Taskomat & Taskolib

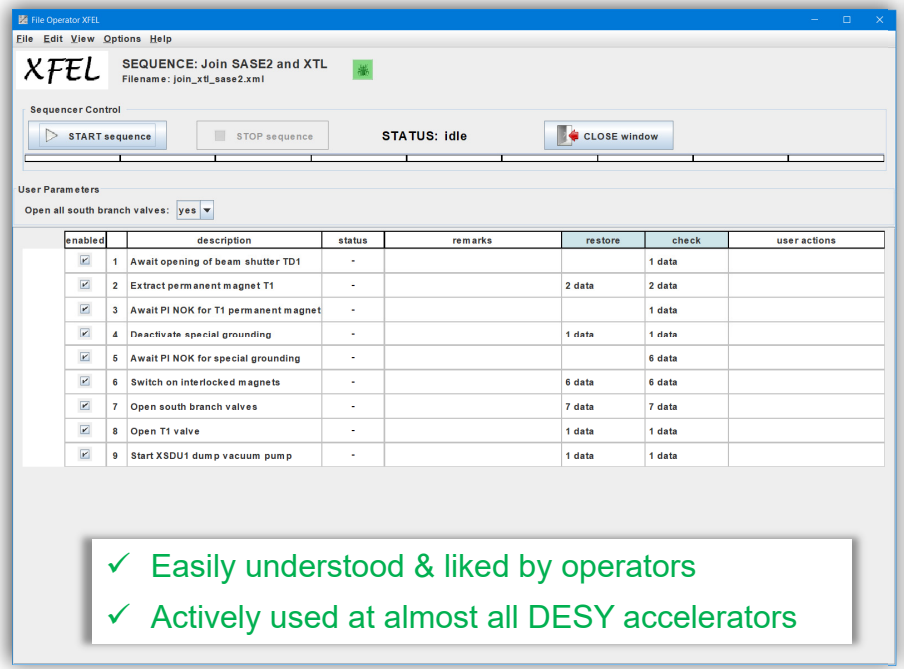
A versatile, programmable sequencer for process automation

- Automation at DESY & in the community
- The DESY “Sequencer / File Operator”
- Design goals for a new sequencer
- TASKOLIB open source library
- TASKOMAT DOOCS server
- Graphical user interface
- Outlook

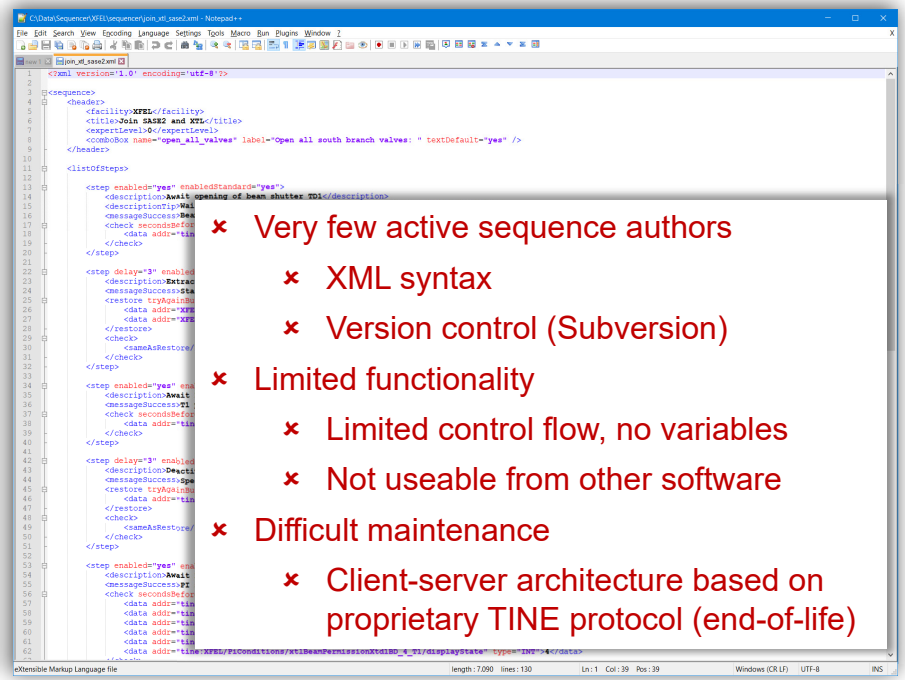
Lars Fröhlich, Olaf Hensler, Ulf Fini Jastrow, Marcus Walla, and Josef Wilgen
PCaPAC 2022, Prague, Czech Republic, 2022-10-07

