobotix, Robotnik, Alberobotics
- RoboNation
- Schunk Lwa4d
- Neura Robotics
- panasonic, staubli
- Agility Robotics
- None, custom built only
- Neobotix
- custom, Rethink Robotics
- Custom made
- PAL-Robotics
- None
- Comau
- Liebherr
- Custom URDF models

- PAL Robotics, Robotniq
- Self-built
- Robotis
- None, i used mine
- NONE
- Fetch mobile manipulator, robotis rx500, nasa centaur 2, pandas
- Franka Emika
- N/A
- Robotnik and vehicle equiped with sensors

Which types of robots have you used URDFs of?

	Percent	Respondents
Robotic arm (single arm)	80.4%	405
Dual arm robot	22.0%	111
Delta robot	4.2%	21
Humanoid robot	16.5%	83
Mobile robot	63.5%	320
Quadrupedal robot	14.3%	72
End effectors	38.9%	196
Drone	14.7%	74
Other	5.8%	29
Total	100.0%	504

Which types of robots have you used URDFs of? - Other

- non humanoid mobile manipulators
- tractor
- point robot
- none of them, just general kinematic trees
- Ship
- Underwater ROV
- Also with simultaneous execution... Talking about moveit property, this is not given with simple action servers
- Cartesian, gimbal, and custom
- Underwater vehicle
- Surface vehicle, underwater vehicle
- aerial manipulator
- Hexapod
- Sensors
- Underwater AUV
- Arm on a 7th axis track
- Snake robot
- Autonomous Underwater Vehicles
- Human motion simulation
- Custom
- multi arms
- F1tenth
- 4-wheeled differential robots
- Exoskeleton
- Autonomous vehicle
- Custom modular robots

- 3-arm robot
- Gantry
- Gantry robot
- Excavator
- Excavators

10. How have you used URDFs?

	Percent	Respondents
Always in combination with ROS.	66.5%	334
Always in combination with ROS, but would prefer not to require ROS.	17.1%	86
Without ROS.	16.3%	82
Total	100.0%	502

Are you aware of any of the following or other tools for creating URDFs?

	Percent	Respondents
Fusion2URDF	17.3%	81
OnShape to URDF exporter	12.6%	59
Phobos (Blender extension)	11.7%	55
PTC Creo to URDF exporter	3.6%	17
Solidworks URDF exporter	56.5%	265
xacro	82.1%	385
Other	4.1%	19
Total	100.0%	469

Are you aware of any of the following or other tools for creating URDFs? - Other

- Note for Q10: I use URDFs both with and without ROS
- Webots URDF exporter
- Blender
- RoboTool (SDF generation)
- RoboSim
- FreeCad exporter
- Note for question 10. I have used URDFs both with and without ROS. No preference on the use - it is situationally dependent.
- Sdf urdf conversation tools
- URDF to USD Converter
- BRICK2URDF
- We are developing our own STEP to URDF (github.com/ReconCycle/urdf_creator)
- sdf
- Build them in MATLAB
- No
- Tecnomatix Process Simulate
- collada urdf
- None

Have you ever used the tool ‘xacro’ to generate URDFs?

	Percent	Respondents
Yes	90.1%	346

No	9.9%	38
Total	100.0%	384

Have you ever used URDF models (xml file + meshes) with geometrical meshes of the robot (i.e., not just boxes or cylinders, etc.)?

	Percent	Respondents
Yes	92.8%	463
No	7.2%	36
Total	100.0%	499

How did you obtain the URDF models? Please mention the specific tools.

	Percent	Respondents
From simulation tool	20.0%	92
From CAD tool	60.1%	276
From ROS package	58.8%	270
From website of the robot manufacturer	47.1%	216
Developed the URDF by hand	67.1%	308
Other	2.4%	11
Total	100.0%	459

How did you obtain the URDF models? Please mention the specific tools. - Other

- Github
- Dont know
- RoboSim p-models
- Generate bu Designer og obj models
- included with drake
- Blender
- From customer
- Blender
- Applied Solidworks to URDF tool then fixed URDF by hand
- Solidworks
- Solidworks URDF exporter
- Solidworks to URDF Exporter
- From GitHub
- ROS-Industrial
- PTC Creo
- Onshape to robot package

If you have earlier created/modified a URDF model, what level of difficulty would you rate the process?

	Percent	Respondents
Easy	6.5%	30
Medium	57.1%	262
Difficult	32.2%	148

Don't know	1.5%	7
I have not created/modified URDF models before.	2.6%	12
Total	100.0%	459

What would you say is the most painful part of creating/modifying a URDF model? (Optional)

- Preventing Self collision
- I've mostly worked with integrating different workcells and end effectors with Universal Robotics ROS/ROS 2 description, and following the chain of xacro includes in many files is tedious.
- Ramp up and coordinate frame assignments as compared to my DH models
- Changing standards and Ability to find documentation on advanced usages
- Placing coordinates and joints in the right locations with a nice graphical tool. Even fusion2urdf still mislabels part names, and has bugs with complex joints
- Getting the poses right and adjusting friction ,damping and effort values
- Generation of model in different program than validation of model
- Finding the proper xyz and rpy position for joints and finding Inertial values for the components when urdf is created by hand, or else most of the things are easy
- Closed chain kinematics and building custom plugins for controllers.
- Getting coordinate frames right in first go.
- Creating it to be parametric for future adjustments
- the long files, sometimes it may get confusing
- The physics properties and collision
- Adding meshes.
  Using variables
- Knowing all the rules and tags...
- The ability to navigate included files
- Usually not having separate CAD files from the manifacturer. Even if we split the CAD file into multiple parts, it's difficult to calculate each part's position and rotation.
- Setting up Transformation between joints, as roll pitch yaw is ambiguous.
- Creating the correct TF tree
- It is very difficult to visualize the process. For example, the ROboDK has a very easy tool to create a robot from the mesh files if you have the robot information such as joint limits. But URDF, you have to write the whole thing and run some ROS package or code to see it and test it, the iteration is much slower, I wish there was a tool that I could use to add the mesh files and the robot information and do some adjustments and that would give me the URDF.
- Missing a component, adding a CAD mesh and alignment of sub components with right dimensions
- lack of good tools
- Orientation mapping especially with long chain robot arms, synchronization between the collision and visual meshes.
- Real-time feedback is lacking when hand-crafting or editing URDF files
- - No editor plugin available
  - understanding how everything connects together because of lack of documentation
- Frames definition for joints links with relation to base and com frames
- Missing a visual editor that would output a sensible Xacro (i.e. with clever macros for repetitive elements). Or at least a fast Xacro/URDF preview tool (using RViz is too cumbersome for this).
- It wasn't painful, because I had it automatically generated.
- I find creating Urdf itself is a complex process
- -It does not support parallel manipulators.
  -The Syntax to define the robot
  -I used it in combination wit ROS which it is inconvenient
  - It is very difficult to import meshes with colors
- - Fixing the individual physical parameters such as Friction, damping, and Inertia.
  - Modelling of each component one by one and then restarting the simulation to verify the changes. It takes way too long to model the perfect URDF.
- Non tree structured robots, Parallel links/robots
- Closed loop robotic models and unable to use it for general purpose reinforcement learning platforms such MuJuCo, Issacc
- - Need to know the syntax
  - difficult to get the setup correct on the first attempt - syntax errors, frame conventions, transformation

