All Summaries in One Doc

1.1

11; The participant opens a terminal, and goes to the catkin workspace, and builds the project.
11; He then launches the TurtleMimic.
10; He then investigates the terminal output from each of these.
11; He then executes the movement publish command, and
12; notices that only one turtle moves.
P; He says he has not encountered a similar error before, and says
3; his first thought is that it may only be publishing to one turtle, or that there is code in the moving turtle that is failing to send movement commands to the stationary turtle.
P; He says he has used turtlesim before.
4; He then opens up another terminal, and looks for the source code for the turtlesim, hopefully a c++ file.
12; He finds turtle.cpp, and opens it in nano.
7; He scrolls through it.
G; He says he is not looking for anything in particular, but is simply looking for something that doesn't look right.
11; He launches rosnode list to
2; see which nodes are executing, and then
10; looks for info about /turtlesim1/sim which tells him
12; the subscriptions for the node, and
13; notices that it is subscribed to the rostopic pub publish command he executed earlier.
11; He then does the same for turtlesim2, and thinks he
13; notices that turtlesim2 doesn't know the type of the topic because he sees ['unknown type]' in the terminal output.
9; He then decides to look at the launch file, and
4; searches through the directory structures for it.
7; He notices the remap directive tags, and says
3; 'this has to be the problem' 00836 referring to the fact that one remap has an input remap tag to turtlesim1 and turtle2 has remap an output remap to turtlesim 2.
3; He says that the 'launch file was wrong', that 'you weren't giving it an input, you were giving it an output'.
11; He then tries the build and launch process again, and
12; notices that one turtle is still not moving.
6; He tries another edit to the launch file feeling he made a silly mistake, and
11; tries the build and launch process again.
12; He notices that one turtle is still not moving.
2; He says that 'it is publishing only to turtle1, right?'
A; He opens up the launch file and explains that he changed the input and inputs, and then

6; he decides to change them back.
3; He explains that the original could be correct, if one is meant to pass on.
3; He explains that the information is only passed to one turtle and that the other turtle is supposed to relay it to the other.
3; He says the only way to find out is empirically.
A; He then draws a diagram about his understanding of the turltes.
He draws two turtles, and
3; presents two opposing altnertaives of possible program architectures, one where the information is published to each turtle, and another where it is relayed on from one turtle to the next.
4; He then says he doesn't know which files to findthis out from, and that is what he is looking for.
11; He then lists the rostopics.
12; He explains that turlte2 is not getting the info because turtle1 is not getting the information.
3; He expains that the first idea is that the first turtle would send it to turtle2 (linear), but that now he is pursuing the idea where the mappings are broken (direct).
1802
7; He now opens up the launch file again, and says he is pursuing the direct model.
4; He then explores the directory tree.
G; He says he is not quite sure what he is looking for.
M; He says that 'many times when you see something that isn't supposed to be there, you can tell'.
G; He says he is generally looking for something that looks out of place.
7; He then opens up mimic.cpp.
7; He scrolls through the file, investigating and clicking on random things.
2154
7; He then says he is looking at code, looking over it and perusing.
13; He says that turtlemimic.cpp is basically mimicing but that
2; he is not sure what 'pose' does.
13; He realizes that it is publishing 'twist', which he recognizes from another file.
10; He then switches to another terminal, and looks at the twist command.
4; He then looks through the directory tree again, and says he is not sure if mimic.cpp will be helpfu later, but says that at first glance it has nothing wrong.
4; He then looks through the directory tree to find turtlemimc.launch again.
7; He reads through the file.
2506
6; He then decides to edit the launch file topic remappings, and the
10; looks at the rostopic list, and compares the two.
3; He explains that they were either published wrong or he was remapping them wrong.
11; He then rebuilds the project, having changed the remappings, and
notices that it still doesn't work.
2; He says he is not sure if the nodes might be registered wrong.
7; He then looks at the launch file, and just reads code, trying to understand it.

3034.
7; He then opens turtlesim.cpp again, and looks through it.
10; He looks at rosnode info for one of the nodes, and still
13; notices ‘unknown type’, and wrongfully deduces from this that it is not getting the published information.
11; He runs rostopic info, and looks at the
10; connection between one terminal window for rosnode info and the rostopic info.

M, 13; At this point, I tell him that he should use the linear model. (**help given here**)
13; From this he deduces that it probably has less to do with topics.
4; He then says he is trying to find a file which is relaying the information, and thinks that this will be mimic.cpp.
3729
4; He says he is search again for a file that he just saw earlier, mimic.cpp.
7; He opens mimic.cpp, and starts scanning the file to understand it better.
7; He looks at the mimic::poseCallback method.
7; He then looks at the Mimic::Mimic method, and
2; tries to understand why there is a ‘1’ in the nh.advertise method.
4; He is now looking for other files which might relate to the problem or to mimic.cpp.
7; He opens up the multisim.launch file, and looks at it for a while, then closes it.
4; He then looks through the directory tree, and opens the
3; turtlesim.cpp and says ‘maybe the turtlesim node is relaying the information’, and
7; checks to see if it can accomplish that.
7; He then opens up turtle.cpp, saying that he might have missed something.
2; He says he is still looking for a part that might relay information.
7; He scrolls through the file, He looks specifically at te function call pos_setY(std::min…)
12; He says it ‘cannot be this’ because he says he can’t find any evidence of a publisher or other evidence that the problem could exist in the turtle.cpp file, and thinks the code which relays movement is somewhere else.
M; He says that he thinks he is missing something simple, because if one turtle works fine,
3; he thinks the main (turtlecpp) code is correct, and believes that the problem will be in mimic.cpp.
4; He then looks through the files to find the turtlemimic.launch file, and says that from a brief scan, he isn’t sure the mimic.cpp file has an error, and thinks that turtle.cpp file is correct because one turtle works, and because of this he thinks it has to do with the remapping done in turtlemimic.launch.
3; He says now that he thinks the problem is with mimiccpp
3; but that the problem is related to the launch file.
7; He now has both of these files open on his screen.
M; He also infers from the stated goal of making the turtles mimic one another
3; it would make sense for the error to be in the mimic folder since this isn’t working.
7; He is now examining the mimic.cpp file
7; looking at the input_nh(input) and compares this line to the launch file.

P; He also says that he hasn't worked with callbacks before
P; saying that his experience is primarily with services and service calls.
7; He then relates other lines in mimic.pp to the launch file
6; and says that he has now reverted the launch file to its original form
12; saying that he changed things to empirically investigate the effects in order to better understand the program.
7; He is now relating the calls in the Mimic::Mimic() method to the launch file.
P; He also explains that he hasn't dealt with topics before
P; and that his main experience is with services.
P; He also says that he hasn't worked with callbacks in any language before.
P; He also says that his ROS experience is to do with python.
7; He says he is now investigating the callback in mimic.cpp
7; trying to do what the callback does and what its purpose is.
P; He then relates this back to his prior experience with services
P; and relates his knowledge about services in order to understand how callbacks work.
M; He says that he feels an example would be online
M; but decides not to do so
M; because he likes to learn things 'the hard way'.
N; 5520 After a long pause
7; looks at the private poseCallback and then looks at how this relates to Mimic:Mimic cpp.
2; He then asks questions to himself
2; about the input and output funciton calls in Mimic::mimic
3; saying that he thinks that in Mimic::mimic
3; output is turtlesim 2
3; and in this case turtlesim 1 is subscribing to the Twist publishing
2; wouldn't turtlesim 2 subscribe to the turtlesim 1 which is supposed to publish.
2; He then says he thinks he might try switching the input and output in the the twist_pub and pose_sub lines in the Mimic:mimic constructor
2; but he doesn't want to make a decision without too much background.
6; He then goes ahead and witches input and output_nh anyway
6; knowing that he can reverse them later.
11; He then builds and tries to run the system
M; saying that he is not sure about what the result will be but says that it may or may not work
12; but it will narrow it down to a 'possible definition of a callback'
12; or what a callback is used for.
M; Thinks this strategy of understanding might not be effective but is helpful when you don't know whats going on.
11; He then tries to build it again
11; which succesfully completes.
11; He then runs roslaunch
11; and the turtlesims come up.
11; He then runs the publish command
13; and only one turtle moves.

2; The turtlesims do not come up in order left to right but stacked
2; so he wants to know which turtle is which.
2; 10301 He is trying to figure out why it didn't respond to his modification of switching the input and output.
2; He says that he expected something to change.
7; He spends a while looking over the mimic.cpp file
7; looking at the PoseCallBack file.
P; He again refers to his prior knowledge of service to explain that callbacks are passed to handle calls to a service.
3; By process of elimination
3; he believes that there is a problem in the main method.
7; He then goes to the turtlemimic.launch file to compare it to the mimic.cpp file.
2; He then asks if there are any syntax errors
H; and I say there are not.
7; He is looking at the main method.
2; 1:10 He says he wants to figure out the way the node starts
7; pointing to the main method in mimic.cpp
12; and he knows this because it has ros::init().
12; He says that at this point he is pretty sure that the problem is in mimic.cpp.
M; Investigator says that he is going to conclude the study at this point.
M; The participant gives a timeline of this task as such: he started by looking at the launch file
M; where 'everything starts' 11646 After that
6; he tried to modify the input and the output in the launch file
2; 'before understanding how mimic.cpp worked'.
M; Then he went to look through the actual files
M; to investigate the structure 'whether it was linear or tree based'.
M; He looked into the files
M; in turtlesim
M; to see if something wrong was going on
M; 11750 based on intuition and prior experiences.
M; At some point Investigator divulged that it was a linear model
M; and the participant explained that knowing this
M; he knows that turtle 1 needs to publish what it is receiving to turtle2

1.2

11; Coded Storyline and Summary: 1.2SummaryCoded Storyline the participant starts by running the commands on the instructions.
M; Before executing the publish command
M; he sets up his desktop
8; and parses and thinks through the publish twist command.
11, 13; He executes it and notices that only one turtle is moving

13; saying ‘clearly it doesn’t mimic’.
4; He then says he is going to try and figure out where everything is
4; looking for source code for the turtle sim and turtlemimic.
A; He goes to the beginner_tutorials/src directory
12; and notices that it is empty.
G; He then explains that he is looking for a ros command which will help him find the source code
M; and explains that he dislikes that ROS is trying to isolate him from the filesystem
M; because it cannot isolate him fully.
11; He tries to use rosedit
L; but it won’t run on turtlemimic.launch.
G; He says he’s trying to open the editor in some meaningful spot
4; looking for the code that describes ‘the behavior of these turtles’.
P; He says that he also knows that ros has ‘many parameters’
11; so he tries rosls.
14; He then says that he knows that because he exectued a launch file
4; he will go into that launch file to see the name of the packages that that launch file is running.
13; He explains that he notices two launch groups
13; turtlesim1 and turltesim2 one node each
13; and then some topic remapping.
13; He states that ‘the first thing doesn’t do as the first thing does’.
3; He explains that either the first one is not publishing or publishing something bogus
3; or the remapping was not done well
3; or the second one is not consuming it properly.
3; He guesses that the code for each of them are actually the same
3; but that the outputs of one are fed into the inputs of the other
4; saying that he really needs to find the code for the turltles and also for the remapping.
4; He says that since he already knows where the remapping is done
4; he’s gonna look at it first.
7; He says that he doesn’t think the remapping makes sense.
11; He says he’s gonna run it again and see what topics are running.
7; He knows that the input has to go to turtlesim1 and the output has to go to turtlesim2
7; pointing to the launch file remapping.
11; He then executes rostopic list
12; and then he says that there is turtlesim1 which is one group and turtlesim2 which is two group
G; saying that he doesn’t understand the symantics of the remapping
13; ‘all the stuff that goes into input goes into turtlesim 1 and all the stuff that goes into ouput goes into turtlesim2’.
8; He then decides to read the manual onlline about the remapping commands googles ‘roslaunch remap’.
8; HE opens the official wikipage on the remap tag.
P; He says he hasn’t understand remapping before

P; saying that he's only written launch files before using copy and paste.
P; He hasn't understood what it means to remap fully
P; in particular
2; if 'input' is some predefined constants from ros
2; or some 'random weird names'.
2; He says that he's seeking to understand the turtlemimic.launch file.
2; If they're constants
4; he says he will have to look inside the code to see what they mean.
2; He googles roslaunch remap input and finds a page on rosanswers which says 'does roslaunch support remapping…".
8; Then he goes back to the remap ros wiki page
8; and starts reading the example
N; and pauses.
M; He criticizes the construction of the remapping tag
M; because he says it is opposite of publishing
M; that you are remappying from one topic to another
M; so one would expect if somebody was publishing to topic one and you are remapping it ot topic two
M; you would expect it to send frmo one to two
12; but instead the syntax is opposite.
2; He says he was trying to investigate the meaning of input and output
12; but seems to have determined that they're not standard tags
12; but user defined instead.
2; He then Googles 'turtlemimic ros example'.
8; Amd finds "using rqt console ros launch"
8; and scrolls through it
12; and finds that the ros launch file seems to be alright
8; and then reads the explanation.
M; He says the explanation is bad
12; but deduces that the root cause is in the source code of the node
12; or in the ros launch file
12; but he thinks this is alright.
4; He then traverses through the file system with the console to try and find the source code for the node.
12; He then finds the makelist file
4; to try and find the path to the source code.
12; He says he finds it ammusing that he can't even find code to debug.
7; He scrolls through the makelist file
N; and then closes it.
4; He then looks in beginning tutorials again.
7; He then looks at the package.xml
7; and searches it for 'cpp'
12; but sees that the path for the source code aren't there either.

4; He looks in /include
12; and can't find anything there either.
M; He then says that he wants to take a step back
M; and decides to start from the root
M; catkin_ws
M; and then says he will explore.
4; He then looks in /build.
11; He then says he might run the 'find' command.
P; He says he thinks he has had the problem of not being able to find the required source code before.
2; He then googles 'ros locate file'
8; and opens 'navigating the file system' from the ros wiki and 'how does my program find the fild system path to a package' on ros answers.
8; He looks at the wiki site
11; and scrolls though and finds 'rospack find' command and tries to try that in the terminal.
L; He doesn't find it helpful.
11; He then tries rospack find turtle.
4; He says that he's trying to use rospack to find the location of rospack packages.
7; He then looks at the turtlemimic.launch files again.
4; He says that he wants to know where the implementation of the turtlesim node is
8; and then looks at a ros wiki page again.
P; He says that he has used turtlesim before.
P; He says that the problem is that he has only installed it using binary files
P; and then run it.
7, 8; He says he's going to compare the launch file with one he found on the ros wiki page.
M; He says that if we're running it from binary
M; he wouldn't know where to go next.
3; He says that if the node isn't customized
3; than everything should be working.
12; He says that the rosfile looks good
8; comparing it to what he found online.
4; He says that he's not sure what the code he's looking for would do.
L; Then he explains the structure of the system
L; saying that the rosnode is listening to topics and publishing to topics
L; and listening to the Twist command which was published.
4; He says hes looking for source code that publishes things
4; and that this would be the code that the other turtle is picking up on and actuating.
11; He then tries running the publish command on the other turtle
12; he sees that it works
12; both turtles are moving but not mimicing one another.
11; He then tries concatonating two movement ocmmands
11; one for each turle
11; so that they both move at the same time.