The DESY “Sequencer/File Operator”

Java application by P. Castro, S. Aytac, J. Maaß



- ✓ Easily understood & liked by operators
- ✓ Actively used at almost all DESY accelerators

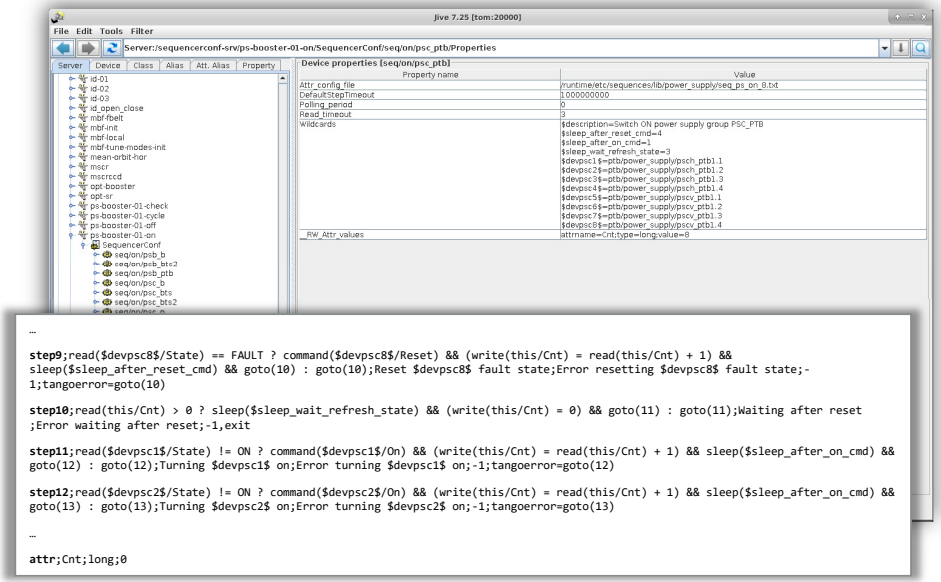


- ✗ Very few active sequence authors
  - ✗ XML syntax
  - ✗ Version control (Subversion)
- ✗ Limited functionality
  - ✗ Limited control flow, no variables
  - ✗ Not useable from other software
- ✗ Difficult maintenance
  - ✗ Client-server architecture based on proprietary TINE protocol (end-of-life)

ICALEPCS 2021: Automation is a Topic

Two contributions on recent sequencer developments

Elettra – Sincrotrone Trieste




```
…

step9;read($devpsc8$/State) == FAULT ? command($devpsc8$/Reset) && (write(this/Cnt) = read(this/Cnt) + 1) &&
sleep($sleep_after_reset_cmd) && goto(10) : goto(10);Reset $devpsc8$ fault state;Error resetting $devpsc8$ fault state;-
1;tangoerror=goto(10)

step10;read(this/Cnt) > 0 ? sleep($sleep_wait_refresh_state) && (write(this/Cnt) = 0) && goto(11) : goto(11);Waiting after reset
;Error waiting after reset;-1,exit

step11;read($devpsc1$/State) != ON ? command($devpsc1$/On) && (write(this/Cnt) = read(this/Cnt) + 1) && sleep($sleep_after_on_cmd) &&
goto(12) : goto(12);Turning $devpsc1$ on;Error turning $devpsc1$ on;-1;tangoerror=goto(12)

step12;read($devpsc2$/State) != ON ? command($devpsc2$/On) && (write(this/Cnt) = read(this/Cnt) + 1) && sleep($sleep_after_on_cmd) &&
goto(13) : goto(13);Turning $devpsc2$ on;Error turning $devpsc2$ on;-1;tangoerror=goto(13)

…

attr;Cnt;long;0
```