- I think for most users the most painful part is the definition of the right transformations and sizes in each link frame. This is what a 3D interface with joints and links hierarchy can help with.
- Updating the joint origin offsets. And determining hardware interface and controller for gazebo
- There should be a quick visualization tool in parallel while editing for fast prototyping. XML editing is laborious and should be able to see the changes in the 3D congratulation, i.e., what was before and how has it changed now.
- Locating parameter and topics
- The most painful part would be when trying to fix a model. I would have liked to have a tool that allows me to visualize, manipulate, and edit the model (without requiring me to load it in a simulation platform like Gazebo), so that it's easier for me to figure out what needs to be changed, or a modified version of the model is exported that I can compare with the original one. Also, some form of validation would be helpful so that I don't have to wait until I load the model into a simulation to notice that there are mistakes.
- Lack of support for parallel links.
- Passing the information (in Form of a file name string "example.yaml") to load it's parameters and finally place the robot somewhere else.
  I would also say relative poses, but math in this files easy, so all good.
- Testing changes
- For custom robots, adjusting the mesh positions with respect to the joints;
  Having a consistently in structure for multiple robots;
  Texture implementation on meshes;
  Limitations in PKMs;
- Lack of documentation and general steep learning curve
- Using CAD software (models and conversion extensions)
- - Ensuring correct paths for meshes etc. to be loaded.

  - Generating correct inertias, masses

  - Generating larger urdfs using xacro
- Parameter calibration
- Keeping up with tag nesting, remembering the syntax, visualising the offsets/origin points quickly without trial and error

  (Side note: 'painful' is a leading word in this question, it makes it clear you are looking for a specific result. 'difficult' or 'challenging' would be more fair)
- Aligning to the axes of the meshes to create an accurate joint.
- You can not easily change the frames manually in a 3D tool. It's either you edit the xml file manually or you edit the cad file and reexport
- Joint efforts
- The syntax
- Defining Inertia

  URDF created for ROS is often not reusable with other simulator with the need for modifications and vise versa.

  The Phobos tool would have been my best choice but it's with old blender version and not maintained.
- Most painful was my toolset did not identify line numbers of xml errors
- Editor
- Sourcing the CAD files for the links
  Sourcing the DV parameters
  Getting the orientations correct
  Linking with end-effectors (and adapterplates etc)
  Mis-match in supplied models.
- Finding a generic structure for multiple Vehicles that be resued.
  For tractors, this related to the modular setup of multiple implements (attachments).
- Automation, static configuration management, and any kind of dynamic changes to the model when in use.
- The amount of overhead due to the xml syntax
- Place the references frames in the good coordinate.
- Lack of time and custom parts
- Dealing with the XML format.
  Dealing with large models and, therefore, large files.

- - Gazebo plugins (not directly related to URDF per se...)
  - closed kinematic chains not supported by URDF
  - typically, only parts of a urdf model are imported into a simulator, requiring lots of workarounds and model changes

  - missing and/or very bad documentation for both urdf and gazebo sdf
  - very unhelpful developer community (both urdf and gazebo classic), literally thousands of unanswered user questions
  - missing gui tool. (CAD->export is often claimed to work, but so far I had 0.0% success rate. Always some workarounds or model changes required.) s
- Moving elements.
  Dimensions.
  Colors (not as relevant).
- Modeling, testing and debugging things necessary for simulation: Inertia tensors, masses, centers of mass; Gazebo plugin configurations; PID controller tuning; friction parameters between wheels + ground
- Setting up the transformation frames. Some parsers/writers interpret them in different way. Such as is Z up or forward?
- Easy when only boxes and cylinders, annoying when including meshes.

  Most typical, we only use the kinematics described in the URDF in our controller framework eTaSL. Sometimes also cylinders or boxes to describe collision models for our controllers
- The visualization tool is not friendly (I always launch the gazebo to check if the urdf is correct and it spends a lot of time).
- When the fille is big, it's hard to keep tracks it is not well for Reading
- Lack of basic maths
  Syntaxis
- I have only created URDFs that include kinematic models, and in such case the gathering of the kinematic parameters of the robot (e.g. distance measures) is the hardest. But I imagine that for people that model the dynamics is even worse. Modifying URDFs is fairly easy, specially if Xacro files are used.
- Doesn’t work well with closed chain kinematics. Don’t know how to handle multiple tools
- Sensors in simulation normally have to be disconnected which is a usual problem for newbies
- Absence of a standard and debugging tools both standalone and with simulator plugins
- It’s so complex
- Messing ud the file with different mesh files and Colors
- Scaling meshes together to form a good 3D visualization. Orienting objects in 6D for effective 3D visualization.
- Research all the components and how it works with the robot, there are not some information outside about this topic
- Circularity work around using plugins
- Having accurate and plausible dynamic properties, mass, moment of inertia tensor etc.
- getting mesh geometry, textures, etc. to work with various renderers. The support for different colors for the same body, etc. is also poor in many visualization tools. A standardized form for additional data (user-specified tags and attributes) would also be helpful.
- xml
- Exporting from a tool directly to a modular format -exported urdf usually has many hard coded elements, I always prefer xacro and use of properties, macros etc to build final urdf
- Complexity due to use of xacro
- rotations
- NA
- Working with many component descriptions can get messy
- The whole process is time consuming for the first time, the second time becomes easier. Still CaD to URDF lacks a uniform workflow.
- Binary VS. Ascii files parse differently
- Making the robot model
- Ensuing mesh files have origin in "correct place"
- cycle time between changing urdf and seeing changes in RVIZ. A tool with instant feedback would be nice
- Inertia terms and links to external meshes
- Physics parameters are hard to tune and configure
- It seems so outdated. It is amazing there is no GUI, easy to use tool available, since when you search the internet many people complain about it.
  When you then start to integrate Gazebo components it becomes a painful process.
  We also need better URDF visualisation tools in general, but being able to edit and to quick development and testing is needed.

- Dealing with scaling and rotations basically by hand
- Getting the right values to plug in
- Officially, the limitation to a single collision geometry per link. In practice most non-ROS URDF parsers support multiple collision geometries.
- Inertia tensor can be troublesome when it is populated manually.
- All the tedium. The biggest time sinks in our organizations are mesh manipulation (scaling, frame alignment, simplification for both visuals and collisions/fitting primitives when increased performance is desired) and aligning / verifying coordinate frames.

  We heavily rely on xacro tooling to simplify the process with "lego" style macros that are reused to build up models from common elements.

  Unfortunately, all of the CAD->URDF export tools go directly to (rather messy) URDF, rather than xacro, and make modification post-export a pain. We instead bite the bullet and focus more on a one-time mapping between generated URDF and our xacro format, which then supports improved maintainability and use with our tooling.

  URDF also has a number of quirks and disconnects with other formats (like SDF) that makes the learning curve a bit of a pain for newer users.
- Would prefer an automated tool that generates it for me
- You have to manually specify all the macros that could possibly be included in a URDF, i.e. combinig multiple urdf macros is not easily doable
- Getting started
- To redo calculate of joint position, orientation ect. Than again & again visualise it in rviz to check everything is correct or not. There must be some features like in Coppeliasim to just drag make urdf.
- Mistakes in the URDF file can be tricky to catch and difficult to identify until run-time. Incorrect/inaccurate parameters may require extensive testing (e.g., robustness testing) to identify.
- Caculating values manually such as moment of links
- Poor introspection into URDF development. If I add a link, modify a joint, etc, it's not easy to see what my change has done.
- Verbosity and lack of debugging tool in case of malformed URDF file.
- Lack of debug tools, linters, lightweight visualizers, and meaningful error messages.