12; This doesn't work
8; so he looks up online how to concatonate two terminal commands
M; knowign that this will not solve his problem.
2; He says that he's trying to understand if by themselves
2; given equal input
2; if they would work.
2; He's trying to understand the symetry of the system
M; saying that they're two things
M; and that they're supposed to communicate.
M; He says he's not sure what he's trying to accompllish by doing this.
2; 2747 But he still tries to find out how to run two commands at the same time to publish one command for each turtle.
12; This doesn't work
M; and he dismisses it as pointless.
4; He says he would like to look at the codes for the turtlenodes
2; to see what they're publishing to and what they're subscribing to.
2; He googles 'ros locate source code' and then 'ros locate package source then 'ros locate package'.
2; Then
2; 'ros locate node'
8; and pulls up a ros answers site can't locate node in package'.
8; He scrolls through it
8; and says that the answer isn't useful.
11; He then tries to see if he can use the 'find' command but says he doesn't know the command.
11; He then returns to the catkin_ws file and runs the find command for "*cpp"
L; and notices that he went 'down the wrong tree'.
M; He says that he gave up on the ros commands
M; saying that they were trying to isolate him from the file system.
12; This showed hi mthat the source code was in the turtlesim package
12; but he had thought that the turtlesim was the infrasturcture for the simulatinon but not the actual thing that shows him the window.
4; He now traverses to the turtlesim package to investigate the source code.
7; He then looks at the __init__.py file
N; then closes it.
4; He looks for the file that defines the turtle as a node
7; and opens turtlesim.cpp
2; and asks if it uses QT as the framework.
7; He scrolls trhough the source code
3; and he thinks that it starts the process in some simple wrapper
3; the frame that draws stuff.
7; He then looks in turtleframe.cpp
7; and scrolls through it

12; and finds it definitely using QT.
7; He continues scrolling through it.
7; He searchers for init in the file
12; and finds taht there isn't any in the file.
7; He then searches the file for "ros"
12; and finds a NodeHandle
4; says that he is trying to navigate somewhere where rosinit is called
4; where presumably subscribe and publish commands are going to be.
13; He points out the 'spanturtle' command where it is creating a turtle
2; but isn't sure if it's doing things.
3; He thinks that the turtle class is doing publishing and subscribing
3; and scrolsl through turtlecpp to see if it is doing this.
12; He finds that it is not
12; but is instead an encapsulation for just more drawing.
13; He notice that it gets a nodehandle
2; and this suggests to him that he should go back to turtleframe to find out where nodehandle is originating.
2; He says that he is trying to understand where all the node initilization is occuring
2; rosnode init.
P; He is looking for this because in his experience
P; when you use rosnode init
P; you use nodehandle to do publishing and subscribing.
3; And he thinks that publishing and subscribing is noting going well
3; becaues the second turtle is clearly not picking up what the first turtle is publishing
2; and he would like to debug it and see waht the first turtle is publsihing and what the second turtle is picking up
4; and for that he needs to identify code that operates this
4; and he isn't sure where this is.
7; He says 'let me klook at the class fields' in turtlecpp.
M; He says the problem with cpp is that the header files are somewhere else.
4; He says he's looking for the header file for turtleframe
4; becauese he thinks there may be thinks like the nodehandle.
4; He found turtleframe.h
13; and says that this is a much better summary of all of the methods
13; and finds that turtleframe knows how to spawn turltes
13; and also sees lots of ros servers
13; saying that he expected a topic.
13; He says that he's not sure the remapping command relates to topics.
11; He then does rostopic list
13; and this tells him that for each turtle there's a few topics
13; and he says that he thinks one turtle should be telling the other turtle.
4; He says that he is looking for the code that subscribes and publishes these.
2; He searches the internet for how the grep command works.

M; He says that he wished he had eclipse
M; because he isn't used to programming in vi.
11; He then uses grep to look for things that do something with 'pose'.
10; This lists many things that contain 'pose' 4353'.
12; He says that this output shows him he missed the publishing and subscribing in the turtle cpp.
7; He then goes to the turtle.cpp and then looks for code that writes the pose.
3; He then explains that a hypothesis is that one turtle communicates the pose to the other turtle
3; and that it mimics this.
M; He says that the ros directory structure is horrible.
12; He finds the pose advertising in turtlecpp
12; along with many other things that it advertises.
7; He says that it publishes pose
7; and subscribes to cmd velocity
2; then asks 'why would the turtles mimic eachother then
2; because they're not listening to the poses
2; instead just telling what the pose is
2; that the turtle is not interested in the pose.
3; From this he dedces that there must be some other code that stiches them together.
7; 4610 He then looks at turtle.h
7; and scrolls though
13; seeing that it publishes the color and the pose.
77; He then decides to look at the mimic.cpp file 4744 from the grep 'pose' command.
77; He then opens it
77; and says 'well there we go
12; there's the input chanels and says he's found all the loose ends
12; the input and output chanels
12; why they're getting redirected [remapped] if they don't even exist'.
M; He thinks that the grep command really helped him out as an exploration tool
M; and that they really should have a ros grep command.
7; He reads through it
13; showing that they'res a twish publishre that's clearly supposed to be a publisher for some other command
13; and that tehy'res a pose subscriber which must act on what it recieves.
13, 7; He then looks at the poseCallback and says that its taking whatever one thing is doing and publsihing it as a twist command.
13; He says this tells him that we're making a turtlemimic node in the init in the main method
12; and that this was the rosinit he was looking for.
2; He also is confused about the Nodehandle thing
3; and that maybe there's a node called input.
2, 11, 12; He is looking if there's a node called input,so he does rosnode list but sees that there isn't.
13; Based on what he saw in mimc

2; he says he's puzled about what the input and output is
2; and he says the next thing for his investigation is why there is only spinonce.
A; He then starts drawing a diagram to visualize what's going on
A; on paper.
A; He draws turtle sim one and two
A; and draws lines to connect them.
2; He still cannot tell if input and out put are nodes
2; then asks if they are
2; then why is there no init for them.
2; So then he tries to udnerstand what the Nodehandle class does
8; and pulls up the ros wiki page of the same name.
8; He reads that it makes an internal node and provides namespace resolution.
M; He says he's suprrised that there's such thing as an internal node
M; or what this is.
P; He says he hasn't written ros cpp files before
P; only rospython
2; so he tires to understand how roscpp files work.
8; He reads through the nodehandle wiki page some more.
12; He thinks now we have two subnodes
12; one input subnode and one input subnode.
13; He explains that the output subnide is advertisng velocity on cmd_vel.
7; He then looks through turtle.cpp
13; and explains that the input one is subscribing to pose.
2; He says he's confused about name resoution
4; and decides to go back to th turtlesim.h file.
7, 13; He then opens up turtle.cpp He notices that it also subscribes to cpp velocity and publishes to pose
2; and that he is trying to find the overall picture of publishing and subscribing to chanels.
2; He then says that wants to know the intent behind the whole code.
3; He says that tehyre suppsoed to publish the pose and get stuff from cmd vel.
7; He then investigates the launch file and wants to think about the subnodes [input/output].
7; He says he's looking at the mimic constructor
13; noticing that it advertises the twist command
13; and that every time something comes into pose
13; it's going to publish something.
2; He says the problem is figuring out with input and output
2; he wants to know which is going into which channel.
3; He say's he wants to start one of the turtles moving/ He then publishes one on turtlesim one
3; and makes one turtle move
3, 11; and then he says 'lets see whats happenign in the topics' and then does rostopic list.
2; Then he investigates if the first turtle is disclosing its pose with rostopic echo /turtlesim1/pose.
12; Lots of output and he says that 'its clearly disclosing pose'.
11; Then he tries the same with cmd velocity

11; the same one with his console command.
11; He then tries the same with turlesim
12; shoing that nothing is happening with the second turtle cmd_velocity
12; because ‘otherwise everything would have worked.
1; He says that namespace resolution is an obstacle
12; that the remapping command substitutes the prefix of the name of the channel
8; and that this wasn’t on the wiki.
2; He says he’s trying to understand the remap command better.
7; He says that from the input namespace
7; we’re subscribing to pose
7; trying to associate it with turtlesim1/turtle1pose.
4; He says that one thought that he didn’t see code that makes mimic.cpp work
4; so he looks for that.
11; He does rosnode list
12; and doesn’t see a mimic node.
2; He says that he is looking for a list of nodes
12; and that he doesn’t see mimic.
13; He says it does say spinOnce
2; but ‘who is calling mimic’
2, 8; who is calling the main in mimic? He says this is strange because there is no such code in a tutorial he was looking at.
2; He asks how can the mimic node be launched if it’s not explicityl told to launch thus assuming that they’res other code which launches it
11; so he decides to grep for that too.
11; He does this
12; and sees that it’s only called in mimic.cpp
M; and says ‘how stupid’.
2; Then he says maybe he’s not undertsanding the semantics of the roslauch
12; then he notices that mimic.cpp is actually launched in the launch file.
12; This confirms for him that the mimic node is launch.
3; So now he says that the spinonce line is suspect
1; because he thinks this is ‘run it once and forget’ and that this is not what he wants.
3; So now he sees if when he separately runs the node
3; if it will run just a little bit.
6; He decides to make a roslaunch node just to launch the mimic node
6; to test this.
6; He does so
11; and calls it
12; and it doesn’t move.
N; But
13; he notices that the thing dies
1; which he thinks suggests that maybe spinonce isn’t the right thing
3; that spin would make it continuously communicate.

6; He makes this change to just spins
12; and it works.
N; 10911Timeline begins at 10957

1.3

N; Coded Storyline and Summary: 1.3SummaryCoded StorylineBegins first task.
11; Makes the project
11; runs the project
11; and publishes the Twist command.
M; Decides to adjust settings to allow multiple spaces.
2; He says he wants to figure out what the two turtles are
2; if they are two different instantiations of the same thing or if they are completely separate items.
4; He says that he will explore the launch file to find this out.
7; He opens the launch file.
12; He says that the launch file shows them to be from the same package
12; and the same type
13; but from different namespaces.
11; Runs Rostopic List
12; to see if the Twist topic is specific to only one topic
12; and not the other turtle
3; to see if he has to publish a different topic or to a different turtle.
13; He notices that there are many topics named turtle1.
2; He says he's unsure about the seemingly duplicated structure of the topics.
12; He says that it appears that based on the launch file
12; the turtles are just different instances of the same package.
7; He sees that turtlesim1 has three things
7; and turtlesim2 has threeish things.
1; He says that he will simply try running the Twist command twice to make each turtle move.
H; I clarify that this would not be a successful fix
H; given the parameters and goals of the experiment.
A; He installs Vim
M; explaining that he uses it frequently for remote debugging over weak or unreliable connections.
H; He asks if he can modify the launch file
H; and I tell him that we're looking for people to fix the problems in code.
7; He opens the turtle.cpp file and scrolls through it
2; and says he is looking for what it does when it gets the Twist message.
12; He finds a callback
12; and says that he thinks the Twist command comes to that callback.

13; He finds that it is subscribing to cmd_vel and says that this is something he wanted to check and that it is something he expected.
13; He says that he learned how the file handled velocity
13; and what it does with it.
2; He also says he's trying to figure out how the turtlesim simulator works in general
2; would like to 'understand whats going on'
2; for example what Pose_pub is doing
2; is there two instances for each turtle? He says he's still trying to get a sense of what everything is in here.
13; He is thinking that turtle.cpp is not doing any specific references to namespaces.
7; He looks through turtlesim.cpp.
7; He then looks through other c++ files.
7; He then opens the launch file again
6; and tries putting both turtles into the same namespace
6; giving one a different name.
3; He wonders if putting them in the same namespace will get the same message and execute them.
11; He builds and runs this
12; and it doesn't work.
11; He tires rosnode list
13; noticing that it didn't use his new launch configuration
13; because they're still in different namespaces.
1; He says he founds that he modified the wrong launch file
6; and tries doing it with the correct one.
7, 2; He looks at the mimic launch group in the launch file and says that he doesn't actually know what it's doing.
7; He tries modifying the mimic launch group
13; but this works.
H; I tell him that the intent is to fix it using cpp files.
M; He says that the way the task is set up doesn't quite represent reality
M; with the launch files.
3; He says that an idea he had was a node that would listen then republish the info
M; but thinks this is too much work for the scope of the study.
P; He says he's not too familiar with namespaces within the program level.
7; He looks through some cpp files.
P; He says that he usually uses CPP for actual ROS work
P; partly because it runs faster.
7; He opens up the turtlebot.cpp file
7; and looks for anything that stands out.
2; He says that one thing he could do is look for cmd_vel in the file
12; but doesn't find it.
P; He says that he's worked with systems before where messages are subscribed to and then handled in a call back.