ITER

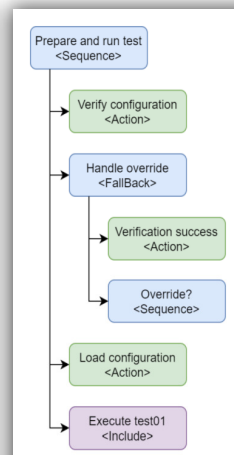


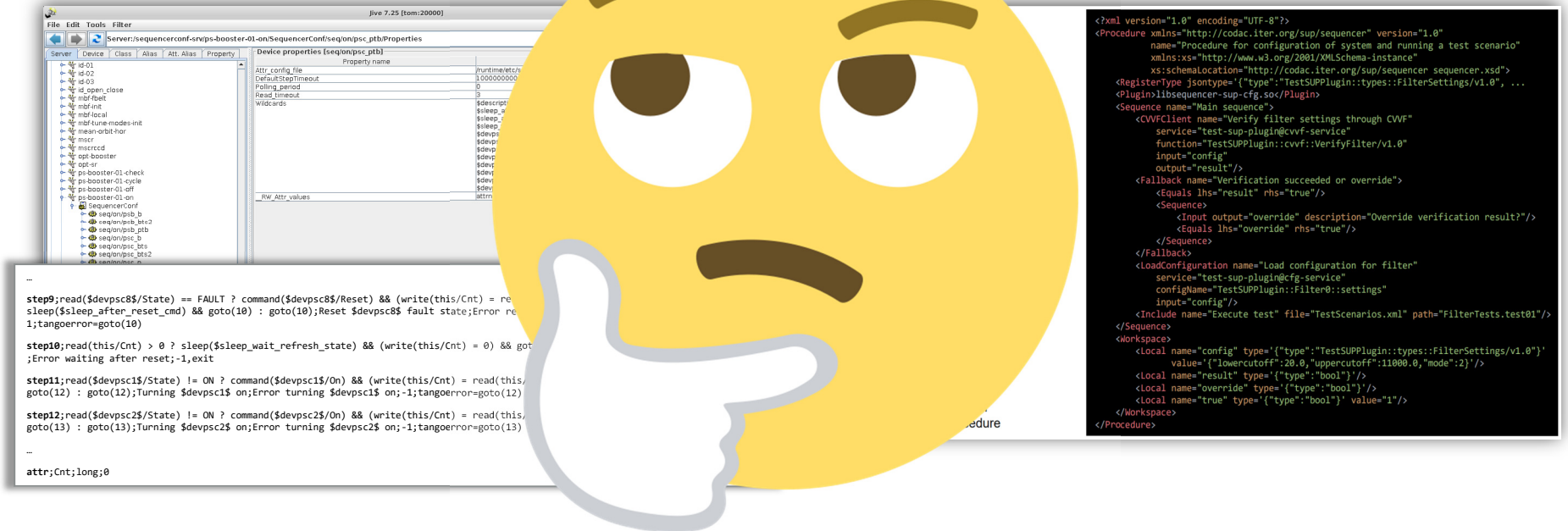
Figure 2: Behavior tree and xml representation of a test procedure

```
<?xml version="1.0" encoding="UTF-8"?>
<Procedure xmlns="http://codac.iter.org/sup/sequencer" version="1.0"
           name="Procedure for configuration of system and running a test scenario"
           xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
           xs:schemaLocation="http://codac.iter.org/sup/sequencer sequencer.xsd">
    <RegisterType jsontype='{"type":"TestSUPPlugin::types::FilterSettings/v1.0", ...
    <Plugin>libsequencer-sup-cfg.so</Plugin>
    <Sequence name="Main sequence">
        <CVVFClient name="Verify filter settings through CVVF"
            service="test-sup-plugin@cvvf-service"
            function="TestSUPPlugin::cvvf::VerifyFilter/v1.0"
            input="config"
            output="result"/>
        <Fallback name="Verification succeeded or override">
            <Equals lhs="result" rhs="true"/>
            <Sequence>
                <Input output="override" description="Override verification result?"/>
                <Equals lhs="override" rhs="true"/>
            </Sequence>
        </Fallback>
        <LoadConfiguration name="Load configuration for filter"
            service="test-sup-plugin@cfg-service"
            configName="TestSUPPlugin::Filter0::settings"
            input="config"/>
        <Include name="Execute test" file="TestScenarios.xml" path="FilterTests.test01"/>
    </Sequence>
    <Workspace>
        <Local name="config" type='{"type":"TestSUPPlugin::types::FilterSettings/v1.0"}'
               value='{"lowercutoff":20.0,"uppercutoff":11000.0,"mode":2}'/>
        <Local name="result" type='{"type":"bool"}'/>
        <Local name="override" type='{"type":"bool"}'/>
        <Local name="true" type='{"type":"bool"}' value="1"/>
    </Workspace>
</Procedure>
```