  When importing into simulators like gazebo, the model with either work or not work, error messages are not provided or at least do not provide a way to debug.

  Also cad tools for exporting to urdf cannot generate a low resolution model based on primitives for optimal simulation performance.

  Getting the inertia parameters correct is difficult.

  Tuning drive system controllers to match real life performance feels impossible.
- Positionning the frames when the 3D are not well made. Scaling the 3D if the dimensions are not good (for example, difference between .dae and .stl)
- Exporting models from 3D software (Solidworks, Blender)
- With a text editor
- Does not have a good linter in IDEs. Makes formatting hard. Also defining sensors, cameras.
- While creating, aligning the parts and setting the orientation.
- All of it
- Obtaining URDFs from CAD models is pretty difficult. I feel like if that CAD tool has information about the joints it's should be as easy as exporting a different file format. We are currently utilizing solidworks to obtain a URDF of our robot which has omni wheels and we are having to do some pretty repetitive work in terms of declaring each of the rollers on the wheel as a moving object.
- Debugging tools, inertia
- Integration with Gazebo (root may not have inertia, all other links may not have inertia below 1e-5 or so), concept of transmission, missing conventions when to use plugins as robot_hw vs direct sensor plugin attached to link/joint
- Imported files from other packages. It is not easy to navigate between files in different packages
- The structure of the file is hard to read / parse.

- No support for closed loop kinematics
- Parametrization
- Proper nesting for reusability and composability
- Controlling the urdf with ROS control is confusing and forces your controller implementation to be compatible. Or I'm an idiot and can't do it
- So many,
  not supporting ball joints
  Not exporting right overidden masses
  Noy exporting correctly joints parameters
  Reference system difficult place
- The seperation between the XML description and visualisation.
- Not user friendly
- This isn't a problem that has affected me, as I have only used URDFs to make serial-link robotic systems. However, making complex interconnected mechanisms or compliant components seems like a challenge that would make the URDF creation process significantly more challenging.
- 1. Parallel joint capable
- Getting the mesh transformations right.
- Adjusting the origins and dimensions
- In general, URDF models are easy to generate. Maybe a painful aspect when working with large models is getting the physical properties right and it is easy to introduce small errors that are hard to find. But, I'm not sure this could be avoided in any way.
- Formatting / figuring out where the formatting errors are
- Setting inertia.
  Mesh scales and orientations
  Visualising changes.
- - Lack of real-time visualisation (VSCode has an okay plugin) partnered with it being a very manual process of creating links and joints which have mountains of optional fields.
- The tools are not accurate and don't create a usable model. CAD models sometimes have to hand annotated. Can't represent parallel mechanisms like delta robots.
- Using Fusion2URDF there where many unnecessarily complicated and sketchy steps I had to take. The process wasn't intuitive at all and even has to fix some bugs on the tool, to have more depth in the component tree.
- Generating proper meshes, dynamic changes
- - That it should not be necessary by this point . . .
  - Creating a parametric xacro based on options
  - Loops in the kinematics
- Getting coordinate transforms right. A simple standalone tree viewer (like a matplotlib 3d script) that allows for live updates would be helpful. Maybe as part of a python package?
- Plugin to convert 3d model to URDF sometimes the set parent child relationship doesn't work properly. It takes time to debug it
- Format is not beautiful.
  Only too simple robot structures are supposed.
- Visualizing the changes or knowing if something was done incorrectly
- The code same think foe every joint
- Setting the joint limits and zero positions correctly.
- There is no distinction between an actuator and a joint. So some mechanisms are hard to model - e.g. a 4 bar linkage like on some Fanuc robots.
- Maintaining the visual components to be in line with the actual model. Also xyz and rpy components are harder to tune without being able to see it change in real time. So every time I changed a small value, I'll have to repeat the process of bringing up the whole simulation to just see what that change caused to the URDF overall.
- Not able to edit easily. not having modular architecture
- not be able of using universal joints
- troubleshome to validate in real life
- It isn't able to represent everything we need to simulate. It is also limited to systems without kinematic loops and has many other limitations.
- Limited parameters
- Some models have virtual links, that have no intertia defined. these should be massless, but hard to solve dynamics with zero masses.
  I have a way around this with AGX Dynamics now, at least.

- Lack of support for customization like rotor inertia and coupled closed chain actuated joints
- The most painful part of creating a URDF is getting an accurate representation of the reality in simulation. I mean adjusting every little parameter is a little "nightmare" without tools that refreshes the representation of the robot to see what are you doing in real-time.

  Also, it is hard to work (at the beginning at least) and understand the difference between the orientation of the *.STL (visual and collision) and the orientation of the reference frame. Because the ideal situation would be that they are coincident, but this is not the case is you cannot have access to the original cad files.
- Finding the correct physical properties associated with links of the robot
- Sometimes hard to find specific components as it is a continous stream of code
- - Lack of validation at design time
  - Typo mistakes while writing the URDF manually
  - Lack of conventions to define the root and end-effector links
  - Issues with the physical properties and how gazebo handle them
- No unified easy toaccess documentation.
  Even for small robots (especially) parallel manipulators, the wirkflow gets extremely convoluted
- Each time entry of links and joint.
- Parallel kinematics
- No real-time visualization, weird errors from the parser, and unable to change during simulation.
- To find the geometrical and inertial parameters of the robot.
- Writing XML syntax
- Editing properties (like mass, center of mass) and everything in xml.
  Making joints, again, in XML, keeping track of names and everything. Not entirely confident if I am getting my desired behaviour from setting the text fields in xml.
  Finnicky exports from solidworks and in general a lack of good documentation and an XML based description is tedious to write and read through. Also getting the physical properties of the robots in ROS
- not strictly related to URDF but maintaining the tf (transformation) links in ROS
- Deprecated term for differents version of ROS
- The syntax is very abstruse and there are several properties which need to be repeated (for e.g. when creating a box object you need to give the exact same details in the mesh as well as collision.)
- Implementing correctly closed-loop kinematics.
- The most complex part is when trying to model a robot with a joint that moves and connects to other links. Also, adding controllers and broadcasters is a complex step to provide a simulation tool.
- getting the robot geometry right by connecting the different joints
- Collision boxes and inertia
- Trouble visualizing when trying to debug.
- Ensuring transformations are correct and match specifications of a target real robot.
- Integrating CAD files and ensuring the scale is correct for the whole workcell
- Getting exact dimensions + transform tree correct numerically.
- Obtaining the coordinates of each body / joint from the CAD model
- time consuming task in particular when the CAD is an early stage model for a prototype
- Documentation is difficult to find. Not all simulators/tools support the same tags
- Adjust the position and rotation of the joints
- Modifing even simple things like rotating the axis of a frame usualy cause nested issues like offsets of the frame and we rather import the whole urdf from scratch
- parallel kinematics
- Inertia data, export the right reference frame for every link
- Geometry mesh and dynamic parameters
- The process is thar you are constantly visualizing by launching rviz for every small change you made and this is tedious. A graphical interface to define a URDF would solve the problem
- Quite verbose but macros can handle that. Debugging the URDF was the most painful part.
- Usually need to clearly specify all the coordinates, geometric relations, and rotational axis in the CAD model (I use Solidworks) before exporting it to urdf. Most of the time the automatic generation fail.
- Lack of many functionalities... would like to discuss these further if there is an initiative to improve the current version of the URDF
- The complexity

- There is no good editor for URDFs.
- Following the xml sections. Knowing the standard/options for each section within the URDF.
- Getting the simulation engine to correctly interpret the model when there are closed kinematic loops or intersecting linear and rotational joints (eg a hydraulic cylinder attached to a universal rotational joints) and making the collision geometry and visual geometry to match.
- While the creation process is a bit time-consuming, after all its ok. The most painful part for me is the validation process.
- Modifying end effectors
- Getting direct visual feedback of changes (especially xacro based models)
- * finding the link->joint->link tags
  * missing clear definition of the forward linkage of rotations (try-and-error of direction of rotations or axes)
- CAD to URDF tools have poor support and resources. There is no easy way to support joints other than revolute or prismatic.
- Coordinate Transformations
- Writing XML is tedious; awareness of the (URDF) scheme is required, and the language is verbose.
- Visualizing the developement
- When generating URDF from a CAD model, if the model is not designed with the export process to URDF in mind, can require a lot of time to set up requisite reference frames and axes to define link and joint locations.