7; He then finds cmd_vel in the turtle.cpp file.
7; He then traces to the velocity callback.
7; He peruses the file some more.
6; He puts in print statements
6; with the cpp printing system
6; in the callback for the velocity stuff
3; in order to see if the callback is being called twice
3; once for each instantiation.
8; He looks up the include for cout
1; but then finds he simply made a mistake in how he wrote cout.
11; He then builds and runs the task
13; but notices that he doesn't get the print statements
3; but then he realizes that he thinks the print is getting lost somewhere.
2; He then looks through the logs to see if it printed the cout statement
P; and says that he uses logs a lot for his work.
12; He says that he doesn't find it in the logs.
8; He then googles 'ros logger debug'.
6; He then tries using "ROS_DEBUG" statements instead of cout.
11; He builds and makes the project.
3; He says that he expects the ROS_DEBUG to print whenever he sends the message
3; twice (one for each turtle).
6; He tries ROS ERROR instead.
11; He builds and runs the program
10; and it prints his debug statement statement multiple times
13; and he deduces that it repeats because the Twist publish command has the '-r' flag.
11; He takes out the '-r' flag and sees that the print statement only prints once
12; meaning that the callback is only called once.
7; He then looks in more detail at the turtle.cpp code
2; trying to figure out what happens.
12; He says 'it gets a velocity command
12; and then gives out a pose command to move
2; one thing I was thinking was was this pose going out to both of those?' He says he's not sure how it'd be getting to both of the turtles.
2; He say that he really should figure out what the <node … mimic> tagged group does in the launch file.
3; He says he thinks that it does is anything that comes into turtlesim just gets remapped.
2; He says he is confused why the Twist command is being published to turtlesim1.
8; He opens a browser and looks up 'node launch remap' and opens the official <remap> tag wiki entry.
12; He says that he verified that it takes whatever is in the input and sends it to the output
12; and where everything goes.
2; He says he wants to check whether the messages are named properly.
1; He says he might add another handler that would listen to pose and then act accordingly

1; in the turtle.cpp file.
M; He doesn't do this.
7; He says that 'nh_.getNamespace()' got his attention.
M; He says he doesn't think that the program is good
M; because if you're sending a command to turtlesim1
M; it should only affect turtlesim2.
M; He then says he isn't sure
M; and that he wants to move on to the next task.
M; I explain that there is already a node which passes on the info from sim1 to sim2
M; and that there is a problem in that file.
H; I ask him if he would like to debug try and debug this file
H; mimic.cpp.
4; He tries to find mimic.cpp
4; and then opens it.
7; He says that it looks like this file 'mimics'
7; and that it is what he assumed it was doing
7; passing on the info.
7; He looks through the file.
11; He then in a new terminal does rostopic list
11; and then rostopic info.
11; He then does rostopic echo to look at the format of the pose message.
2; He says he thought the format of the Pose message was different.
2; He says that he isn't sure what the input and output tags are doing
6; and tries switching the inputs and outputs for the twist pub and pose sub.
11; He then tries building and running the project again.
13; He says he recognizes the input and output from the launch file.
11; He then publishes the Twist command
12; and it doesn't work.
M; He then wants to know what's wrong and move on to the next task.
M; I explain that it is doing spin() and not spinOnce().
M; He says that he looked at it
M; in the turtle.cpp we're looking at
M; but not in mimic.cpp.
M; He then suggests that we say things are 'ros specific'
M; don't touch launch files.
M; He then proceeds to give a timeline for the task.
N; 1:01:21

1.5

11; Coded Storyline and Summary: 1.5SummaryCoded StorylineBuilds and launches
13; notices that only one turtle moves.

3; He says most likely some node is publishing to a topic and both turtles are reading from that topic
3; but that the broken turtle may not be reading from the correct topic.
P; He says that he has not worked with turtlesim much
P; but has worked with ROS for one year.
2; He is trying to figure out which topic are being published to
2; and which topics exist
2; and then go from there.
11; He does rosnode list
11; and rosnode info on one of the turtles
10; and briefly looks over the output.
7; He does the same for the other turtle
7; and notices that we're publishing to a cmd_vel topic.
13; He explains that turtlesim1 is subscribing to the correct topic
3; but he thinks turtlesim2 is subscribing to the wrong topic.
11, 4; He uses the 'find' command to look for the turtlemimic.launch file
11; and then he opens it in gedit.
8; He says that he thinks he'll have to look up things on the web to understand how launch files work.
8; He opens up the roslaunch/XML official ROS wiki page.
8; He then opens up documentation on the <group> tag and the <node> tag
7; and switches back to the launch file.
7; He says it helped but that he still wants to read it over more.
2; He says he wants to understand the <group> tag means
2; and whats the large <node> tag understands.
6; He changed the namespace
11; and then launches it again to see what happens.
2; He says he changed the namespaces to see how topic names are concatonated with the namespace names.
2; He wants to know if the topic names change when he changes the namespace name.
11; He does rosnode info on turtlesim/sim.
12; He confirms that they do
12; and that the first part of the topic name was the namespace name.
6; He tries changing both turtles to operate in the turtlesim1 namespace
11; tries to launch it
13; and then notices that it will not launch while they have the same namespace and name.
6; He changes one of the names
11, 13; and it launches.
11; He tries the publish method
12; and it works.
13; Both turtles mimic one another.
H; I explain that we're looking for a fix in the cpp code.
H; I ask him to draw a diagram of how publishing and subscribing works now.

H; [see diagram]I ask him to revert the launch file the original state and try and fix it using cpp files.
4; He says he wants to figure out where the source code is
4; and he goes to the src/turtlesim folder
4; and he says that he is now trying to figure out what file he needs to change.
7; He opens turtlesim.cpp
7; and scrolls through it.
7; He says he wanted to look for the code
7; and realized that we were in the catkin workspace
3; so most likely he thought they would be in the normal directory.
4; He says that he's trying to figure out where it is publishing and subscribing.
4; He uses the terminal to look around for source files
4; not looking for particular file.
2; He says he still doesn't see where the topic subscriptions are happening
2; so he's looking for that.
8; He looks online at the offical wiki 'writing publisher and subscriber' to find out the function you use to publish to a topic
8; and finds the 'advertise' method.
11; He uses grep to find files which call this method
11; many files show up.
11; He does the same grep command but for 'subscribe'
11; but only the turtle.cpp file shows up.
7; He searches within turtle.cpp to look for where the subscribe method.
M; He says he doesn't have a theory right now
M; but that he's still trying to figure things out.
7; He looks in the turtleframe file
7; but doesn't make any changes.
2; He says he is trying to trace the sequence of calls to the constructor in turtle.cpp.
11; He greps for Turtle
11; so see where it is called.
12; He finds it is only called in turtle.cpp and turtle_frame.cpp.
2; He says he is looking for the place where the turtle constructor is called.
7; He opens up turtle_frame.cpp
7; and goes to line 180 to find the place where grep told him the Turtle.cpp constructor is called.
3; He says that he thinks there is some connection between the name in turtleframe
3; and the topic name
3; and he's trying to figure out what this is.
7; He looks through TurtleFrame.cpp briefly.
2; He says he's trying to figure out the connection between the name in 'spawnTurtle' and the topic.
M; [not sure what he's doing this
M; or why].
10; He looks at a log file path in the terminal

10; copys it
4; and looks to see what the directory contains.
11; He then greps for the string ‘Spawning’
12; and finds that it is in two of the log files he found in the aforementioned directory.
11; Be builds the system.
2; He says he’s trying to figure out what the connection between the name which is being passed in (?) and the topic name.
11; He launches the system
1; and does rosnode list.
10; It lists 3 nodes.
10; He then copies a log file printed during launch
7; and then searches for the word ‘spawn’ in that launch file.
10, 7; There are two results in rosout.log.
M; He says he doesn’t know why what he is doing is helpful right now.
7; He scrolls through turtleframe some more.
3; He says he will try changeing the name used in turtlerframe.cpp and see what happens.
11; He kills rosmaster.
6; He changes the name on line 168 of turtleframe
11; and builds it.
12; It builds successfully.
11; He tries running it
11, 2; and then does rosnode info on one of the turlesim nodes He says he thinks he needs to change the namespace for one of the subscribers
3; becasue we’re publsihing to one topic
3; and he thinks we’re subscribing to the wrong topic.
11; He does rosnode info again.
6; He changes the turtle_frame file back.
8; He googles ros namespaces
8; and opens up “names’ in the ros wiki.
8; He looks through the page
8; at the ‘resolving’ section in particular.
8; He looks over the ‘rostopic pub’ movement command again
8; thinking through it out loud.
11; He does rosnode list again
; and sees three nodes.
13; He then does rosnode info /turtlesim1/sim
2; and checks what it subscribes to.
6; He modifies turtleframe.cpp
6; changing the name of an instantiated turtle
11; and builds and runs it again.
1; He says he is trying to make it so that one turtle subscribes to the correct topic.
G; He talks about the diagram he drew
G; talking about how he thinks the system should operate.

7; He looks at the turtlemimic.lauch file
13; and notices that there is a third node which is not being started [mimic] and wonders why this is.
11; He does a find command for ‘mimic’
12; and finds the mimic.cpp file.
3; He says that he thinks the mimic node should be started
3; but he says he should be able to find the executable
2; but he doesn’t find it
3; so he thinks there may be some problem with the compilation.
11; He does find again for ‘sim’
10; and there are many results.
12; He finds an executable named turtlesim
12; and an executable named mimic.
11; He tries running the mimic node separately with ‘rosrun turtlesim mimic’
8; but then stops and looks up rosrun on google.
3; He says right now
3; he thinks that it reads in messages and publishes them on.
8; He looks at the rosrun page on the wiki
8; ‘rosbash’ page
8; and scrolls around.
8; He talks about the remap arguments in the launch file
8; and he wants to figure out to parse these arguments.
11; He does manual remapping in the terminal by copying from the launch file
11; ‘rosrun turtlesim mimic input:=…”.
3; He says he thinks the mimic node should be continuously running
3; but it is not.
11; He greps for mimic
12; and finds the location of the cpp file.
11, 12; He also greps for ‘input’ and finds that the string exists in mimic.cpp.
11; He uses gedit to open mimic.cpp.
7; He notices that it says spinOnce()
7; not spin()
6; changes it
11; builds it
11; and tries running it.
3; He says he thinks it should be continuously running
3; and that spinOnce will handle the node once and then exit.
11; He then tries rosnode list
12; and sees that mimic is running now.
11; He then does rostopic pub
12; and it makes both turtles work.
M; Concluding questions at 1:00:39

1.7

M; Coded Storyline and Summary: 1.7SummaryCoded StorylineNo screen visuals during this recording (frozen desktop under moving pointer)
M; so I am only able to go off of what he says.
M; Also
M; sometimes he said something that I don't understand because I don't have enough context
M; due to the lack of visuals
M; to know what he's talking about.
11; Starts by executing the commands as given on the instruction sheet.
11; He builds the tasks
19; and looks through the build output
19; nothing quite surprising
13; he notices that everything compiled correctly.
11; He tries sourcing the sh file
P; because he often encounters others making the error of not doing so.
11, 13; He starts the program and notices that a process 'finished cleanly'
13; meaning that something excited immediately and probably shouldn't have.
P; He says he may have used turtlesim a long time ago
P; in the ROS tutorials.
11; He then tries publishing the twist movement command
M; typing it in by hand because he says it makes him more aware of what he's typing.
13; He says that one turtle moves continuously in a circle
13; but that the first turtle doesn't move.
3; He thinks there should be two things: a turtlesim2 with a turtle1 listener on it
3; and he doesn't know if its correctly implemented where one turtle is sending messages to the second program telling it what to do
3; and if that is the case
3; there was a program which exited
3; which could mean that something disconnected.
3; He said that there could also be a problem with the original set up where if you send a command to one its not gonna make the other move.
M; He says that assuming there's a tiny bug
3; he thinks that the first turtle is receiving and publishing that message again and publishing that to the second turtle
3; and that the second turtle is an exact copy of the same program of the first turtle.
3; He says that anotehr idea is taht the program is entirely wrong
3; and there's no communication.
11; He tries using a rosnode command
12; and sees that there are two turtle nodes.
2, 11; He does rosnode info on turtlesim1 to see who is connected to who.

13; He sees that its not publishing another cmdvel
13; so they're not talking the way he expected.
13; He sees that turtlesim2 is subscribing to its own topic.
2; He says he wants to know why the think finished cleanly
2; but he also wants to look at the launch files to see if there's another node that should be running but isn't.
7; He opens it
13; and sees that it starts a turtlesim1
13; a turtlesim2
13; and does some remapping
13; and he traces out this remapping in his head.
3; He thinks that in this remapping is where the mimicing is supposed to happen.
M; He says that because we're not editing the launch file
M; he knows the bug is not in the launch file.
M; He says that the launch file is a clue as to expected operation.
4; He says the next thing he will do is look at the code
4; and see what is actually going on.
2; He wants to know what the turtlesim node code is.
12; He explains that in the launch file
12; one program is responsible for two instances of the turtlesim.
7; He looks in the cmakefile to see what the node is.
M; He looks at the cmake file as a 'table of contents'
M; to make sure he actually goes to the correct file.
7; He opens a QT file
3; and he doesn't like QT but he doesn't think there is an error in this file.
M; He likes 'libraries' and smaller
M; well designed frameworks
M; when he doesn't know the details or want to know the details.
M; He defines framework as something which imposes structure
M; and a library as something you can use in a way that you choose.
M; He says that ROS is absolutely a framework.
M; He says that ROS developers make a library or ship a node or a service.
M; He says that QT has many libraries
M; but if you're making a QT app
M; you're using the framework.
7; He looks in the turtleframe file.
7; He looks in the header file
M; because he only wants a high level view right now.
7; He looks at 'update turtle'
12; and thinks that it is 'exactly what [he] wants'.
3; He says that it is looking like the turtleframe.cpp file will contain the error
3; though he's not quite certain.
13; He reads through the cpp file and notices images

13; images
13; callbacks
13; etc.
3; He says that he thinks the subscriber callback is setting something inside the turtle class
3; and then updating its state around the frame
3; and also doing something to communicate with the second turtle.
3; He says that the turtleframe file is doing ‘all the internal work’
3; such ask turning off
3; setting things up
3; etc.
12; He then decides that update() was simply a call inside QT
12; and that it is no longer interesting.
3; He says that because nothing is happening with subscribers
4; he now wants to look at turtle.cpp.
7; He says he is specifically now looking for the place where the ‘position of the turtle changes’
3; which he thinks will be in the subscriber.
4; He says the higher level thing he’s looking for is that he’s publishing to a topic
4; he wants to go to the subscriber of that topic.
M; He says that he is not sure if he’s looking for a bug or for something that is un implemented
2; and he also says he’s very curious about what mimic is supposed to do.
2; He says that mimic could be something that is doing ‘queueing’.
2; He say that it looks like something is unimplemented
2; because he hasn’t found anything yet that does anything like mimicking.
7; He looks at the remap tags in the launch file
P; and explains the ways in which he’s seen remaps used before
P; but explains that he hasn’t seen them used to do mimicking before.
4; He says he’s looking for something that might subsribe to the pose
4; and maybe then use the teleport() to move the second turtle.
7; He says that in the turtle.cpp file
13; there’s only one subscriber
12; and that’s evidence against one turtle subscribing to the next.
7; He sees he finds another subscriber in mimic
13; and looks to see where the mimic code is.
12; He then realizes that this is what died.
12; He then realizes that mimic was actually launched
12; and that it is the think that subscribes and then publishes to the second turtle
12; and says that this explains the remapping.
M; He says he overlooked that the group was launching a node.
M; He says he thinks the similarity in appearance and indentation between the mimic launch statement and the two launch groups above made people not realize it was a separate launch.
3; He says that he says he is 100% sure that mimic.cpp contains the error
3; because it died
3; and it obviously wasn’t mimicking.