ICALEPCS 2021: Automation is a Topic

Two contributions on recent sequencer developments

Elettra – Sincrotrone Trieste



ITER

```
<?xml version="1.0" encoding="UTF-8"?>
<Procedure xmlns="http://codac.iter.org/sup/sequencer" version="1.0"
           name="Procedure for configuration of system and running a test scenario"
           xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
           xs:schemaLocation="http://codac.iter.org/sup/sequencer sequencer.xsd">
    <RegisterType jsontype='{"type":"TestSUPPlugin::types::FilterSettings/v1.0", ...
    <Plugin>libsequencer-sup-cfg.so</Plugin>
    <Sequence name="Main sequence">
        <CVVFClient name="Verify filter settings through CVVF"
            service="test-sup-plugin@cvvf-service"
            function="TestSUPPlugin::cvvf::VerifyFilter/v1.0"
            input="config"
            output="result"/>
        <Fallback name="Verification succeeded or override">
            <Equals lhs="result" rhs="true"/>
            <Sequence>
                <Input output="override" description="Override verification result?"/>
                <Equals lhs="override" rhs="true"/>
            </Sequence>
        </Fallback>
        <LoadConfiguration name="Load configuration for filter"
            service="test-sup-plugin@cfg-service"
            configName="TestSUPPlugin::Filter0::settings"
            input="config"/>
        <Include name="Execute test" file="TestScenarios.xml" path="FilterTests.test01"/>
    </Sequence>
    <Workspace>
        <Local name="config" type='{"type":"TestSUPPlugin::types::FilterSettings/v1.0"}'
               value='{"lowercutoff":20.0,"uppercutoff":11000.0,"mode":2}'/>
        <Local name="result" type='{"type":"bool"}'/>
        <Local name="override" type='{"type":"bool"}'/>
        <Local name="true" type='{"type":"bool"}' value="1"/>
    </Workspace>
</Procedure>
```

Design Goals

Requirements for a new sequencer

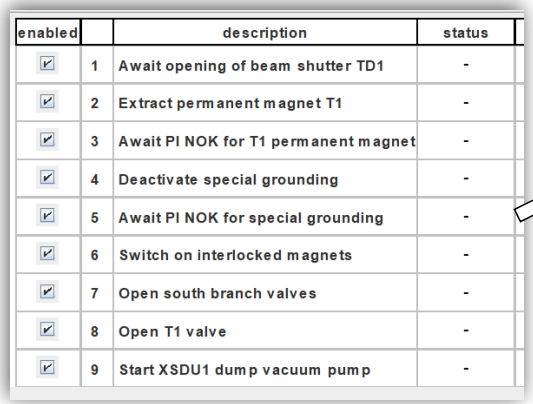
Easy editing & maintenance

- Operators, technical & domain experts

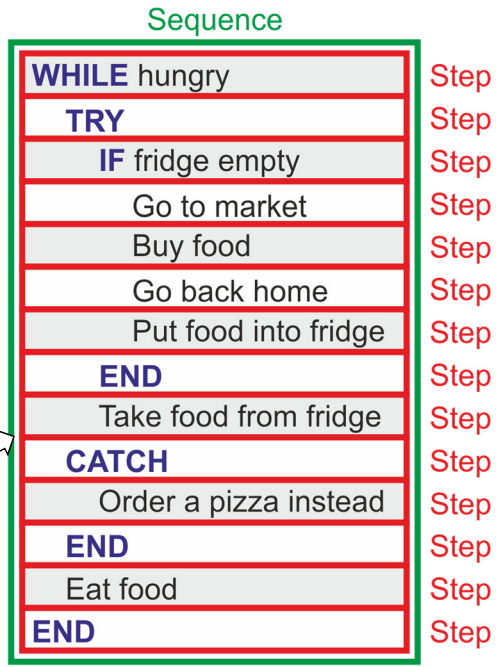
More programmability & functionality

- Implement loops, exception handling, complex conditions

Add control flow to sequence of steps (at abstract level)



enabled		description	status
☑	1	Await opening of beam shutter TD1	-
☑	2	Extract permanent magnet T1	-
☑	3	Await PI NOK for T1 permanent magnet	-
☑	4	Deactivate special grounding	-
☑	5	Await PI NOK for special grounding	-
☑	6	Switch on interlocked magnets	-
☑	7	Open south branch valves	-
☑	8	Open T1 valve	-
☑	9	Start XSDU1 dump vacuum pump	-