  When editing a URDF file, no tool (that I am aware of) exists to allow adjustment of values in the URDF with an accompanying real-time adjustment of a visualisation of the model to reflect these changes.
- Unsupported joint types like screw joint or cylinderical joint
- Unsupported joint types in gazebo or other platforms
- Getting offset orientations between joints and links correct
- Getting the rotation correct. Never knowing which way you have to rotate... Positive or negative, especially when you have already rotate about another axis.
- Recently i was using if conditional statement in xacro and was creating a joint xacro macro tag that can receive joint type as parameter but was unable to work with the if statement part, so i think logical statement are difficult to implement in urdf
- There is nothing that isn't painful. XML is a bad format, parsing seems to be inconsistent, it's not clear what is and isn't possible, there's no clarity on conventions, files do not seem to be compatible across programs, exporting from tools tends to give bad geometries to the urdf, etc.
- The CAD Exporter is generally not detailed enough and require the mechanical engineer team to follow some guidelines so the export is successful.
  I always end up, working in SolidWorks first to correctly export the model.
- Tracing all links and check the syntax
- parameterization, namespaces, connection to other frames in the world which are not in the urdf, allowing online modification...
- Getting the poses for various transform fields, like origins used in links and joints, correct.
- Reviewing and testing the model. Once you have created a model it's hard to verify that it is correct.

  Making models by hand can be tricky, but using the CAD plugins makes messy URDF files that are hatd to understand
- Visualising it. It would be nice to see what you are making
- 1. Any change in the urdf affects everything. And every time you do that you need to load the robot description again and check with some visualization tool if the frames are correct.
  Being able to change the robot description online with some command line or python scripts would be a nice feature to develop new urdf files, then later export them. Also to allow online sensor calibration.

  2. It is not the URDF model problem, but sometimes you want to test some or third party code and there are some hard coded transformations, that changing the urdf make it easier than trying to change the code, however there is no standard, like ZED came, how you attach the camera frame to the end effector and then connect the odometry to the base of the robot. There is no easy way to do that I think, at least the process is different then with a realsense, etc. A more clear standard for connecting multiple urdf would be nice.

  3. Plug-ins such as gazebo could also be a separate thing as a standard and not in the middle of the urdf, I think it is confusing. Like always call another xacro to add the plugins
- loading and reloading the model in rviz or gazebo, fine tuning is a mess. xml format can be annoying
- Preparing the meshes and set the origins correctly, as well as obtaining the correct transformations
- geometric calibration addition is quite painfull
  Use of Denavit-hartenberg was also complicated (did not tryied for a long time)

- There is not a friendly editor to change the properties
- It is tough to visualize the transformations and coordinate frames. It's also hard to add plugins.
- People don't follow any standard in writing URDF files. They are all over the place from different manufacturers, datasets, etc.

  It is very hard to have different type of schemas. USD seems better alternatives.
- XML is a quite verbose description. Some packages i.e. ABB try to implement configuration management (different versions of an arm in the same file) not supported by some URDF tools. Hard to use URDF in own applications (without ROS) as no library available.
- XML format is very pedantic
- mass frictions etc
- Format is not well-documented.
  Error-checking is non-trivial and time-consuming.
- Probably, that sometimes there are incompabilities with the version or distro, and that causes unexplainable errors when downloading an URDF file and trying to edit it.
- debugging xacro output
  maintaining parts in urdf that will compose a robot
  having a ready packages for sensors
- Repeating and nested structures, ex: collision and visual, are under link tag, but as you scroll through the screen, u often get confused about which tag you were in, an example is collision tag may appear in multiple places.
- - Managing separate mesh files for visual and collision models
- Keeping track of origin and orientation for each link.
- I want to have GUI to look over my system
- The back-and-forth between writing the file and visualizing the outcome. A GUI would be welcome.
- I have had the need to modify URDF models when working on a research project involving ROS with rViz and Gazebo. The most painful part was ROS itself and having worked with Robotics for almost 10 years I will never understand the push of ROS in the manufacturing industry where reliability is key. I think URDF is a very well structured and comprehensive tool to digitally define a manipulator. I am not creating or modifying URDFs on a daily basis anymore because most of the robots I work with have URDFs made available by robotics manufacturers. However, the move towards ROS is painful and costs companies a lot of money. Especially when next to 0 serious manufacturing companies use Linux systems for their simulation software because all the serious simulation software or design software runs on Windows, e.g. Panasonic DTPS or SolidWorks. Just because something is free that does not mean it does not imply some indirect costs. I support the implementation of URDF in mature software, like SolidWorks. This survey made me aware that such system is available for SolidWorks and I will check it out. Thanks!
- The absence of tools to editing the URDF in a visual way.
- The frames are not much clear, cause confusions sometimes. Creating multi body systems with large number of links are take too much time and easy to make mistakes.
- visualizing and getting the relative transformations correct. Sometimes the syntax is not very easy to use and requires to make the urdf description as a ros package which is not ideal.
- I would be happy if we could create a closed-loop kinematic system. (Like human-exo coupled model)
- Not able to create closed loop kinematic system.
- 1) To create SDF compatible URDFs.
  2) Use of Plugin tags for e.g. Gazebo.
  3) Distribution of content over several files when using Xacro and the length of generated URDFs
- For a single arm robot, we typically use dh or mdh model to define its simulations. However urdf follows different conventions and makes conversation complicated
- To define each and every physical link and joint and the appropriate geometry to ensure the physics of the simulation world to be as accurate as possible. Especially during the procedure of creating a custom URDF.
- Lack of tooling, especially for live editing
- XML & doing everything by hand
- Correct association of joints, links and their poses.
- Lack of immediate visualization.
- Creating parallel links and funding a way to make them act in unison
- *Tagging/labeling.
  *Lack of visualization when composing via a text editor.
  *Endless scrolling when composing a model for a robot with so many dofs (humanoids/quadrupeds)
  * Necessity of creating pseudo links (0 mass/inertia) to accomodate multiple dofs at a certain frame

- The lack of code automation, specially for robots with multiple links
- Not exactly specific to URDF, but lots of robots come in multiple versions (due to error or actual physical modification), and making sure you have the right robot / URDF isn't always trivial.

  I've built some SDFs off of URDFs, which can be kind of annoying. Most ROS files use the XACRO & a launch file, not the URDF, which means if things don't work out you inevitably do some hack debugging.