M; He also says that having mimic in another folder to the other cpp files also helped him ignore that file.
7; He opens up mimic.cpp
7; and scans through it quickly.
7; He says that we ‘advertise a twist on the output node’
7; and then realizes that we spinOnce() instead of spin()
7; and that because spinOnce() exits.
M; He says that he was not expecting the way it redirects the pose info to the twist
M; and also wasn’t expecting a custom pose class.
3; He says that assuming the pose stuff works
3; he’s 99% sure that it will work.
11; He launches it
10; and notices that mimic no longer exits immediately
11; and then publishes the twist command
12; and it works.
N; Post questions at 33:08-40:43

1.10

11; Coded Storyline and Summary: 1.10SummaryCoded StorylineBuilds task
11; publishes movement command.
13; Notices that only one turtle moves.
N; Opens another terminal
11; and navigates to the catkin workspace.
11; Opens the launch file in vim
N; makes no changes and closes it.
11; Navigates to the source directory
11; beginner tutorials
11; then back to opening the launch file
11; closes it again.
4; Looks around the directories.
7; Opens turtlesim.cpp
7; and scrolls around and then closes it.
7; Opens the launch file again.
7; Changes nothing
7; closes it.
8; Googles “rostopic pub”
8; and opens the rostopic official wiki page.
N; Closes it quickly
7; goes back to the launch file
N; then closes it again.
11; Opens up the source directory

7; then opens turtlesim.cpp.
7; Scrolls through it
7; but doesn’t make any changes and closes it.
7; Opens turtle_frame.cpp.
7; Scrolls through it
7; and searches for “/show(“ but doesn't find it
7; closes it.
7; Then opens turtlesim.
7; Then opens turtleframe again
7; closes it.
7; Then opens turtlesim again.
7; Then opens the include directory
7; looks at turtle_frame.h
7; then closes it.
7; Then goes back to the source directory
7; then opens turtleframe.cpp
7; and searches the file for “subscribe”
7; doesn’t find it
7; closes it.
4; Goes to the devel directory
4; then the build directory.
4; Then the turtlesim directory inside the build directory
4; then back to the source directory.
4; Then goes to the launch directory
7; then opens the launch file
7; closes it.
11; Then tries launching the thing again
13; but only one turtle moves like last time.
7; Opens the launch file
6; then comments out the first remap tag
11; relaunches
13; and notices that still only one turtle moves.
6; Uncomments the remap tag in the launch file
6; then closes it.
4; Goes to the source directory
7; then open the launch file then closes it.
11; Then goes back to the source directory
7; and opens turtle.cpp.
7; Looks through it
7; then closes it.
7; Opens up the CMAKELISTS file
7; scrolls through it
7; then closes it

7; then opens it again.
7; Opens mimic.cpp
7; scrolls through it
7; looks at the twist_pub line
7; then scrolls to the spinonce line
7; then closes it.
8; Then googles “ros nodehandle constructor”
7; and opesn up the official class reference for the NodeHandle class.
7; Closes this
7; goes back to the mimic file
6; and adds a global variable
6; “ros:Publisher twist_pub2;”.
6; Opens mimic.cpp in gedit and adds a corresponding line in the constructor and the callback.
11; Builds
11; and it succeeds.
11; Launches the project
13; but still only one turtle is moving.
7; Opens up the mimic.cpp file
7; then closes it.
7; Opens mimic in geddit
6; changes cmd_vel and pose for the subscriber and advertising lines in the constructor
M; they all now read cmd_vel except the last which reads pose.
6; Comments out bouth twist publishers in the callback.
11; Makes the project
1; launches it
11; only one turtle works.
7; Opens mimic again
6; uncomments the publishers
6; and deletes his added publishers.
6; Closes the file.
11; Launches the project
11; does rostopic list and then rostopic echo on turtlesim2/cmd_vel.
10; It says it isn’t published yet.
11; Does the same for turtlesim2/turtle1/cmdvel
10; and no output.
11; Does the same for turtlesim1/turtle1/cmdvel
10; and there is a steady stream ouf output.
11; Does rostopic list again
11; and does rostopic echo turtlesim2/turtle1/pose
10; and there is a lot of output with varying positions and nonzero velocity..
11; Does the same for turtlesim1
10; and there’s also a lot of output but the velocity is 0 and the pos is constant.
11; Tries rosmsg

10; but no useful ouput.
8; Then gogoles “rosotpic msg type” and clicks on the rostopic official wiki page.
8; Searches for msg
7; then closes the page and goes back to mimic.cpp.
11; Ros rostopic echo again on the moving turtle
11; then again on the stationary one.
1; Hightlights the “once” part of the SpinOnce() in mimic.cpp
M; but doesn’t change it!

1.11

N; Coded Storyline and Summary: 1.11SummaryCoded StorylineSpin TaskBegins at 1:07:54.
A; He installs vim
11; then does catkin_make.
7; He opens the launch file.
11; He then launches the turtle nodes
11; then executes the movement commands
13; and notices that only one turtle moves.
7; He opens the launch file
2; and says he wants to see if the first turtle is publishing any of its commands.
11; He does rostopic list
3; and then says he thinks there are two turtlesim nodes.
13; He says he thinks the mimic node is taking the pose from turtle1 and putting them cmdvel for turtle2
13; based on the launch file.
7; He opens turtlesim.cpp in vim.
7; He looks through it
7; then closes it and goes back to the launch file.
4; He then says he’s trying to locate the mimic node.
7; He opens up the cmake lists
2; because he says this will tell him where the source file is.
12; He then finds the line which tells him this
7; and then opens up mimic.cpp in vim.
7; He scrolls through it
13; and sees that it subscribes to the pose
13; and publishes to cmd_vel
13; which he thinks is understandable.
7; He then looks through the poseCallBack
2; and says it doesn’t make much sense because he asks how you’re going to get velocity from pose directly.
N; He says
N; high level

3; he expects that mimic will take velocity of turtlesim1 and assign it as the velocity for the second turtle.
2; He doesn't understand the pose pointers.
4; He opens another terminal window and looks though the directories
4; and then finds the header file Pose.h
7; he looks through it
7; and sees that it does indeed have components for linear and angular velocity.
7; He goes back to the mimic.cpp file.
11; He then does rostopic list
2, 11; and says he wants to see the type of message that turtlesim1… is and does rostopic info on turtlesim1/turtle1/pose.
11; He then does rosnode list
M; but doesn't seem to notice that mimic isn't there.
11; He does rostopic echo on the moving turtle's pose.
12; He says that this means that turtlesim 1 is publishing correctly.
11; He then does rostopic list.
7; He then looks at the mimic again
3; and says that he thinks it is getting the pose and that it should publish the cmd_vel.
4; He then says he wants to look at the topic remapping.
3; He says that it seems that the nodehandlers have been remapped.
3; He says the turtlesim node seems to be working because one node works
3; so it is able to move and it seems to be able to correctly publish its stuff.
3; He says the second turtle isn't working.
6; He then puts a rosinfo print statement in the posecallback
6; which prints the linear and angular velocities.
2; He says he wants to see if it's working properly.
11; He then builds it
11; and runs it
12; and notices that still only one turtle is moving.
3; He says 'something tells me that it didn't subscribe properly'.
7; He opens mimic in vim again
7; then closes it.
3; He says he doesn't think it's subsribing properly because once once it subscribes it should print
3; and nothing printed.
11; He then does roslaunch.
13; He then notices that a node just shut down.
6; He then changes spinOnce to spin
P; saying that ros spin() keeps checking for a callback on the topic until a node shut down is requested or recieved.
P; He says he hasn't done this much.
2; He says that the spin may explain the node shutdown
10; which he only just noticed in the output

10; saying that mimic shut down immediately.
11; He then builds it
11; and runs it
12; and it works.
N; Post task discussion at 1:31:15

1.12

11; Coded Storyline and Summary: 1.12SummaryCoded StorylineSpin TaskMakes the project
N; it makes successfully.
11; He then sources setup.bash.
11; He then does roslaunch
13; and it launches correctly.
11; He then sends the movement command
N; and in doing so
M; he says ‘by typing it character by character he is more careful’.
13; He sends it and only one turtle moves.
11; He does rosnode list.
3; He says he assumes that when he publishes to turtle1
3; it republishes to turtle2
3; basically take whatever commands its receiving and pass it on.
3; He says he thinks this because to get it to run
3; he only publishes to one topic
3; or alternatively
3; they both subscribe to turtlesim1.
11; He does rostopic list
10; which makes him think that its doing the republishing thing.
11; He then tries the same publish command for turtlesim2
12; and they both spin.
3; He says from this
12; he knows that they both use the cmd_vel correctly
13; knowing that the data simply isn’t being passed but he’s happy that the turtles work.
7; He then opens gedit
7; and opens the launch file
7; he wants to see what nodes they’re launching.
13; He sees the two turtlesim nodes
13; and a node called mimic
12; and he sees from this that this is how the information is supposed to be getting from the first turtle to the second turtle.
P; He says this is mainly intuition.
2; He says he’s suspicious of the fact that its remapping the namespace and not the topic.
4; He says he’s gonna look at the mimic node first.

3; He thinks the mimic node would subscribe to turtlesim1 to the other turtle.
7; He opens mimic.cpp.
13; He then says ‘it spins’ while clicking on spin once
M; but doesn’t pause for thought.
11; He then does rostopic info on the first turle
2; to see what’s publishing and subscribing to each of the turtles.
11; He does it on the second turtle.
13; He mentions that he doesn’t see the mimic node.
10; He then looks at the launch output
13; and sees that one of the node crashes
1; and says he thinks it is because the topic type is incorrect.
13; He says that the mimic node is subscribing to a topic called pose
13; and publishes to cmd_vel on the second turtle.
3; He thinks that if you’re gonna republish a twist
3; why would you subscribe to a pose.
3; He says it could be teleporting.
2; He questions why the subscription type and publish type is asyncronous.
8; He then opens firefox and searches “rosmsg pose” and then clicks on the docs.ros.org official method definition.
13; He sees that it contains a position and a orientation variable.
7; He then goes back to the mimic.cpp file and says that the pose makes sense now
12; becasue there’s a defined thing called pose in our system
2; and now he wants to see how we define it.
7; He looks at the msg type
7; and sees that the linear and angular velocity exist.
11; He then does rostopic info
7; and for the first turtle’s pose
11; and then he does rosmsg show pose
2; saying that he wants to make sure it uses our own pose.
3; He says that since he didn’t do a clean build
3; it could be using a different pose.
10; He then traces through the poseCallBack.
4; He then looks at the logs
2; to see what the mimic node did before crashing.
10; He looks at the launch output to see if it printed anything
12; and it didn’t.
7; He then looks at the mimic.cpp again
7; and scrolls through it and thinks about it.
13; He then highlights the spinOnce command
13; and then says ‘its spinning’.
2; He says he is making sure that it seems like it was calling the callback
12; and that it was subscribed to the actual callback.
M; He then says that there isn’t really anything else that stands out to him.

13; He then highlights spinOnce
1; and then goes ‘thats why’.
3; He then says ‘it will execute once and then close
3; oh and it said finished cleanly (didn’t crash per se)
3; I think this will fix it’.
11; He then makes it again.
11; It launches it
11; and does rosnode list and then rosnode info mimic.
11; He sends the movement command
12; and it works.
N; Post task discussion at 1:00:11.

2.1

N; Coded Storyline and Summary: 2.1SummaryCoded Storyline12314 task begins by the isntructor explaining intent of task
N; and what successfully completing task would look like.
11; 12437 the participant begins by trying to build
11; launch the task
13; and notices that no tracks are returned when he executes the script.
4; He then tries to find the launch scripts
4; and traverses through the directory structure to do so.
7; He opens up the LISP example
H; so I explain that this is irrelevant to our task because we’re interested in the CPP example.
7; He then opens and reads through the service message definition.
7; srv file.
7; He then opens up the mp3_controller code
7; because this is the one that the whole thing is started with 12816.
7; He starts reading through it.
2; He then looks for files that the controller file is referring to
2; specifically something like “mp3inventory interaction” as referred to in the controller code.
7; He opens mp3_inventory.cpp and scrolls through it
7; just reading the code.
M; He says he always skims through the code whenever he first opens it.
M; He explains that he always skims through files that are referred to
M; before looking deeper in files that he thinks may impact or cause the bug.
4; He then says he’s going to look for a file that contains the strings we’re looking for (referring to printed strings?).
P; He says that this is the first time hes seen a program like this.
7; He then looks through the mp3_controller.cpp file again.
11; He then runs the example code again

2; to find the strings that are printed.
13; He analyzes the outputted strings to note that it doesn't enter the for loop in the controller file
13; and deduces that this means it must not be getting a response.
3; He notices that it starts to malfunction when it hits ROS_INFO ALBUMS
13; noting that there isn't an actual error but that its just not recieving the response from the service call.
4; He then looks for the 'server' for the service
3; because he is pretty sure that controller is the client
3; and that it calls the service
3; so he is pretty sure that mp3_inventory is the server.
7; He opens up mp3_inventory
13; noting that it looks for a library path
7; but then looks through it more generally.
7; He then looks through the method definition for mp3_inventory_interaction
7; and then looks at str_directoryKey and str_directoryValue
13; then he looks at the param::search command and assumes that this is what finds the file
13; inferring from the if statement is true 'if it finds the file'.
H; He asks about strcmp
H; and I explain that this is just a string comparator function
H; comparing if two strings are the same in the second if statement.
7; He then scrolls through the file some more.
H; He acts about struct
H; and I explain what a struct is.
2; He says he's now looking why the service isn't returning anything
7; but particularly at the callback
2; saying that it should be printing out the albums.
2; He then looks for what DIR is
H; and I tell him that it's fine.
4; He then explains that he is guessing that it is not accessing the right file
7; looking at /library_path.
7; He then repeats that he's looking at /library_path.
2; He then asks if "/library_path is representing an actual folder
2; or from c?"
H; and I decline to answer.
13; He says that he asked this because it [inventory] know where the files are.
3; He says that the problem starts on the line right after the for loop when returning the list of strings.
H; He asks what "->" does
H; and since we're not testing C++ comp I explain
H; and tell him that the problem doesn't involve the use of this operator.
10, 7; He then clicks around between the terminal output and the inventory.cpp file.
2; He then says he is considering printing something to see what is returned

P; but says that he doesn't know cpp very well.
M; He notices that his lack of understanding of C++ is impeding his understanding of the code.
7; He then looks at the mp3 controller.cpp file
7; and compares this to the mp3_inventory file
7; comparing specific lines.
7; He says he's looking at the mp3 inventory file.
7; He then reads through specific lines of the mp3_controller file
7; the if statements and the for loop.
13; He explains that inventory doesn't return any strings and he knows this because otherwise the for loop would execute.
3; He says that he thinks there is a problem between the communication between mp3_inventory and mp3_controller
3; controller which requests things from inventory.
M; He thinks this because of intent of study.
13; 15819 He says that the call is working fine
13; because it is returning the response string to controller.cpp.
2; He says that he is trying to figure out the way in which contorller and inventory are communicating.
P; He says that he would usually print things out and go 'step by step'
8; and decides to google how to print things in C.
M; The test Surface computer then has internet problems
M; and I fix them.
8; He then googles how to print things in c to the terminal.
P; User starts to give timeline at 21033He says that form experience that when you start a node or package with rosrun
P; you have to go to /src.
13; He knows that there must be a service client and a service controller.
13; He says that he knew from controller that there must be a server.

