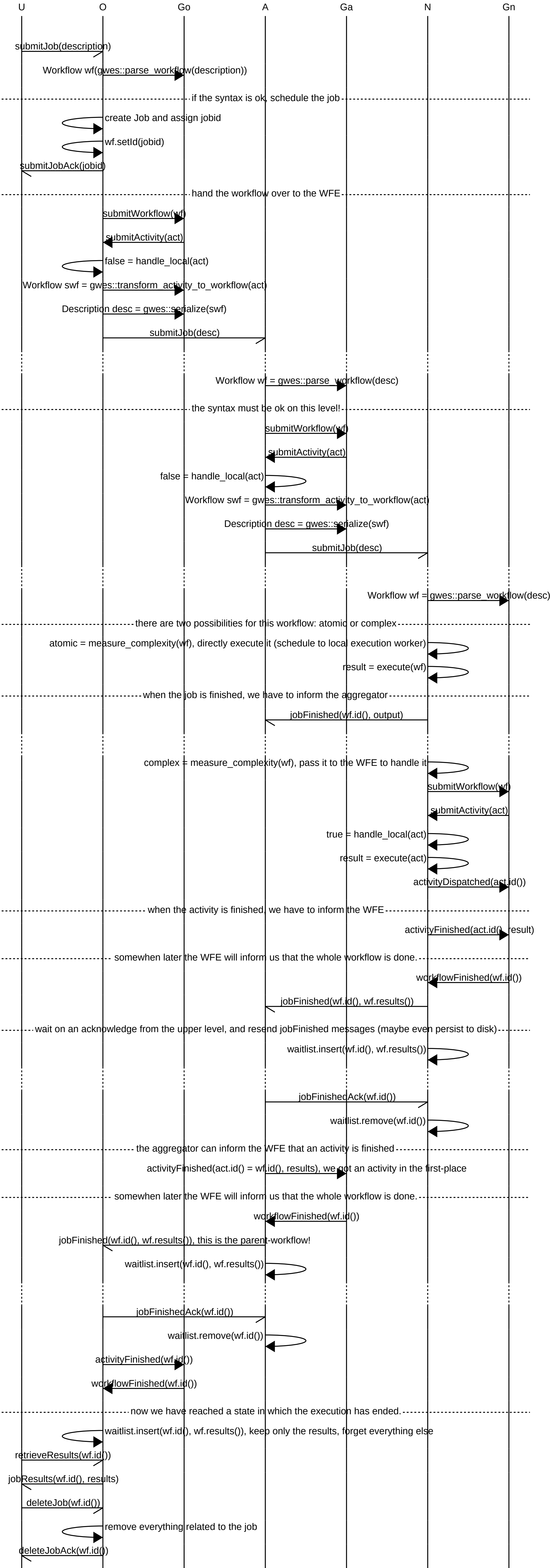
U
O
Go
A
Ga
N
Gn
submitJob(description)
Workflow wf(gwes::parse_workflow(description))
if the syntax is ok, schedule the job
create Job and assign jobid
wf.setId(jobid)
submitJobAck(jobid)
hand the workflow over to the WFE
submitWorkflow(wf)
submitActivity(act)
false = handle_local(act)
Workflow swf = gwes::transform_activity_to_workflow(act)
Description desc = gwes::serialize(swf)
submitJob(desc)
Workflow wf = gwes::parse_workflow(desc)
the syntax must be ok on this level!
submitWorkflow(wf)
submitActivity(act)
false = handle_local(act)
Workflow swf = gwes::transform_activity_to_workflow(act)
Description desc = gwes::serialize(swf)
submitJob(desc)
Workflow wf = gwes::parse_workflow(desc)
there are two possibilities for this workflow: atomic or complex
atomic = measure_complexity(wf), directly execute it (schedule to local execution worker)
result = execute(wf)
when the job is finished, we have to inform the aggregator
jobFinished(wf.id(), output)
complex = measure_complexity(wf), pass it to the WFE to handle it
submitWorkflow(wf)
submitActivity(act)
true = handle_local(act)
result = execute(act)
activityDispatched(act.id())
when the activity is finished, we have to inform the WFE
activityFinished(act.id(), result)
somewhen later the WFE will inform us that the whole workflow is done.
workflowFinished(wf.id())
jobFinished(wf.id(), wf.results())
wait on an acknowledge from the upper level, and resend jobFinished messages (maybe even persist to disk)
waitlist.insert(wf.id(), wf.results())
jobFinishedAck(wf.id())
waitlist.remove(wf.id())
the aggregator can inform the WFE that an activity is finished
activityFinished(act.id() = wf.id(), results), we got an activity in the first-place
somewhen later the WFE will inform us that the whole workflow is done.
workflowFinished(wf.id())
jobFinished(wf.id(), wf.results()), this is the parent-workflow!
waitlist.insert(wf.id(), wf.results())
jobFinishedAck(wf.id())
waitlist.remove(wf.id())
activityFinished(wf.id())
workflowFinished(wf.id())
now we have reached a state in which the execution has ended.
waitlist.insert(wf.id(), wf.results()), keep only the results, forget everything else
retrieveResults(wf.id())
jobResults(wf.id(), results)
deleteJob(wf.id())
remove everything related to the job
deleteJobAck(wf.id())