  URDFs are pretty easy to work with for robots that are modeled serially, but in my experience the biggest pain is when someone adds more advanced features like mimic joints that aren't supported everywhere. I haven't had to create such URDFs (just import and modify), but I imagine it would be frustrating.
- When I needed to use URDF files several years ago, the wiki (http://wiki.ros.org/urdf/XML) had a lot of ambiguity about how things were defined. For instance, Planar joints never made sense. Are dynamics and limits of these joints applicable to both translational and rotation freedoms? Just looking at now for the first time in a long time I do admit there have been improvements to the descriptions though. In general, don't like the child-parent structure and inability to model closed loops. rpy as the only way to specify orientation can be annoying. The whole specification is very restrictive.
- Unintuitive
  obtaining appropiate parameters
  time between changing something in urdf and visualisation (especially bad in gazebo)
  lack of gui based tools
  creating stls for simple shapes
  inconsistency between how urdfs are processed between tools (especially colours)
  applying textures
- Creating STL mesh files with appropriate coordinate systems.
- Using the right xml properties to set up frames and joints with the right simulation properties (friction coefficients, etc.).
- reloading the model every time you make a small change in the URDF/xacro
- slow iteration when making a change since you need to reinitialize the simulator
- Determining the order and effect of transformation. When trying to calibrate the model in simulation with real-life robot, minor deviations in real-life are tough to integrate into the simulation as it is hard to track rotation orders and effects.
- Getting plugins to work
- For large xacro based urdf trees, finding the correct urdf file to make the change in. Since they are all linked by ros pkg reference, you need to go from pkg to pkg to check what’s in there.
- - getting the frames right
  - understanding transmissions/controllers/hardware_interface
  - unpredictable behavior when simulating 2 DOFs joints
  - understanding the effect of attributes such as friction, effort and damping
- Learning where to start on your own.
  A little bit of confusion in the beginning
- Have to edit the URDF/XACRO files by hand, save, launch a couple of nodes with Rviz to verify, edit again if there are any problems. It's just not WYSIWYG.
- Hardware Interfaces and the transformation between the links, so that the robot looks as it should and was planned in CAD. Constructors don’t use convention of Denavit Hartenberg, which gets painful when trying to automate the generation of URDF-Files.
- xacro is really awful because of xml overly verbose syntax and the large amount of includes and arguments make it impossible to understand what the final output will be
- Code duplication and rotation handling. To be honest I think xacro solves 99% of my problems. I love it
- Generating a reasonable moment of inertia
- Extremely time consuming task
- Unable to visualise, debug
- I've been editing URDFs in a text editor and then using RViz/OpenRAVE/Urdfpy to visualize it. This takes a little bit of time, as for every change one needs to save the file and launch the visualization tool. An editor tool where one could edit and visualize the urdf at the same time would be great.

  Also, if I recall correctly the joint mimic functionality is a bit limited.
- Running RVIZ/the simulation again every time the model is changed. A live viewer would be nice
- Easy to mess up the XML format. I sometimes need to hunt for a syntax error. Using a better XML viewer/editor might help.
- No support for closed kinematic chains.
  Many tools do not support custom tags.

- Meshes
  Closed loop system
  Adding the control layer (ros2_control)
- unsufficient error message, it takes long find the missing part in a urdf, compatibility issues with gazebo
- Correctly integrate it to the simulator. Learn how to use it and how to structure the code is not very intuitive either.
- If a link needs to be shifted, I usually launch the simulation repeatedly to check if it was shifted in the right direction
- Tuning the model for the robot self filter to fit 3D sensor data
- It is a powerful tool, however sometimes everyone of us struggle with adding the transformation. Apart from it, myself I find it hard as it has a limitation to only open linkages. If you want to define the closed kinematic chains it is currently not possible
- Exporting all the assemblies, the exportation does not work well all the time.
- Manually editing the xml to take into account joint limits, offsets, physics properties. Linking geometry files can also be a little frustrating sometimes.
- Understanding transformations between parents and children. Assigning mass and inertials.
- Aligning mesh origins and properly understanding the relationship between mesh origins (in the mesh), the various origins in the urdf file took a bit of trial and error.
- The whole process is cumbersome and error prone.
- How to deal with kinematic loops, or non standard joints (eg garage door)
- Having to manually specify ever link, joint & transformation as well as the joints capable of motion. We had problems getting our CAD model (SOLIDWORKS) to work correctly so we manually exported each part and did the assembly using xacro.
- The conversion from URDF to SDF is not always right.
- Creating the convex hulls of the meshes (My recommendation: https://github.com/kmammou/v-hacd)
- combining multiple robots and other objects

Have you faced any challenges or limitations with URDFs?

	Percent	Respondents
Incorrect forward kinematics.	26.4%	118
Difficulty importing the URDF into simulation tool.	44.1%	197
Difficulty creating a URDF file.	48.8%	218
Difficulty adding end effector model to robotic arm model.	32.4%	145
Could not find a URDF with the specific version of the robot I want to simulate.	38.0%	170
Limitation in URDF did not allow creating model of robot with parallel linkages.	39.8%	178
Other	10.1%	45
Total	100.0%	447

Have you faced any challenges or limitations with URDFs? - Other

- Incorrect dynamics
- link visual geometries can only be solid bodies. No (working/accepted)*.xsd
- Missing dynamic model support of vehicle type
- No tool to create more robots it of the Box. Like what am I supposed to do with only 2 robots hell. Also they are no beginner stuff.
- Difficulty generating correct inertias
- No support for elastic / deformable links
- Gazebo only used color of first shape in a link
- Missing dynamic model support of vehicle type
- Challenges and verbosity when mixing with SDF.
- Limitations modeling non-rigid structures (like flexible tubes)
- No good way to model singularity free spherical joint.
- Robot manufacturers such as Kinova give lots of options for the URDF generation through Xacro, but do not document it well. So one has to read the URDF Xacro specification and spend some time understanding it before using the desired configuration.
- not able to link the the hardware with URDF in rviz

- Incorrect inertial parameters
- Discrepancy between link origin and geometric mesh origin
- N.A.
- Lack of intellisense
- Not strictly parallel, but more complex linkages for some serial robots.
- In my case it is not incorrect forward kinematics, but if you wanna try new configurations or some that are not standarized, you might end-up using tricks to compensate the differences as auxiliar robot_publisher_states.
- Writing XML
- Difficulty to hand and to model behaviors of real world
- Unsupported stuff in the simulator
- missing frames support
- Tool limitations to verify correct URDF parameters for current application
- Material definition mess when using several instances of the object
- parallel kinematics, few dynamics parameters
- For cannot create model of robot with parallel linkages, I turn urdf to sdf, though SDF is also nested and hard to use.
- Documentation
- Have to install ROS just to get the URDF out of the XACRO, which is the actual shared item
- Adding acceleration (or even higher order) limits to joints
- Some industrial robots have coupled joint limits, which is not support by URDF.
- inflexibility in defining dependent joints as functions of driving joints
- N/A
- Code duplication and maintainability
- Mimic joints not well supported; conveyors hard to model
- Only linear joint mimic relationships
- Acceleration and jerk limits cannot be defined. This would be useful for MoveIt.
- Limitations of inverse kinematics computation
- Mimic joints with linear relationship is anothing restricing issue
- Some errors
- Errors in the URDF file
- The form is giving me errors saying something is incorrect but not what. 60 more seconds and I'll abandon this..
- It is an incomplete format for simulation
- Just tedious and time consuming.
- Adding the ros2_control tag dynamically to allow specific interface setups

Are there any improvements you would like for URDF?