2.2

N; Coded Storyline and Summary: 2.2SummaryCoded StorylineTask 2 begins at 11927.
P; The pariticipantsays that he doesn't often handle code directly
M; explaining that he is rusty on the day to day things.
11; He runs the build command.
11; He runs roslaunch media.launch.
11; He then runs the rosrun ros_service_examples command
11; which we will call the searcher command.
4; He tries to find the directory it is searching to see if the output could be correct
12; but sees that it is incorrect and thus it doesn't work.
4; Now he knows he must find the source code which implements the MP3 controller service.

3; He thinks the name of the node will be somewhere in the media.launch file he says he's looking for the mp3 launch file.
7; He says he is still looking for the launch file
7; searching through the directories in the terminal.
11; He runs the roslaunch command to see what it can do.
11, 4; He then uses the terminal find command to find 'media.launch' so he can investigate it.
12; He finds it in the root directory.
7; He opens the media.launch file
13; and sees that there is one node and that it gets one parameter named library_path.
2; He says he is not sure what the type of the node is.
2; He says that he's gonna look for something named 'searcher'
2; because he saw this word in the launch file.
2; He is not sure what file extension such a file would have.
7; He opens up the package.xml file
7; but doesn't see anything useful in there.
7; He opens up cmakelists
7; and doesn't see searcher in there either.
2; He says that maybe it is called mp3inventory instead of searcher
7; so he opens mp3_inventory.cpp
7; because it is mentioned in the launch file.
M; He explaims that it is really long.
13; He mentions that there are more than one nodes
13; and that there is another think called mp3 controller
3, 8; which is calling maybe supposed to output things (looking at instructions).
4; Then he says he is looking for mp3controller
N; But
10, 11; he runs it again to see what strings are outputted in the terminal when the launch sequence is ran.
2; He says he doesn't know in which directory it is asking for the stuff
13; and he mentions that he knows that there is a library path parameter in the roslaunch file.
4; He says he wants to edit the roslaunch file
7; and he opens this file.
3; He says if 'this is the parameter it is using
3; the library path
3; then it seems to have the correct value'.
2; But he says he doesn't know if it is indeed using the correct paraemter.
2; He says he wants to check what parameter it is using in the mp3controller.
3; He explains that there must be a caller and a reciever of the service
3; and that clearly the mp3 controller is the caller and that mp3inventory must be the reciever.
M; He says that he is pretty sure there isn't a service visualizer.
2; 13125 He says that now that we know this architecture
2; he wants to know the service message structure of the call.
4; He looks in the srv folder

7; and sees the message definition
3; where the response string is presumably the list of albums.
13; He already sees some apparent duplication because there is a parameter here and also a similar sounding parameter in the roslaunch file ‘library-path’.
7; He opens media.launch
3; and looks at ‘library path’ which he thinks also has to do with where the albums are
3; representing possible duplication or discrepancy.
2; He says that he doesn’t know which one is being used
2; and that this information is probably in the package.xml file
7; so he opens this.
12; He decides that this is not helpful and that it only contains ‘external dependencies’.
7; He opens CMakeLists briefly.
4; He says he is now looking for a cpp source file.
12; He navigates through the directories and finds mp3_inventory.cpp again.
2; He says ‘ok lets read and see whats its sending
2; and populating’
7; and then he reads though the file.
12; He then looks at individual lines in this mp3_controller.cpp and finds out that it is not a single request
12; but many requests to mp3_inventory
3; and he guesses that maybe some of these are getting through and some are not.
13; He then sees from the terminal output that it is sending 0 albums from one terminal
13; and that the requester terminal also is not recieving and albums.
3; He says that he thinks that either the controller is aksing for albums in the wrong location
3; that problems are in the caller side (controller).
M; [this is not the case].
3; 13737 He says that because there is agreement in the fact that it is sending back 0 albums
3; the mp3-controller is looking for wrong things
12; because inventory does have the code which traverses down the directory.
13; He says clearly it is not getting any albums
13; so there are no more consecutive calls to get the titles since no albums are found in the first place.
M; So now he says something different
3; maybe its asking for albums in the wrong directory
3; and just getting no albums.
3; He says more than anything
2; he is intersted in where the name of the directory is coming from
2; and that tehre seems to be something in the roslaunch file
2; and that the directory in the roslaunch file is not empty.
11; He runs rosservice list and sees mp3inventoryinteraction.
2; He says he doesn’t know what he can find out about services.
11; He runs rosservice info mp3inventoryinteraction
13; and sees that searcher is just a custom name

M; and says he wishes that there was a think that logged the calls that were made to a service in the last hour
M; and these calls were made with these arguments.
6; He says he doesn't know how to see a log of service calls so he will put in debug prints.
7; He goes back to mp3_controller.
7; He looks specifically at the line making an object of the service
7; and then send.
3; He says that there seems to be no population of the other part of the srv message definiton
7; so he opens up the service definition
7; to cross reference the files.
13; He says 'weve populated the request string
13; but we haven't populated the album
13; so maybe its an optional argument'.
12; He then figures out that this album field only need to be populated if you want the titles of a paricualr album
13; and then realzes that he doesn't think there is an error in mp3controller and will unfortunately have to wade into the mp3-inventory
2; but that for now he is trying to make sense of the protocol between the two nodes.
13; He says that he know the only thing he is sending is the album list
2; but he isn't completely sure tha tthis is a lgit request
13; but that we're clearly not sending the directory where it is located
13; but 'thankfully we have the paramater in roslaunch'.
7; He opens up mp3inventory
7; and begins reading the main method
7; and then goes to the top to read the service handler code.
M; He says 'before you fix a system
M; you have to understand what a system is' and 'how they interact'.
13; So he notices that the mp3inventory parses the request string
13; and the request string is the album list
13; then it starts doing something.
2; He wants to know where it access the parameter
7; and he finds the ros::param:search call
13; which he says it shows that it access the library path parameter from the roslaunch file and that it stores it
2; but says he doesn't know the semantics of this command.
6; He says he will place debug printing in
6; and does so with "ROS_INFO" prints
2; and he says he is doing this to see if the library path parameter was found so he makes an else statement for this case
M; and repeats the mantra 'crash early
M; crash often'.
11; He then builds the project again and executes the lauch command.
10; He notices that he messed up the string format

10; but this was warnings only.
10; He notices that his print command displays ‘no albums found’
13; showing that the library path parameter was not found.
11; He builds it again
8; and he googles how to print with ROS_INFO.
11; He builds it again and it works.
10; He sees that it again prints ‘no albums found’
12; telling him that the param::search didn’t find a parameter called ‘librarypath’ in the parameter directory
7; so he opens the launch file to look at the parameter.
3; He says maybe this time it is a naming issue
3; because it doesn’t have a slash in front of it
3; and this one does
6; so he tries removing it and then building it again to see if it will work.
11; He runs it again
11; and it works.
N; 15454 We then discuss concluding questions
N; recorded in the google form.

2.3

N; Coded Storyline and Summary: 2.3SummaryCoded StorylineTask 2 begins at 1:16:00 or so.
11; He makes the project.
11; He then does rosnode list.
11; Looks around catkin workspace.
11; Launches the project
11; “roslaunch rosmedia.launch” and “rosrun ros_service_examples…”.
13; He notices that he doesn’t get any albums back
3; and that “maybe it didn’t get any so it just shut down”
3; and says maybe that “controller” is working correctly but “inventory” is not sending back any.
3; He says things media.launch is a service
3; and rosrun ros_service_examples calls it.
13; He notes that it only sends back 0 albums.
7; He says that he’s going to look in media.launch.
7; He says from the launch file
13; it takes a param ‘library’ path
12; and determines that the path is valid.
7; He opens up the srv file
13; and learns the structure of the service.
11; He looks around the directory structure of the program.
7; He opens up the mp3_controller.mp3 file

3; and he thinks it has the main()
12; and it does.
3; He says that he thinks mp3_controller is the ‘server’
3; but then expresses uncertainty.
3; He then opens up mp3_inventory and says that he thinks this is the service.
7; He says that he learned this from the names in the media.launch.
3; He says that if he calls ‘roslaunch’ to start something
3; then the launch file will be a first start.
2; He then ‘checks if theyres any namespaces that may affect the parameter search
P; saying that he’s seen this before where if something’s in the wrong namespace the parameter search may fail’.
7; He’s just looking through the mp3-inventory file.
2; He says he wants to check the value of the library path param search
2; in order to make sure that it actually gets populated.
2; He says he wants to find out the contents of library-path.
13; He didn’t realize that there was album-list and title-list
2; and was trying to figure out what they each were.
6; He writes code to print out the directory that it thinks its in.
M; He thinks he may be too focused on ROS
M; and that he may be overthinking things.
11; He runs the system
13; and he notices that the strDirectoryVal.c is not printed.
P; He says that he usually uses Nodes
P; not Services.
6, 2; He then puts a print statement in the if(serachparam()) to see if it gets past the search param.
11; He runs it
12; and sees that it doesn’t get there.
8; He starts up firefox
2; and says ‘time to see how to pass something to a service
2; so he’s assuming that the paramsearch failed or didn’t find the library path.
2; He expresses uncertainty as to how the param search should work
2; with the key and the value and such.
8; He opens up the ros ‘parameter server’ wiki page.
12; He says that the wiki page tells him that it(?) returns false if it doesn’t exist.
6; He then puts in an if(hasParam) call
2; to see if it even has /library_path.
M; He then explains that he thinks vim is better than vi
M; which he is currently using
M; and why they are both better than gui editors.
2; He then says that he’s wondering if /library_path is a global or relative variable
3; which he says he could just test by removing the slash
M; but doesn’t do this.

6, 2; He puts in the hasParam if statement iwth a ROS_ERROR to print if it has the library path
6; and does an analagous one for library_path with out the slash
N; for two total if and print statements.
2; He has problems because the NodeHandle isn't declared
2; and it was nh.hasParam().
11; He fixes this and then builds the project
12; and neither print statement will print.
6; He moves it to below the service starter line
3; and thinks that maybe it has to be after.
11; This doesn't work either
6; so he tries hardcoding the directory path into the code.
11; He then tries running it
M; and gets some more output than he used to
13; but no list of titles.
6; He puts in more if statements.
2; He then wonders if he could use rosservice.
2; He wants a way to get it to tell him about rosservices as they happen.
6; He decides to try deleting the leading slash
11; builds it and it works.
N; Post task interview begins at about 1:59:00

2.4

8; Coded Storyline and Summary: 2.4SummaryCoded Storyline the participant reads through the commands
A; and opens a terminal
11; makes the project
11; launches the service and runs the searcher node.
2; She notices that she is thinking 'what is it talking about
2; with album list'
3; and that there should be a hardcoded album list somewhere with 'directory information'.
3; She says obviously its wrong
2; and wants to find where the directory is coded.
3; She thinks the first one is running the ros master
3; and the second one is running the service.
4; She wants to look for the source and the launch file.
A; She installs vim.
7; She opens the launch file
M; but says she doesn't know what she wants to find in it.
13; She notices that there is a parameter called library path
13; which has the path as the value
4; and then she navigates to the specified location in the file explorer.

4; She says she would like to look at the source
4; to see what the rosserviceexample is.
4; She says she would like to see what it is doing and why its not working.
4; She looks around
4; trying to find where it is.
4; She opens up the.
7; srv file
7; to see what is in it
13; and says that this is the type of argument that the service recieves and sends out.
M; She says this will help her when she's looking at the source.
7; She opens up mp3controller.cpp
2; and is looking for the service and how it works.
7; She looks through
7; notices a node
7; and a service client.
2; She says she isn't quite sure what it does
3; but that she guesses that this file calls the service because she sees a service client.
3; She sees that it is perhaps calling the service perhaps the 'album_list' argument.
13; She says that there is a service of type MP3INVENTORYSERVICE
13; which is the other file in the directory
13; and that it is passing the albu list as a parameter.
13; She then says it loops around the service response
13; and that it then prints them.
3; She says she thinks that the problem is in the service itself
3; because the mp3controller doesn't seem strange
3; and that it seems to follow a logical and understandable process.
7; She opens up mp3inventory
12; she notices the library_path parameter and thinks this is the parameter that we saw from the launch file.
7; She scrolls through the file
13; and notices that it opens a directory and reads through the files which are in it.
13; She says that it's reading through the folder and putting the names into an array of strings.
L; For the four files she's seen
L; she says one is the launch file which launches the rosmaster
L; another is the one which is calling the service
L; and one is the one providing the service.
3; She says she thinks that the mp3inventory file
3; the provider is likey to have the fault
P; because she thought the client was not 'feeling strange'.
3; She says the mp3controller felt simpler
3; and that mp3inventory feels like it has more things can go wrong.
10; She says that right now
10; there are a lot of log messages that she did not notice printing.

10; She looks at the original terminal output to see which logs ended up appearing.
2; She then is trying to figure out why one of the log messages didn't print int he mp3inventory file.
12; She says it didn't write anything from the server
12; she's only seeing messages from the client
3; so maybe the client is not calling the service right.
7; She looks through the mp3controller file
7, 10; comparing the printed statements to what was actually printed
2; in order to see how far into the mp3controller file it got
12; and notices that because it didn't print any albums
12; the albums parameter is empty.
2; She says she wants to know how long the returned list is
2; and if it is indeed zero.
6; Makes a rosinfo print statement to show this.
6; She makes it
11; and then runs it
12; and sees that it prints that the size of the list is 0
3; which leads her to believe that there's something wrong with the service provider
3; because it doesn't return anything.
7; She goes back to mp3inventory to see what is wrong.
11; She runs everything
12; and notices that the service provider is returning a list with 0 elements
M; which is wrong
3; meaning that the error might be in the mp3inventory 'server' side.
13; She sees that the request string is album list
3; so she thinks there is a problem in the while loop.
6; She puts a rosinfo print in the loop
6; printing the file name opened.
11; She builds it and then runs it
12; and notices that its not going into the loop
3; meaning that there's something wrong with the loop.
3; She says either its not able to open the directory
3; or the while loop defintion is wrong.
3; She says she will change some things around to see what value is assigned to dp
3; and see if it's assigned to null
3; because she thinks there may be something wrong with the permissions or something.
P; She's trying to remember how to print out a directory pointer
P; what format to use.
3; She's guessing that it's not able to open the directory or see the files.
11; She tries building it
11; and it fails.
1; She notices that there's something wrong with the parenthesies
6; she fixes it and it fails again.

6; She adds a semicolon
11; and it works.
G; She wants to print out a boolean
G; but she doesn't know how to do it.
11; She builds it again
11; runs it
12; and says that it was not able to open the directory because the directory string doesn't contain anything.
2; She says now she has to look where it's filling it
2; the string.
2; She says her strategy is to see where the string is getting filled
2; and to try and see what is wrong with that.
12; She notices that it gets it's value from the rosparam::search call.
2; She says she doesn't know what is happening here.
2; She says she's gonna try to see what the rosparamsearch and rosparamget do.
8; She searches on google
8; and opens up the wiki page for the parameter server page.
8; She reads through the listing for the get() call.
2; She says she's trying to understand what the two functions are doing.
8; She reads through the accompanying text for the rosparam:search() example code.
8; She reads about namespaces
12; and notices that the example uses relative namespaces but the task uses global namespaces.
13; She notices that the search will find it with the 'closest' namespace
M; but that she isn't quite sure what this means.
2; She says she still isn't sure what is happening with keys and values.
6; She takes away the slash
11; builds it
12; and notices that it still isn't working.
M; She says she's confused.
7; She says she will look at the launch file and see if it even could have caused the problem.
13; She notices that she didn't build the project using the correct method from the instructions.
11; She runs it using the instructions and it works.
N; Post questions at 38.18

2.7

N; Coded Storyline and Summary: task#.participant#SummaryCoded StorylineBegins at 40:43.
13; He notices that there is a ros service in here.
13; He notices that it sends back 0 albums
13; and that there are no albums
13; and that it prints 0 albums just like it said it would.

G; He says he's going to look at the server
G; the thing collecting the track names.
7; He looks at the launch file.
12; He says that he learned the file he wants to look at
13; and also sees a parameter named library path
M; and he doesn't like the way library path is set because you have to know exactly where things are
M; and the path can easily get screwed up.
7; He looks at the slash in the launch file
3; and says it is probably correct because there 'isn't a namespace'.
3; He remarks that often ROS namespacing is hard to remember.
7; He looks through the mp3_inventory cpp file but it is very hard to tell what he's talking about without seeing it.
3; He says he thinks there may be a bug in the way it handles directories.
M; The hour is up
M; but he says from here
G; he would have checked to see what mp3controller was doing
G; making sure its calling the services correctly
G; then would add many print statements around the files.
N; Post task questions at about 50:00

2.8

N; Coded Storyline and Summary: 2.8SummaryCoded StorylineBegins at 6:05.
11; Navigates to catkin workspace
11; then makes the project
11; and runs the server.
A; He then opens a new terminal
11; and runs the client node using rosrun.
8; He says that from the command we send them
3; it is looking for the album list
3; but it doesn't return any albums.
H; He's not really sure what it's supposed to do
H; so I explain the instructions and desired output in more detail.
4; He says he's going to go to the source code
7; going into mp3controller.cpp
7; and he knew to go there from the rosrun … mp3_controller command.
7; He scrolls through it briefly
7; and then begins reading through the code.
7; He says he's noticed that the service takes in a string value
3; and if it matches album_list it goes into the if statement (this is incorrect
M; if it returns true it goes into the top level if).

3; He says that the first if statement is working because it prints things after that.
3; He says that the second one is working
3; because prints after that too
3; “Albums”.
7; He traces the if statement in mp3_controller
8; and googles the string compare function to remember how it works
12; and deduces that it returns 0 if they’re equal.
7; He looks at the for loop after that
2; he’s trying to see why it doesn’t execute anything in the if statement inside of the for loop.
8; He googles to see what string streams are
8; and finds a cplusplus.com website.
P; He says that he usually uses C++ with ROS.
P; He says that he uses ROS with Point Cloud Library image processing libraries.
2; He says that he’s not sure what happens with “sts << “album_list”” is executed.
M; He say he still doesn’t see what he’s looking for
2; it seems like he’s confused as to what the “<<“ operator is doing.
2; He wants to know why we later set sts as blank when we previously had set it as “title_list”.
8; He googles “using the << symbol for std::stringstream”
G; saying that he wants to understand it in order to understand the control flow of the program.
8; He changes his search term to “std:stringstream usage”
8; and opens a cpluscplus.com/reference website
8; and a stack overflow site.
8; He looks at the stack overflow site.
8; From the cplusplus site
12; he says ‘its a way of adding
12; adding data in to the string’
12; and says that this answers his question ‘somewhat’.
7; He then looks at the code again
2; and says he is still confused what “sts.str(“”)” is doing
3; saying that it may be clearing or refreshing the string.
7; He then looks further on in the code.
8; He then googles c_str
8; and clicks on the clplusplus reference for string::c_str
8; and reads through it.
1; He says that the error is that it is not completing what it’s intended to do
3; so maybe its not a problem in the code but the way it’s supposed to communicate with the other parts.
G; He says he’s mainly trying to understand the source file first.
2; He looks through the directory tree to the places where the dummy mp3 files are stored.
7; He also opens the cmakelists
13; and says it looks fine and also says that it compiles correctly so he thinks he can eliminate compilation errors.
7; He opens up mp3_inventory

7; and scrolls through it.
3; He says he thinks the mp3-controller calls the mp3-inventory
3; and the mp3-inventory file is responsible for finding and returning the files
3; so he thinks there is a broken link in the mp3_inventory file.
M; He says that this file has helped him understand the system a bit better.
7; He scans through the code
7; tracing along lines with his cursor.
7; He says that comparing the temporary file name with the period character
7; and he doesn't know why.
3; He thinks it finds the correct place to look
3; because there is an else statement which says "faulty request string"
3; and that isn't coming back so he thinks it finds the correct directory.
A; He opens up the file explorer
7; and looks at the mp3inventoryservice.srv file
A; then closes it.
M; He says he is not sure what is happening right now.
M; He says he would normally bang his head against the screen.
7; He says he's just scrolling down the mp3_inventory file.
7; He wants to look at the mp3_controller again
3; because the way he sees it
3; the mp3_controller is the upper layer
M; and he wants to go from the top layer inwards.
8; He looks copies "request.request string" from the file and googles it.
3; He says that in the mp3_controller file
3; we are requesting for the album list
2; and he's trying to verrify that what he thinks is happening is actually happening.
M; He says that he has no idea why its not working at the moment.
2; He's trying to figure out how and why all the if statement comparisons are making in mp3_controller.
3; He thinks that one of the if statements or the for loop is the source of the error.
2; He says he is trying to understand where srv.response.list_strings comes from
2; to see if it is running correctly in the for loop or not.
7; He then scrolls through mp3_inventory.cpp.
G; He is looking for place in mp3_inventory where srv.response.list_strings pops up.
2; He searches within mp3_inventory for the string "list_strings" to see where it is defined.
3; Currently he thinks the error is either in the for loop in mp3_controller or beyond
3; but thinks the error is in the mp3_controller file.
M; He says he's still 'clueless'.
3; He thinks that the if statement "if(client.call(srv_titles))"
M; but isn't really sure.
3; He thinks that it is in the if statement
3; because otherwise one of the else statements should have executed.
M; He says that he's about 70% sure.

M; He says that he doesn’t really have a strategy for trying to fix the thinks.
3; He thinks the error is in the area of the inner most for loop
M; but he thinks he’ll have to look over the entire code again to see what things can affect this area of the code.
7; He looks over many of the conditional statements in the mp3_controller file.
3; He thinks that it is getting stuck in the for loop.
2; He is still wondering why the else statements do not get executed.
6; He tries commenting out the contents of the outer for loop.
2; He wants to test using his own string for something (not sure what) in the loop.
G; He wants to print the whole list
3; but then realizes that if nothing is printing in the for loop
3; then nothing would print if he adds something else.
6; He adds his own print statement
11; then builds the project again and runs it.
10; There is a compile time error to do with mismatched brackets
13; and he realized that he commented out too many brackets.
6; He fixes this
11; and then remakes the file
11; and there is another error
H; and I tell him that he forgot a comma where he is printing something.
11; He builds it again and it works.
11; He then tries running the system using the normal launch process.
10; It prints an x
10; but not the srv.response.list_strings().
7; He scrolls through the mp3_controller.cpp file.
3; He says that there should be somewhere where the service is getting the information from
2; so he’s trying to find that.
7; He’s now going through the start of the mp_controller file
2; so see where the service is created.
7; He opens mp3_inventory and looks around.
3; He now says that the error must be in the mp_inventory file because that is where the service is getting all the data and such.
3; He thinks that the list is empty.
7, 2; He searches for occurrences of “list_st” in the inventory file to see places in which the list string is referenced.
3; He then highlights the code block in the second if statement and says that he thinks that this is where the error is
2; but that he is not sure where the error is.
'B; He chose that block because that block is where things are added to the strings
2; he thinks nothing is being added ot the string.Post questions at 1:03:14

2.11

11; Coded Storyline and Summary: 2.11SummaryCoded StorylineMP3 TaskStarts at 23:43.Builds task.
10; It builds successfully.
7; He opens the launch file
M; saying that he usually likes to do this becasue it gives him some idea what the package is.
13; He says ‘it seems like there’s some rosparam which tells it which directory contains []”.
11; He does the roslaunch command
11; and then does the He does the rosrun command
13; and he notices that ‘nothing happened’.
4; He says he’s gonna dig around a bit
4; he’s gonna look at the ‘node’ and see what its doing.
7; He opens up the launch file again
4; and form this sees that he wants to open the mp3inventory file.
7; He scrolls through it in gedit.
M; He says he cares how big it is because if its small
M; he’ll read them from start to finish
M; but if they’re big
7; he’ll go to the part which is used more.
M; He says he doesn’t think this should be a super complex task
M; so if its a large code
M; then only some parts are used
3; and there may be some functions which may not be used in this instance.
12; He said the launch file told him which node was running
12; which pointed him to the cpp.
13; He also noticed that we’re passing the location of the directory as a ROSPARAM
2; so he doesn't know why we need an mp3controller node.
4; Scrolls though mp3_inventory
7; then opens mp3_controller.
N; Changes the color scheme to the solarize theme because it is best for the eyes.
7; He chose to open mp3 controller because that was the node he was running with rosrun.
2; He says he doesn’t know where the error is.
7; He scrolls through the controller file.
7; He is comparing the response to the album_list
8; and he googles to find out what strcmp does.
P; He says that he normally writes in CPP.
7; He opens the controller file
7; 10; and begins to compare the print statements to the terminal output to see how far it gets.
10; He says he can’t see any of the print statements printed in output after “Albums:”.
11; He greps for “list_strings”

10; and gets many results.
6; He adds a print statement to print the list_string
3; which he thinks is a vector of strings
2; and he wants to check the size of it
2; because the for loop which is supposed to iterate through it doesn't execute.
11; He makes and runs the project
12; and sees that nothing got added to list_string.
2; He says now he wants to see why nothing was added.
11; He greps for it again
13; and sees that it is modified in mp3_inventory.
7; He opens mp3 inventory
7; and searches for list string.
12; He finds it in at least 4 locations
2; and says he's looking for where its being modified.
7; He Looks at the outer if statement
10; then closes the file and goes back to the terminal.
G; He says he's trying to go through the flow.
7; He traces through the program
7; with the if statements and conditionals.
2; He says that somewhere
2; the album list is not being populated.
P; He says he's not super familar with C directory commands
2; but says that he thinks we're trying to open a folder pointed to by a string.
2; He then asks what is "strDirectoryVal" highlighting the line which searches the rosparam server.
4; He then says 'let me open up the header file
2; you have a rosparam in your launch file
2; right?".
4; He then browses to try and find a header file for mp3_inventory.
L; He doesn't find it.
7; He then opens The cmake lists to see if there was a header file
L; and sees that there wasn't.
M; He says that this is a bit weird.
2; He says he can't find the header file
2; but says it should be somewhere.
7; 2; He opens the launch file and looks at the parameter which is passed.
7; He looks at the cmake lists again and says he wants to find where the header file is
7; but is actually pointing at the message definition (.h) file.
2; He says that in the header file
2; he expects to find the way we've defined the ros param.
6; He puts in a print statement to print the directory value.
11; He remakes it
11; and runs it.

12; It reports that the directory itself is a null string.
P; On what he calls intuition
6; he puts in another print statement.
2; He says he doesn't know why we're defining two strings
2; directoryString and directoryVal.
7; He opens up the MP3InventroyService.h file.
7; He searches within MP3InventoryService.h for 'param'
L; doesn't find it
A; and closes it.
2; He then says that he's gonna see what exactly ros::paramSerch does
8; so he googles it
8; and opens the docs.ros::param namespace reference.
8; He reads through the docs
12; and reads that it serches for a parameter called 'library path'
12; and then assigns the value of library path to DirectoryString
12; and then returns true if it is successful.
2; He then asks what get does
8; and then reads a similar doc for ros::paramGet.
6; He then tries commenting out strdirectoryKey and replaces it with StrDirectoryVal.
8; He then reads the doc again for Get
11; and then builds the project.
11; He runs rosparam list
2; and asks 'why are you giving it a namepsace?'.
11; Hen then does 'rosparam get /'.
10; 13; He then says 'its inside some namespace called searcher'.
7; He then goes to the launch file
13; and notices that the parameter has a namepspace called 'searcher'
13; the name of the node.
6; He puts a print statement inside the search if statement
11; and builds it.
11; He runs it
13; and nothing prints.
12; This means it did not return true.
13; He reads that it must not have found the parameter.
8; He reads the part where it talks about searching up a tree for the parameter.
2; He says he doesn't know what this means.
6; He tries adding 'searcher' to the beginning of the library path rosparam serach path
6; adding the namespace.
M; He says he did that because when he did the rosparam list
M; he saw it written that way.
M; He says he thinks there should be a more elegant way of doing it
M; but at least this way he knows he's on the right track.
11; He makes the project.