	Percent	Respondents
A versioned standard (i.e., URDF v1.0, URDF v1.1, etc.)	35.1%	164
Better tools for combining and manipulating URDF files	70.4%	329
Better tools for validating URDF files	64.7%	302
Better documentation	55.5%	259
Easier methods or tools to create URDF files	68.7%	321
Standardized metadata like originator, date, version, etc.	27.2%	127
Other	11.3%	53
Total	100.0%	467

Are there any improvements you would like for URDF? - Other

- More official manufacturer support with accurate kinematics/dynamics
- inclusion of deformable bodies
- Standard
- Better parallel structures support

- Deformable links / continuum rods
- It is unclear what 'validation' means in this context, but if it means pre-checking syntax and correctness, then yes
- ROS2 Link name suggestion/linter
- We are missing a way to specify "joints" related to the task at hand, not to the kinematics of the robot itself. Would like to have joints that can have functional dependencies to e.g.to model coupling of joints due to transmission, trajectories etc. Tools for on-line combining and manipulation of the URDF-model, e.g. when picking up objects and performing tasks with them. Better tooling to describe the complete scenegraph, not only the robot. We typically read and process URDF ourselfs, so it is essential to be based on standards such as XML (or JSON if needed). A standard way to describe extensions to URDF, such as a new type of joint, Further in the future, a standardized way to describe constraints on the scene-graph tree (e.g. parallel robots, closed-loop linkages,...)
- Be able to deal with parallel linkages
- the need to use xacro to dynamically generate URDF is difficult. A more portable library/standard for dynamic generation would be great
- Improve SDFormat so that it can be used everywhere URDF is used.
- Make it like SDF
- Ability to handle closed chain methods, easier to integrate multiple urdfs for different robots in common scene
- Better integration with Gazebo plugins
- Support multiple collision geometries per link
- standarisation of refenrence to wich mesh position is defined e.g realtive to previous link or robot root
- Somewhat inline with "better documentation", but I'm sure the community has a ton of best practices and helper functions/macros that would be well worth adding to the existing documentation. Otherwise, I've always dreamt of a tool that, similar to a 3D printer slicer software, would allow you to manipulate and place meshes and primitives, create joints, etc, to simultaneously build and visualize URDF files. Would help quite a bit when placing frames, ensuring the correct axes are selected, aligning meshes, visualizing inertials like you can in gazebo and validating that they make sense, visualizing collision vs visual geometry to make sure they're well registered, etc. Inline with versioned standards, it could be great to get a superset of current URDF that is cross compatible with other formats (like SDF) without breaking compatibility with existing tools (e.g. RViz, TF2 which don't support closed kinematic chains, for instance). There is quite a bit of literature on features that could/should be added for different fields of robotics (e.g. automotive would have very different needs than soft robotics). Versioning could be a good way to deal with that
- Convert everything to sdf and add standardized plugin system for different simulators
- C
- Dynamics parameters (elasticity)
- Closed loop kinematics support
- Better Gazebo integration
- Closed kinematic or loops
- Lots of boilerplate code for e.g. gazebo, simulated camera is overwhelming but shouldn't be
- N.A.
- Representing parallel structures
- Higher level GUIs for creation would be helpful when creating customer grippers or robots. Based on the STL model, and interacting something like the moveit setup assistant. Also, expanding to allow STEP files would be beneficial for custom kinematic chains, this would mean that the files used can be directly imported into CAD software to be fixed or adjusted if need be.
- I don't expect much on URDF.
- Deprecation of URDF in favor of SDF, there are too many fornats
- Closer loop kinematics
- An URDF interpreter with suggestions, and a better design for custom controllers
- Additional functionalities
- URDF linter for VSC
- support for closed loop robots (e.g. delta)
- I would like a URDF replacement that isn't bad
- Editing urdf online to allow sensor calibration in real robots
- more dynamic parameters
- Easier methods or tools to modify URDF files
- The fact that it is impossible to differentiate between link and frames (see https://discourse.ros.org/t/urdf-ng-link-and-frame-concepts/56)
- Implementation of closed-loop kinematic system

- No closed loop kinematic
- A general frame-based specification that allows parallel linkages
- Better conversion support to other formats, like USD or MJCF
- Extend the scope to no just robots, but obstacles, items, etc.
- Support of parallel links
- Visualization tools like rviz that is more lightweight
- Support for tool change at runtime (including tools that have additional joints, like grippers)
- Parallel linkages support
- N/a stupid form won't let me continue... Ahh didn't understand that the last question below had buttons.. just looked like labels
- Better extensions for formatting urdf in vscode - can not get it to be human readable with auto formatters
- Extend format with simulation properties (instead of<Gazebo> tags
- Separation of robot structures and visualizations
- Including end effectors

Do you think that URDF will be more commonly used in the future? Please justify if possible.

	Percent	Respondents
Yes	53.4%	256
No	12.3%	59
Don't know	34.2%	164
Total	100.0%	479

Do you think that URDF will be more commonly used in the future? Please justify if possible. - Yes

- As part of the ros standard make sense to continue the use and improve URDF descriptions to be a middle representation in any 3D software
- Perform visualization for testing dangerous controls
- Seems to be a de-facto standard. SDF would be a good option but less used
- It would be preferable to use a single urdf with all simulation packages
- It seems commonly used, it's not too difficult to manipulate, And I found no major problems with it
- Yes, URDFs will become more useful as more out of the box tools are developed on top of the platform (e.g. planners)
- Probably... but only if there are more tools and development put into making the process much easier and more intuitive.
- even if it is a bit annoying sometimes, it still gets the work done and with xacro extension you get most of the necessary tools you need to describe a robot
- Xacro Files are the future, for better recombination. Digital twin wise, URDF + ROS2 are one of the best and most standardised solutions available.
- With the rise of ROS robots, URDF be more commonly used for sure.
- It is integrated in ROS and many tools are already using it
- I think it is easy enough to be created by hand but offers scalability
- because it is used with stateoftheart robotics simulators
- For model based risk validation
- Accessible for non experts
- It's a defacto standard
- Yes, it has become a de-facto standard in ROS.
- As a target langauge.
- as it allows the developer to simulate the robotic system without buying the hardware. Any number of sensors and actuators can be mounted
- no
- Yes if it is recognized as a standard outside the ROS world
- I think that since they are used in ROS, they will be continuously used, probably more
- I don't know what the trend is, but even if there will be a new standard, I would hope that it follows in the footsteps of URDF. I've been using URDF for years, and by now, it represents a natural way for me to model robots.
- Is the best current option I'm aware of currently for visualisation of mechanisms when combined with moveit/ROS/Rviz
- Don't know something else tbh. Especially they are used in every moveit tutorial by doing something with the panda robot, so it's the entry point in that open source robotics field.