6; He then reverts his edits where he switched the directoryKey and directoryVall in the search call.
11; He makes it again
11; and runs it.
12; It works.
M; He says ‘its not a very clean way to make it work
M; because if you change the name in the launch file
M; you’d have to change it in the code’.
M; He says he would have had whatever node is calling the service should pass the name of the node along with it
M; or something
M; and it could be concatenated.
7; He then opens up the automatically generated mp3inventoryservicerequest.h header file.
11; He then finds a rosanswers page ‘programatically get node namepsace’
11; and thinks this may be a more elegant way of doing things.
6; He puts in the namespace getting code into the code.
6; He continues to try and make it more ‘elegant’
3; and says he wants to see if he gets the namespace of the node calling the service and append this to the parameter being searched.

2.12

4; Coded Storyline and Summary: 2.12SummaryCoded StorylineMp3 TaskTask begins at 20:20Navigates to gome directory.
11; Echos the ROS_MASTER_URI.
11; Makes the project
11; it makse successfully.
11; Launches the service node.
11; Does ‘rosnode list’
11; and then does ‘rosnode info searcher’
2; to see if it actually launched.
11; He then does ‘rosservice list’.
2; He says he wants to see what the type of the service he’s gonna be calling is.
11; He then does ‘rosservice info’ on mp3inventoryinteraction.
11; He then does ‘rssrv show ros_service_examples/MP3InventoryService’ which shows the message type.
P; He says he usually does this to learn about the system
P; to see what is connected to what.
12; He knows that the launch file starts a node which starts service which we’re gonna talk to.
13; He knows the type
13; and what it is passed and what is returned based on his most recent call.
11; He then calls the service using rosrun

13; and nothing is returned.
10; He compares the terminal output to the screenshot
3; and he says he thinks its either passing the wrong directory or there's something wrong with the service.
3; He says he either thinks the callee didn't give a proper request string
3; which he thinks might be the directory
3; or he thinks there's something wrong with the recursive directory listing.
3; He thinks the caller has less code
7; so it will be easier to start there.
7; He opens mp3_controller.cpp corresponds to the node of the same name.
7; He looks through it
13; and sees we have a service set up
13; and the request string is just 'album_list'.
7; He wants to look at the service provider and see if this is the request they're expecting.
3; He says that it then calls it
3; and if it comes back ok
6; it prints it
7; it checks to see if the response is 'album_list'.
11; He then does 'rosservice call --help'
11; and then he decides to manually call the service using rosservice call to see if it just returns an empty string or what.
M; He says that doing so will give it the raw service response
M; to see if it is even returning anything.
11; He does so
12; and it doesn't return anything
12; which tells him that 'album_list' isn't something its expecting
12; or there's somethign wrong with the service provider which caues it to return nothing.
7; He then opens mp3_inventory.copp.
7; He highlights "/library_path"
M; and says 'its interesting that you're using the parameter server to get the directory that its supposed to be looking in.
3; It could also be the parameter server'.
7; He says that it is looking through the parameter server for something called library path for something
3; and if the parameter is set to nothing
3; it could be looking in the wrong directory.
P; He says he's used the parameter server before.
11; He then does 'rosparam get /searcher/library_path"
12; and it returns the valid and correct directory path.
3, P; He then pauses and says 'except I don't think … I can't remember if it searches … I think the fact that there's a beginning slash means that instead of looking under this node's namespace' its gonna look under the global namespace and this parameter doesn't exist.
6; He then puts in a ROS_INFO print statement to print something

N; then pauses.
3; He says he thinks its trying to look in a directory that is just an empty string but he can't remember the exact api for the parameter search.
8; He then opens google and says that he thought it was going to check that he understands how the rosparameter server works.
8; He then opens up wiki.ros.org/rosparam
8; then closes this.
8; He then opens roscpp/Overview Parameter Server
8; on the official wiki.
8; He then looks at the section called "searching for parameter keys'.
M; And then he finds the directive
12; and says 'yeah so the forward slash means searching globally.
6; He then continues to write the 'ROS_INFO' print statement.
6; He then makes a simiar print statement for strDirectoryVal.
3; He says the leading slash is making it look for a parameter in the global namespace
3; and he thinks that if he removes the slash it will work
2; but he wants to see what its returning first in his print statements.
11; He makes the project again
11; and it is successful.
11; He then runs it
11; and then calls the service.
10; It prints an empty list string
10; and empty vals and keys.
6; He then deletes the slash
11; and tries it again
10; and it returns albums when he directly calls it.
11; He then tries the official command and it works.
N; Post task discussion begins at 35:58

3.0

11; Coded Storyline and Summary: 3.0SummaryCoded StorylineBegins running commands at (6:31)
10; notices first error in terminal at (6:45).
2; First thought is "to find out the file where the errors are coming from".
7; Opens turtle.cpp.
7; Goes to line 66
M; which is the method signature I modified.
10; Says that in the callback function
10; 'Velocity' is not defined
3; which "means that there must be some problem with the message".
10; Currently only examining the first part of the terminal error message.

2; Asks the question to himself: “What should the fields of the message be?”.
M; This suggests to me that instead of challenging the usage of a Velocity message type or thinking about using a different message type
M; he believes that there is some problem with the way that Velocity is defined.
11; Lists the msg definition files in the turtlesim package
13; which does not include Velocity.
11; Tries to open “Velocity.msg”
10; which doesn’t exist
10; thus creating it.
8; Googles “ros point”.
8; Opens second result
8; “geometry_msgs/PointDocumentation”
8; bringing up the message definition (not a tutorial) for the Point Messgage type.
12; Says that the turtle.cpp file is reading the linear and angular values
N; and because of this
3; he believes that “most likely this is going to be a Point message”.
P; I got the impression that he has seen similar data types used in Point messages before.
6; Begins editing the new Velocity.msg file to use the Point message type.
1; He says that he will have to modify the CMakeLists file to include his new Velocity.msg file.
1; He says that he is doing all this because the terminal error says that the Velocity message type does not exist.
7; Opens the CMakeLists file.
8; Googles “ros publisher subscriber”
8; (13:26) and clicks on the official tutorial entitled “writing a simple publisher subscriber
8; C++”.
8; He goes back
8; and searches “ros message” instead
8; choosing the result entitled “Creating a ROS msg and srv” (13:50).
6; Adds his new Velocity.msg to the CMakeLists file.
11; Tries catkin_make again.
N; (15:30)
10; and the build fails on the turtle.cpp file with the original error
10; and user recognizes this and does not yet have a new idea in mind (16:43).
8; Googles “ros publisher subscriber”
8; finds “Writing a Simple Publisher and Subscriber” page.
4; Searches the beginner_tutorials package for “Velocity.h”
12; finds that it doesn’t exist.
4; Searches the whole src folder for the same
4; still no luck.
C; He realizes that because that command should have found the header files
C; and that it did not find them.
4; Looks for any occurrence of “Velocity” in the whole src folder
12; and the only result is the file Velocity.msg which he created earlier.

4; Performs a similar search for “Color.msg”
4; a ready made file
12; and finds it in the same location.
N; (19:40).
8; Searches Google for ros velocity message
8; clicks on “Understanding topics”.
8; Searches on the page for “velocity”
2; and says he is hoping to find a velocity message (definition?) on the page.
8; He is now looking directly at the place where it points out that ROS Groovy and earlier use a custom Velocity type whereas Hydro and later use a standardized Twist type
8; and remains on this section for some time.
C; He says that the turtle.cpp file doesn’t make sense because the tutorial says that the Velocity message type uses float 32 (21:40)
6; and then points with his cursor at the assignments using vel->linear.x and such in the velocityCallback method.
10; He reiterates that the error message says that “Velocity does not name a type”.
10; Searches the whole ros/kinetic folder for occurrences of “Velocity”
10; and finds 8 results.
10; From the results
10; he opens the include/turtle_actionlib/Velocity.h file.
6; He looks at it briefly and then offhandedly states “I’ll just include it” and then includes it in the turtle.cpp file.
11; Tries building the project again
11; with the include statement in turtle.cpp.
10; This fails in the same place as the first two times.
7; Opens “Velocity.h” file.
M; I ask him about the high level nature of the error
1; and he says “It says that the type is not defined
1; and I am trying to define the type”.
M; This means to me that he is still thinking that the Velocity message type is necessary to the function of the program.
N; (27:54).
M; At this point
M; The professor asks to speak to the instructor outside.
6; While they are outside:Subject tries putting turtle_actionlib::Velocity:: … as the method signature
6; instead of Turtle::Velocity.
11; He then tries building the project
10; and it fails on compiling on the same line again
10; but this time saying that the header (.h) file needs a Twist message.
N; The professor and the instructor come back in.
H; The instructor explains to subject that Velocity is not the correct message type
H; and suggests looking around to find a different message type.

6; (29:46) Subject explains that he tried changing the method signature and that on build
7; he has now seen that the method definition in the header file uses a Twist message type.
6; He then tries including and using a twist message
11; 10; and finds that it builds successfully.
11; He tries running it and publishing to it
10; and this succeeds as well.
N; Task complete at (32:12).

3.6

G; Coded Storyline and Summary: task#.participant#SummaryCoded StorylineHe says he first wants to find the catkin workspace
A; and does so using the GUI.
13; He notices that there are only two packages
13; beginner_tutorials and turtlesim.
7; He opens up the cmakelists
7; and says it seems good.
7; He opens the multisim.lauch (wrong) launch file
13; and sees that it just has two turtle nodes.
13; He says that the launch file starts two turtles nodes.
4; He opens up the src directory
7; and opens up turtle.cpp.
7; He scrolls through it quickly and closes it.
2; G; He says he's trying to understand the whole architecture of the code.
M; He says that he thinks it will be pretty simple because of the launch file.
G; He says he's going to now try starting the task
11; and opens a terminal
11; does catkin make
10; and it fails.
10; He notices that there is an error
4; he says that he will start with turtle.cpp
10; and noting the error message.
12; He says that the error says "velocity in class … doesn't name a type"
8; and he decides to google it
2; to see if Velocity is a class under the Turtle namespace.
8; He opens a web browser.
P; He says that if he remembers correctly
P; it should be "Twist" instead of "Velocity".
P; He says that he has done a similar thing before
P; because he learned from the ROS tutorial which uses turtle.
P; He says he has not done mimic.
8; He opens up the turtlesim ROS wiki page.

N; He says the internet is slow
N; and it takes a while to load the page.
2; He says that he's trying to figure out which classes are under the turtle namespace.
N; The page doesn't load
8; so he googles turtle::velocity and googles turtle::twist in a new tab.
N; No pages load
N; so he closes the web browser and re opens it.
8; He then googles "ros turtle namespace"
2; and says that he wants to know what classes are under the namespace
3; because he thinks the error has to do with Velocity not being one of the class names.
8; He opens a third party tutorial 'nootrix.com/software/ros-namespaces' but then goes back.
8; He then clicks on a 'docs.ros.org' link named "turltesim namespace reference' for the diamond back version.
8; He looks through
13; and sees that Velocity exists
8; and scrolls through its methods.
8; He clicks on the velocity documentation from this page
8; and scrolls though it.
2; He says that he tried to find out the correct name from the web page.
6; In turtle.cpp he tries changing "Turtle::Velocity" to "turtlesim::Velocity" (in the callback header) as he sees in the diamondback docs.
2; He says that he is trying to modify the name
4; because he thinks the error may be because of a typo.
11; He then tries building the project again
10; and it fails in the same place
10; this time saying 'turtlesim does not name a type'.
6; He modified Turtle to turtlesim because he found it documented in a web page.
8; He scrolls through the web page some more.
M; He asks if there is an IDE
M; and I tell him that there isn't but that he can install packages if he wants.
G; He tries to download and install QT
3; saying it is his favorite.
3; He says that on the velocity reference web page it tells him that he needs to include the header file velocity.h
3; but he's not sure if that is the case
G; and if he installs an IDE
G; it will automatically find error for him
G; and be able to control+click thinks to see if they are valid.
A; He downloads QT.
1; He switches to the terminal
1; and checks the ROS package path with echo $ROS_PACKAGE_PATH
10; and says that what is returned looks good.
2; He says that he wanted to do that

2; because if you're trying to find a package
2; you must first find this workspace
2; (explaining why he wanted to know the package path).
6; He tries including the velocity.h file in the turtle.cpp file and then tries making the project again.
10; 11; It fails with the error "Velocity.h no such file or directory".
6; He then deletes his include statement for velocity.h.
11; He tries building it again
10; and gets the original error "Velocity … does not name a type".
3; He says that right now the Velocity.h header file is missing
3; and that this is what he thinks the error currently is indicating.
6; He searches for more references to 'velocityCallback' in the turtle.cpp file.
8; He then launches the web browser again.
8; He then googles 'dos tutorial' and clicks the first 'ros tutorials' wiki page link.
8; He clicks on the tutorial 'understanding nodes'
8; and then clicks on the 'Writing a publisher subscriber node'.
10; He says that there are four errors reported in the terminal for the velocityCallback function
3; so another problem is that the callback is in the 'wrong format'.
8; He says he wants to use the ROS tutorials to find out what the correct format is to write a publisher or subscriber.
8; He scrolls through the publisher subscriber tutorial
77; and references thinks against the turtle.cpp class.
7; He is looking at the portion of the code where it registers the publishers and subscribers.
7; He says that our code is a bit different from the tutorial
7; in that our turtle.cpp code has a namespace with a method as its callback.
2; He says that he wants to make sure the format of the callback is correct.
8; He googles and finds an official wiki page called 'using class methods as callbacks'.
8; He scrolls though it.
2; He is now looking for the definition for the class turtle.
8; He googles 'install.
8; run file"
G; as he is now trying to install QT.
10; He does this in the terminal.
11; He does many thinks for this
M; I will not transcribe much while he is trying to install QT.
G; He says that QT will help him get a whole picture of the code structure
G; because this is someone else's code
G; not his code which he wrote from scratch.
P; He says that he uses QT when developing for ROS.
7; In the mean time
7; he looks at the turtlesim.cpp file
7; just to have a look at other files.
7; He sees that there isn't much in the turtlesim.cpp file

13; and says that it wasn't very helpful.
3; He thinks that the error tells him that Velocity cannot be found.
2; He says that he wants to find the correct type name for the call back.
8; He finds the 'turtlesim' wiki page
13; and notes that he sees a 'cmd_vel' topic documented with the type "geometry message :: twist"
6; so he says he should try changing it in turtle.cpp in the call back definition.
8; He says he wants to find some sample code
8; but there is none on this wiki page.
8; He then googles "github turtlesim"
8; and opens up the 'ROS tutorials' repo.
H; I tell him that its fine to look for sample code
H; but that this code exists in a working version online and to please not try and find the real working version.
8; He goes back to looking at the 'looking class methods as callbacks'.
G; He thinks this will be helpful because the code is using a class method as a callback
G; for the velocityCallback.
L; He says that QT will not open correctly.
6; He tries changing the callback to geometry_msgs::Twist
6; referencing the 'turtlesim' wiki page.
11; He tries building it
11; and it succeeds in building.
N; Post questions begin at 40:59.