- It provides a standardization for robot model that can enable an interoperability. There should be always a format like that.
- Because most recent tools are built around urdf
- The standard greatly simplifies and accelerates robotics research
- It's still better than anything else
- 'more commonly' is a stretch, but I think its use will continue and it will be standard to release accurate URDF models with future robot releases, so it will continue to be widespread in the field, but I don't think it will explode in popularity
- Well integrated with ros
- It is more intuitive and one can understand how the robot work through the process
- there are not better alternatives
- Good to standardize the development of custom robot
- It is already one of the most common tools, should start to dominate. It is basic for visualization in ROS environments (Rviz, Gazebo).
- They're pretty handy to use and quick to write/generate.
- It is a great tool, but need some usability improvements
- Integration trends
- Since the number of people working with Open Source robotics is growing, and URDFs are quite popular. Also several robot manufacturers are starting to provide them and maintain them.
- yes... because more people are now interested in constructing their own urdf
- It is a widely accepted format. The only other formats are SDF and USD
- Because use this kind of tools you do not have risks
- It is used with ROS which is becoming the standard for robotics applications
- the status quo standard
- Still de facto
- Yes, it makes kinematics easier.
- But other tools might take over
- Possible with xacro features
- default in ROS
- It is generally a very common/reliable format, and there is no real need to switch.
- I don't see another standard and everyone has put in the effort to write parsers...
- It seems to most used standard so I would presume that it will be continue in the future
- Hard to say. but I would lean towards yes. Its certainly difficult to develop in ROS without using URDF/Xacro. Moveit, Navigation, Tf2, RViz, Foxglove, and a ton of other large-user-base, high-inertia projects/tools rely on URDF. Outside of the ROS community, there are many tools that support URDF but also offer their own formats (MATLAB/Simulink, NVidia Isaac, Webots, Unity, etc...). I could forsee Gazebo/Ignition being overtaken by a different simulation platform, but the software backbone that robotics developers find so useful is tightly reliant on URDF
- A
- ROS is growing fast
- I think it's a great idea conceptually but it should be made accessible and easy-to-use.
- Grow with the success of ROS
- Better integration with cad will boost their usage
- Dd
- Probably, due to the success of ROS2
- Hope so
- Why shouldn't be?
- Cuz it's already doing good job
- More of a push towards open source options in both industry and academia leads me to think ROS will be further adopted and it seems to be the standard for that.
- The need is there and there aren't any fundamental flaws.
- It's the default format for robots in ROS so yes
- Yes
- Many robotic companies are understanding the importance of simulating before real. Hence I think URDF will play a huge role in the coming years.
- Unless a better alternative comes up
- The idea of the URDF is so good and aligned with ROS. Even though it might be a pain working sometimes with the URDF and the scene, once it is set up, you can forget about it and work normally. So in my opinion, the initial pain is not a big deal.
- The flexibility

- .
- Some type of standard for specifying these things is necessary. The current implementation is very bare bones and lack tooling, so I hope the future is bright for the format.
- it will spread alongside with ROS/ROS2
- It is a good starting point, it could be best
- Either URDF or a similar standard e.g. USD is extremely needed as a cyber-physical interchange format
- if easier to export and work on linked kinematic chain
- It is a very powerful tool describing robots
- I believe yes because there isn't any other option.
- Because of ROS and ROS2
- Could be adopt as standard for the kinematic description
- It is an open standard for editing and creating robots for ROS simulations but it needs a lot of improvements that make it easy to use
- It is easy to understand and it is used in many simulatord
- Facilitate integration of robot mechanical design on overall development cycle. Innovation of robotics will heacily depend on novel mechanical designs.
- supported by a wide range of simulator, and algorithmic libraries, less limitations then those classic modeling (dh table), support mesh/visual metadata
- There is no alternativ
- It's well known and wildly supported
- I'm sure there are better standards, but it's so widely adopted already I can't see anything else gaining traction
- The main approach for robot description in ROS
- the ability to use it in many different software tools is a great benefit, but the workflow of urdf creation should be less cumbersome
- It makes it possible to work acroos platforms.
- It feels like a standardized way of representing 3D objects with internal physics and rigid body. It is highly beneficial to have the same files and notation for metadata.
- We continue to attempt to do more complex operations with manufacturing robots which require collision avoidance to operate in flexible, changing domains. Urdf is an important format for describing robots in every collision avoidance software I have encountered to date so I expect this use to grow
- easy to read and efficient.
- Simply because it's currently the default tool in the ever growing ROS community.
- But version 2.0, we learned what it is missing.
- It is actually not that hard after some time
- its bound with ROS
- As ROS continues to grow, URDF use will continue to grow with it.
- It's a very sensible format but it misses on the ease-of-use factor
- So long simulation environments ask for them I see it grow
- Robotics community is always growing and urdf is a great tool. Simplifies a lot writing all the transformations by hand and is a good way to visualize our robots
- URDF adoption grows together with ROS adoption.
- I saw colleagues using without ROS , and is easy to understand as well for beginners
- Because UDRF has powerfull robot description
- I think URDF will continue to be used as long as the community is able to solve the problems. It's a straight-forward way to represent a kinematic chain.
- Human readable format is the way
- For some applications, e.g. industrial robotics, the URDF format has most capabilities that matter, so most robot arm manufacturers already provide URDF files for their robots. If URDF would be better documented and easier to use, I think it could become a true standard.
- With increase in the accessibility and ease of use of robotic simulators, use of URDF will increase.
- Standardization is key for technical progress. If we have one million methods to represent the same thing, it is worthless. That is why I support URDF, but with more focus on serious software. First time I have used URDF was in combination with ROS and now I am happy to see adopted in serious software too. When it comes to manufacturing ROS is definitely not the way to push forward. It may be perfect for other robotics applications. So I definitely support URDF to be adopted in all the serious design and simulation software out there.
- While more complex and professional and formats may exists (SDF, USD), the URDF is simple enough that most developers can use it and write tools to process it without being lost in the complexity.

- already established standard. practically all robots have it specified.
- Defacto standard
- No other possible alternative seems to be found yet
- hard to use other simulation formats with Ros
- Y
- Yes, if the process becomes less tedious
- Due to its readibility and potential to improve.
- It is more straightforward to use compared to our 3D models (eg,. step or IGS)
- It is already a base for several robots that are being used with a lot of support from the founded communities.
- It's one of the few OSS resources that are really universally accepted beyond ROS. New entries to the field typically support URDF as a first move (e.g. Unity, MuJoCo, etc.)
- with better tools and more consistency urdf will remain popular
- I think this standard is flexible enough that it will continue to be used for many years in the future.
- Currently a standard with all the tools we used
- It is a straightforward way to standardise robot kinematics.
- Sim is important, not sure of an alternative of similar popularity.
- Still the most standard opensource way to simulate robots
- It's a standard that is widely adopted but needs iteration to improve upon.
- No other description file is as widly used as URDF AFAIK.
- Bcz of wide adoption and flexibility
- -
- most of the times robot link CAD models do not match DH parameters
- It is nearly the golden standard of ros already
- Since URDF is the standard now and robotics is going to explode in the future, the use of URDF will also increase.
- It's the most cost-saving way of validating concepts
- As far as I can see it, it's the most widely used standard.
- More common since robots will be more common
- This is the only standard on which everybody agrees on. SDF is evolving in a too strange manner
- Becuase standards are very important for the community, since they allow the understanding of a project independent of our background. That being said, the existence of a simulation standard facilitates not only the understanding of the robot's kinematics but also its dynamics and how the robot should interact with its work enviroment.
- Because ROS supports it
- If it can add the support to the closed linkages and nonlinear mimic joint expressions, I think it can work pretty good for the future usecases
- I
- Broken field ?
- Default ros tutorial format. Lots of legacy examples for deployed arms
- It is a manufacturer-independent format which provides the necessary information for the simulation of a robot
- Has the potential for a standardized exchange format

Do you think that URDF will be more commonly used in the future? Please justify if possible. - No