3.9

11; Coded Storyline and Summary: task#.participant#SummaryCoded StorylineHe says that he will start with running the commands on the sheet.
11; He builds and makes the project
13; and notices that there are some errors in the code.
10; He begins reading through the error messages.
4; He looks into the src directory
4; saying that he's trying to locate the file from which the error has come
4; turtle.cpp
4; and trying to open that file.
7; He opens turtlesim.cpp file
G; because the error says that there is some error in the file.
G; He says he wants to see what's happening inside.
7; He says 'does not name a type'
3; and thinks that there will be some error in the callback.
7; He says he wants to go through the code briefly.

7; He starts reading through the code.
7; He says that he's trying to locate the function in the turtlecpp file
4; (but he has turtlesim.cpp open not turtlecpp) so he says he will probably have to look in the turtleframe.h file.
G; He says that he is trying to locate the velocityCallback
3; and because it isn't in the turtle.cpp file (he doesn't know this
3; he doesn't realize he opened up turtlesim.cpp not turtlecpp) it may be in the turtle_frame.h
7; he also notices that there is some public inheritance from the QApplication
2; and he doesn't know what that is.
11; He then switches to the terminal to navigate to the turtlesim/src file
11; and lists the contents.
N; 7:45.
7; He opens up turtleframe.cpp
7; and scrolls through it.
4; He says he is trying to find the callback from where the call was made
G; but he doesn't see it.
G; He says he's trying to search for the callback
G; and hasn't been able to find it yet.
7; He opens up turtlesim.cpp
7; and the geddit editor tells him that it is already open.
L; He then realizes that instead of opening turtlecpp earlier
L; he opened turtlesim.cpp
L; realizing that he opened the wrong file.
7; He then opens turtle.cpp
7; and scrolls through it and searches to find the velocityCallback.
13; He notes that it is inside the namespace turtlesim.
10; He then looks at the error in the terminal
4; and says that he's trying to see what the four errors in the terminal is.
3; He says that the code for velocityCallback seems to be correct
P; because it looks similar to what he's written before
P; with a constant pointer
P; which he says is good practice because you save memory rather than passing an entire object.
10; He recites the first error
10; saying that its not able to recognize the velocity data type.
P; He says has encountered something similar but not exactly like this.
3; He says that the 'velocity is actually a message'
3; and then asks to himself if there is a message folder in the class.
13; He notices that there isn't a velocity message in the message folder
3; so he thinks we're trying to use the standard velocity command.
7; He reads through the instantation of the subscription for the velocity listener.
10; He looks at the error message again
10; reading it to himself.

3; He says that he thinks the errors below will go away if he fixes the first error
3; and he thinks that the errors are connected.
3; He says that because it is not able to recognize the velocity data type
3; its unable to process things after that.
4; He then looks in the /include directory and opens turtle.h.
7; He says that he opened this because there is an inheritance from turtle.
7; He looks through the turtle.h file
7; and notices that there is a Twist type
3; and thinks that it should not be Twist.
3; He says that he thinks the error is that because he has Twist in the header file
3; but then pauses to say that Twist does have linear as well as angular velocity (as if it should be correct).
6; He then changes Turtle::Velocity to Turtle::Twist in the turtlecpp file (not changing the namespace
6; which is required).
11; He then tries building the project again.
10; It throws the 'does not name a type' error
C; and he says that he isn't sure because the error is still the same.
7; He opens up turtle.h again.
7; He notices that in the h file
7; its defined as Twist (but with a different namespace - doesn't notices this)
2; and he wonders why it isn't working.
3; He says that the particular callback is throwing an error with the data type (Twist)
3; and he wonders why it isn't able to reference it.
7; He opens turtle.cpp
7; and looks for 'anything else [he] is missing'.
7; He then looks in the /include/turtlesim file.
7; He opens turtle.h.
7; He says he's just exploring
C; because hes not able to figure out exactly why its still throwing an error.
7; He's looking between the turtle.cpp files and turtle.h file.
13; He exclaims that the callback is private
2; and he's not sure if that could be a problem.
2; She says that since its inheriting from the main turtlesim class
2; he wants to see if there is a similar callback that is defined in some other way or using another data type.
4; He looks in the turtlesim folder
4; but doesn't find anything helpful.
7; He opens turtlesim.cpp
7; and he says he opened it earlier but it didn't help much.
13; He explains that the turtle.h file is the main header file
13; and that the turtlecpp is where you're defining your functions
13; nodehandlers

13; and variables
3; and he thought that maybe there is some mismatch between the two files
3; but he says he tried that and it didn't work
3; so he's looking for some other thing to fix.
3; He says that he thinks that the change he made is correct
3; but that there needs to be more changes made.
3; He says that the error he has now is essentially the same as the one he got before.
7; He scrolls through the turtle.cpp file
7; muttering about namespaces.
8; He googles "does not name a type error in ros'.
8; He clicks on a rosanswers site
8; "how to solve error geometric_shapes_msgs does not name a type'.
8; He scrolls though the page
8; and then goes back.
8; He clicks on the rosanswers question "error in initlizing costmap - does not name a type'.
8; He then copies over the error message from the terminal and googles it.
8; He then opens qtcenter.com forum thread 'class does not name a type error'
8; and scrolls through it
8; then goes back.
8; He then opens stack overflow 'does not name a type error' question.
8; He then explains the error the person is having
8; and briefly reads the answer.
8; He doesn't think the error the SO asker is having is similar to his
13; but he explains that an answer explains that you must declare a class before using it.
7; He then looks at the turtle.h file and sees that it is 'geometry_msgs::Twist' but then says in the turtle.cpp file its 'Turtle::twist'
6; so he changes the cpp file to match the h file.
11; He saves and tries building the project
12; and it works.
0; Post questions start at around 40:26

3.10

11; Coded Storyline and Summary: task#.participant#SummaryCoded StorylineNavigates to catkin workspace
10; sees build error.
7; Installs sublime textBuilds task again.Goes to the turtlesim/src/ directory.Opens turtle.cpp in sublime.Opens turlesim.cpp in sublime.
10; Goes to look at error message in terminal.
7; Goes back to turtlecpp in sublime.Opens turtlesim.cpp
7; scrolls around.
7; Goes back to turtle.cpp

7; scrolls around.
7; Opens up turtle.h.
7; Looks at turtle.cpp.
7; Looks at turtle.h
7; then back to turtle.cpp.
7; Notices the definition for the callback in the cpp file is different than the h file.
6; Decides to change to Twist
M; but doesn't change the namespace.
7; Looks at the source file again.
11; Builds it
10; doesn't work.
6; Changes the namespace to geometry_msgs
11; builds it and it works.

3.11

11; Coded Storyline and Summary: 3.11SummaryCoded StorylineVelocity TaskNavigates to directory
11; builds the file.
10; Build fails
13; and he notices that there are some errors in the code.
4; He wants to go into the source file to the turtle.cpp file in the turtlesim package.
3; He thinks something may not be included.
4; He tries to open turtle.cpp in vim
N; but vim isn't installed so he installs it.
3; He highlights in terminal and says that "Turtle::Velocity:ConstantPtr" must be defined as a data type in some other place.
7; He then opens turtlecpp in vim.
7; He looks at the veloctyCallback
7; and he says that it looks like we've defined a class called 'Turtle' somewhere.
7; He opens up the header file in a separate window.
13; He says that in the header file
13; we've defined it with a geometryMsgTwist
13; but in the cpp we've used a different data type.
2; He says he doesn't know what velocity::ConstPtr is.
2; He wonders if we've defined a Velocity data type in the header file
12; and sees that he hasn't.
3; He says that he thinks that if he changes it to twist it will work
8; so he looks up 'geometry_msgs::Twist' on google to see if it has linear and angular components.
13; He says that it is a non returning type functions
13; and it will change those values of the object that we've passed

13; and in the call back we're modifying the linear and angular.
3; He says that unless the Velocity const pointer is used somewhere else
3; if he changes it to a Twist
3; and if it has linear.x and angular.z
3; it should be able to compile.
7; He chose to open the header file because he wanted to know how it was defined.
10; He reads through the error message again
7; then looks through the header file.
7; He searches for occurrences of 'typedef' in the header file
2; and he says he wants to see if the Velocity occurs anywhere else.
7; He opens turtlesim.cpp in vim.
7; He searches for Velocity
10; and there are no results.
4; He then says that he's looking for more header files to see if anything called Velocity inside.
7; He opens the turtle_frame.h file.
8; He says that the web search shows him that the Twist type does has linear and angular
8; x and z.
3; So
3; based on this
3; he says that he could fix it using Twist.
3; He now says he's trying to see if changing it could break it down the line.
2; He wants to see if he can find Velocity anywhere.
P; He says that there's a better way to see if there's any occurrence of Velocity
4; and that's to see where any function calls have been made.
11; For that
11; he says he will grep for 'VelocityCallback' in the project
11; and he does so
10; and gets three occurances
10; twice in the turtle.cpp file and once in it's header file
13; he says 'its just subscribing to a topic'.
P; He says that he's used on other people's codebases in the lab
P; and he says they're bad
P; and everything breaks
P; so he says before making changes to a function he wants to see the locations in which it features
P; and for that he uses grep.
12; He sees that in this case its only used to subscribe to a topic called cmdvel somewhere
2; but he wants to see if we're publishing cmdvel somewhere.
2; He says he wants to see if cmd_vel is a 'type of geometry_msg Twist'.
13; He sees that in mimic.cpp we've written a publisher which purblishes a Twist named cmd_vel.
13; He says that this is what the velocity callback is subscrived to.
7; He looks at the velocity callback

13; says it will take cmd_vel and assign lin_vel and ang_vel.
3; He says that right now
3; he thinks we have one node that is publsihing the velocity that the turtle should have
3; and you're subscribing to that topic.
2; He notices and says he doens't know what qreal is
8; so he googles it
8; and he notices that it's a QT thing
8; a GUI data type
8; and that we're using it to render an image of the turtle moving about.
6; He copies the header definition in the cpp file
6; comments out one copy
6; and writes in "geometry_msgs::Twist" as copied from the h file.
11; He makes the file
10; and it builds correctly.
7; He opens the launch file
13; and says we're starting three nodes
13; with two different namespaces
13; and then another node with the global namespace.
7; He walks through the remap tags.
11; He launches the project
11; and then publishes the movement command
10; and it works.Discussion from 19:00 to 22:06.

3.12

11; Coded Storyline and Summary: 3.12SummaryCoded StorylineHe echos the $ROS_MASTER_URI environment variable
10; it prints.
11; Builds the system
10; notices the error.
13; He says there's a couple errors with the cpp file
3; and wonders if there's a header file which wasn't included because its saying 'it doesn't name a type
3; either that or there's a namespacing issue.
10; He says he's looking at the first error.
4; He says he's gonna open turtle.cpp.
7; He opens turtle.cpp in gedit
8; and looks up in the help how to show lines.
7; He wants to go to line 66.
8; He googles gedit goto line.
8; He finds an askubuntu site that tells him to do ctrl+L.
8; He goes to line 66

8; which is the velocityCallback.
P; He says he's used the turtlebot before
P; but he didn't use it to learn ros
P; but he said he had done some tutorials with a physical robot which had used turtlebot.
2; He says he wants to see what is registering the callback
2; because it will tell him the type that it is supposed to take.
13; He says the callback wants a cmd_vel.
11; He looks up in the terminal what version of ros it is
12; and determines that it is kinetic.
P; He says he thought that cmd_vel took a geometry message
3; and says this could be the reason for the error.
P; He knows that with cmd_vel
P; he knows there are standard types like Twist and others.
P; He says he's used these with his own APIs.
P; He says that while cmd_vel is not a type
P; it is user defined
P, 7; but he knows from context and some memory that it usually uses geometry message types.
8; He then Looks up the turtlesim node on the official ROS wiki
12; and finds that cmd_vel uses geometry_msgs/Twist
8; but says 'thats for x and why'
2; so he wants to check what we're actually subscribing to.
2; He also knows that linear.x and angular.y is part of the Twist type.
P; He says that this is pretty standard
P; and that his robots use them.
6; He includes a geometry_msgs/Twist.h file in the top
2; but he says he says he's not sure if this package is included in the cmake lists.
3; He says that we could have defined our own type.
4; He wants to look at the turtle.h file
2; he wants to make sure that the Velocity type hasn't been custom written in.
8; He looks at the turtlesim package summary page
2; saying that he doesn't remember how it works
3; and that he thinks it is a separate package that you 'can just pass in'.
3; He wonders if this is just a fork of the turtlesim package which has been broken
3; because normally you'd just install the turtlesim and then launch the node
3; and then you'd have a turtlemimic just pass the nodes.
2; Becasuse he's seeing things like the Velocity type and QT stuff
2; he wonders if this is a custom node.
3; He in the src folder
3; then determines that the source code was probably copied then modified
3; and that someone broke it.
6; He then changes Velocity to geometry_msgs::Twist
G; saying that if this is supposed to follow the turtlesim api

3; he assumes it is supposed to be taking the Twist message type
P; because this is how you represent velocities
P; and he knows at least that Twist is an actual type
2; whereas he doesn't know if Velocity is.
7; He then decides to look at the CMakeLists file
2; to make sure that the project depends on geometrymessges.
12; From this he determines that it will include the geometrymessges
; because otherwise it won't know where to find them.
2; He then checks if they're linked
12; and he determines that they are.
11; He sources devel/setp.bash
11; and then has it echo an environment variable "CHMAKE_PREFIX_PATH".
P; He did this to make sure that the 'overlays' were correct.
P; He says that if he hadn't sourced it
P; he would have expected it to not recognize the launch file.
11; He then does rostopic list.
11; He then manually types out the rostopic pub
11; the movement command
10; and both turtles move so it works.
N; Post task discussion at 15:48