- New version of representing the robot model will come as urdf is very limited to use in terms of if support to mechanisms, plugins support for simulations
- There are many new software's in the market which don't use URDF for visualization
- Omniverse
- XML does not feel convenient for model development: there are no (to the best of my knowledge) good abstractions (e.g., replication of links)
- several simulators dont use URDF
- Other simulators now have a more robust description e.g. SDF, Mujoco which might make URDF outdated
- There isn't enough value in standardization, so someone will need to invest a lot of money on both technical challenges and on getting more adoption. For example, even Open Robotics has not merged URDF with SDF.
- The robot brands need to develop the correct and ready to use format for easier use in any simulation tools , lot of time is wasted during model creation and configuration. Cad customisation is very difficult can not model and give the required inverse kinematic definitions
- Industrial robotic suppliers want you to buy into their proprietary crap
- Hopefully, other formats like SDFormat would be preferred

- Too many issues indicative of poor abstractions so unlikely to resolve these issues in a timely manner
- I think XML is inherently flawed, and so like ROS launchfiles, will probably move towards more elegant solutions (e.g., YAML).
- Sdf is just better at the moment.
- SDF and other formats are strictly better
- I Hope a more decent replacement comes along
- I think they'll be substituted by something simpler/easier to generate and mantain
- Cc
- I think the description can be formatted in a cleaner way. A lot of ros is cool but hastily implemented and messy as a result
- No
- I think a new better tool should be developed
- Cuz people tend to invent easier stuff
- Better SDF for the parallel structure
- I think python may overcome the use of urdf due to the simplicity and the performance
- USD might be nicer.
- It's tough for the format to evolve, it will become obsolete quickly
- It is too limited. How to model looped systems with for example a linear actuator? We need something else.
- .
- It does not support the description of actuators and sensors which many times should be simulated along with robots.
- I imagine everyone has a proprietary robot description or use something thag comes with their simulation like. Unity or just code all the transforms and everything themselves
- the Mujoco format (MJCF) is being supported more and more these days, such as in IsaacGym.
- SDF is better documented and more general
- why not switching to the better maintained SDF?
- SDFormat is a better alternative
- Another file format will dominate
- It's a poorly designed format that only continues due to legacy. We are just waiting for a better format to come along
- No
- I think USD files have a lot more potential.
- It is too limited for joint structures found in common products
- It is being superceded by newer formats (SDF, MJCF)
- In is too cumbersome to use.
- Though largely adopted, no one seems to like it and extensions/alternatives are often proposed
- not without significant improvement to the above things
- Without standardized support of closed kinematics...
- Because there is no standardisation and little updates (that I know of)
- O
- Broken field?
- URDF has too many problems and currently they are better ways to describe your robot like SDFs

Do you think that URDF will be more commonly used in the future? Please justify if possible. - Don't know

- Common standards could emerge
- For now I consider URDF files a common standard in robotics but maybe a new file format will appear
- there are other representations built upon the URDF which have been proposed
- URFD seems limiting with respect to modeling systems with more intricate dynamics, like soft and deformable manipulators object etc
- Possibly yes, but not sure.
- I think it needs improvement to be more universally used
- Will SDF obsolete URDF?
- Extensive adoptation by industry will be key to more common use, but such adoption is questionable (it is simply too much of an expert tool for now).
- Depends on the firection of the industri
- ROS 1 is end of life, but the fate of ROS 2 is far from certain.
- No idea, but it is popular in academia
- There is no competitor, but the steep learning curve and current available resources for learning make it a good incentive for an alternative

- its main benefits right now are the wide support on many different robotics frameworks and relative simplicity / human read/writeability. Maintaining that simplicity and portability will be key to making any more complex features still add value to the community.
- Possible, at least this kind of robot description is very useful
- not experienced enough in the industry to know
- There should be more standardized formats gaining popularity soon
- If the improvements suggested above are made, then probably, yes. The format is closely tied to ROS, and so the growth and adoption of ROS will likely see an increased adoption of URDF
- There might be easier ways to describe the robot
- Not allowing parallel linkages is a big issue
- B
- I use them a lot and see the potential, no idea if any industry uses urdfs
- Don’t know
- Don't know
- URDFs are currently acceptable, however, they have limitations which makes me think progress will create a tool to address these limits
- Might be more tools to help with creating urdf
- It works. It sucks, but still. It works.
- Yes if tooling becomes better.
- It's difficult to find an alternative for de facto standard even if there is a better one.
- SDF seems to superceed URDF in many usecases, and I'd like to see one be more "standard"
- Depends what you want to do
- We're developing a much better format, BRICK
- not enough experience
- There are some initiatives looking for an alternative.
- Mainly use SDF for the moment
- If it has modifications that allow developers to improve its work, I think it may be a useful tool for simulation environments.
- Don't know of other alternatives, but would definitely consider other if provided better usage.
- Urdf can evolve with the right pluggins and tool to be convinient. But it can also be replaced with a better method
- URDF fills a void, however it has everal limitations and XML itself is getting dated.
- Have not worked in industry to understand what current trends are.
- The URDF format is fairly generally adopted and is usually the common one between simulation tools; I am transitioning to prefer it as a base for my work.
- It's widely used, but has significant limitations. If updates are made to the URDF standard, I can see it receiving wider use. Otherwise, if another more comprehensive standard gains traction, URDF's use could fall off.
- who knows, it’s far from perfect
- Dn
- Unless is greatly improved, other robot description format that fills the gaps that URDF has can take over
- There are exporters or scripts as Fusion2URDF that can avoid the necessity of creating a urdf from scratch
- Please no, we need an easier solution
- It’s a commonly used robot definition files in many simulations tools, even though it may not be the best way.
- There are many other options that seem to overcome the disadvantages of URDF
- I usually put up with its difficulties to continue research.
- In the industry, it will be dictated by the supplier which just want to make it simpler for their customer
- yes, but there might be some better solution that I dont know of and which renders URDF obsolete..
- URDF are in comparison fairly easy to setup and start simulations, while there are other preferred tools for better simulation of physics in the environment like Unity. But the complexity increases with these additional requirements. So in my opinion I do not see a de facto tool for setting up models and simulations
- I hope so.
- It is difficult to predict software trends
- There are competing standards like SDF and XML, hard to say which will be most dominant
- Hope for a simpler innovation
- I don't see it being replaced, so I guess the status quo will remain.
- I feel that URDF is already used quite commonly within the ROS community.
- I now use Visual Components for all my simulation needs. Obviously this is an expensive software, but to create tasks and workspace geometry the time savings is massive. For URDF to be the industry standard, building the application needs to be made easier.

- If not evolved I believe it will be replaced by a more flexible and capable format
- The main value of the urdf standard is that robot oems understand it and will use it to publish their cad models.
- It’s useful but hard to create customized robot models
- Probably depends on ros at this point, if it stays mostly inside of ros it won’t change much from how it’s used now.
- It is not straight forward to extend a model with custom patameters
- There are companies pushing for other standards (e.g. USD) so I'm not confident.

E-mail

Overall Status

	Percent	Respondents
New	0.0%	0
Distributed	0.4%	3
Partially Complete	25.1%	174
Complete	74.4%	515
Rejected	0.0%	0
Total	100.0%	692

Collection Status

	Percent	Respondents
None	0.4%	3
Partially Complete	25.1%	174
Completed	74.4%	515
Rejected	0.0%	0
Total	100.0%	692