Design Goals

Requirements for a new sequencer

Easy editing & maintenance

- Operators, technical & domain experts

More programmability & functionality

- Implement loops, exception handling, complex conditions

Use a scripting language to define actions and conditions within the steps

```
<step enabled="yes" enabledStandard="yes">
    <description>Switch on interlocked magnets</description>
    <messageSuccess>Interlocked magnets switched on</messageSuccess>
    <restore tryAgainButton="yes">
        <data addr="XFEL.MAGNETS/MAGNET.ML/BD.2077.T1/PS_ON" type="INT" write="1">1</data>
        <data addr="XFEL.MAGNETS/MAGNET.ML/QF.2083.T1/PS_ON" type="INT" write="1">1</data>
        <data addr="XFEL.MAGNETS/MAGNET.ML/BD.2084.T1/PS_ON" type="INT" write="1">1</data>
        <data addr="XFEL.MAGNETS/MAGNET.ML/BD.2097.T1/PS_ON" type="INT" write="1">1</data>
        <data addr="XFEL.MAGNETS/MAGNET.ML/QF.2098.T1/PS_ON" type="INT" write="1">1</data>
        <data addr="XFEL.MAGNETS/MAGNET.ML/QF.2110.T1/PS_ON" type="INT" write="1">1</data>
    </restore>
    <check>
        <sameAsRestore/>
    </check>
</step>
```

```
dset("XFEL.MAGNETS/MAGNET.ML/BD.2077.T1/PS_ON", 1)
dset("XFEL.MAGNETS/MAGNET.ML/QF.2083.T1/PS_ON", 1)
dset("XFEL.MAGNETS/MAGNET.ML/BD.2084.T1/PS_ON", 1)
dset("XFEL.MAGNETS/MAGNET.ML/BD.2097.T1/PS_ON", 1)
dset("XFEL.MAGNETS/MAGNET.ML/QF.2098.T1/PS_ON", 1)
dset("XFEL.MAGNETS/MAGNET.ML/QF.2110.T1/PS_ON", 1)
```

```
for magnet in iterate_over_magnet_addresses() do
  dset(magnet, 1)
end
```

Design Goals

Requirements for a new sequencer

Easy editing & maintenance

- Operators, technical & domain experts

More programmability & functionality

- Implement loops, exception handling, complex conditions

Integration into the control system

- Sequences can be started/stopped/configured via control system *set* call

Scalability

- Maintain a library with hundreds of sequences
- Run dozens of sequences concurrently

Separation of main logic, control system interface, and GUI

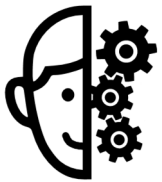
- Sequencer library (control system independent)
- Sequencer server/device (embedded in the control system)
- Graphical user interface for the server/device

TASKOLIB, TASKOMAT, GUI

First implementations

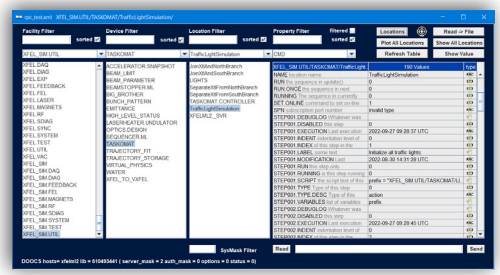
TASKOLIB

- Control system independent library
- Handles creation, modification, execution, storage of sequences and steps
- Platform-independent C++ 17
- Steps contain sandboxed LUA scripts (lightweight, easy to learn)



TASKOMAT

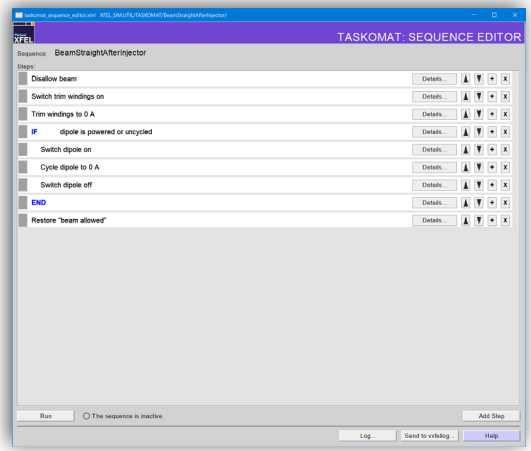
- Control system specific server/device for creating, modifying, and executing sequences
- First implementation: DOOCS server



Injects control system specific commands (dget, dset, …)

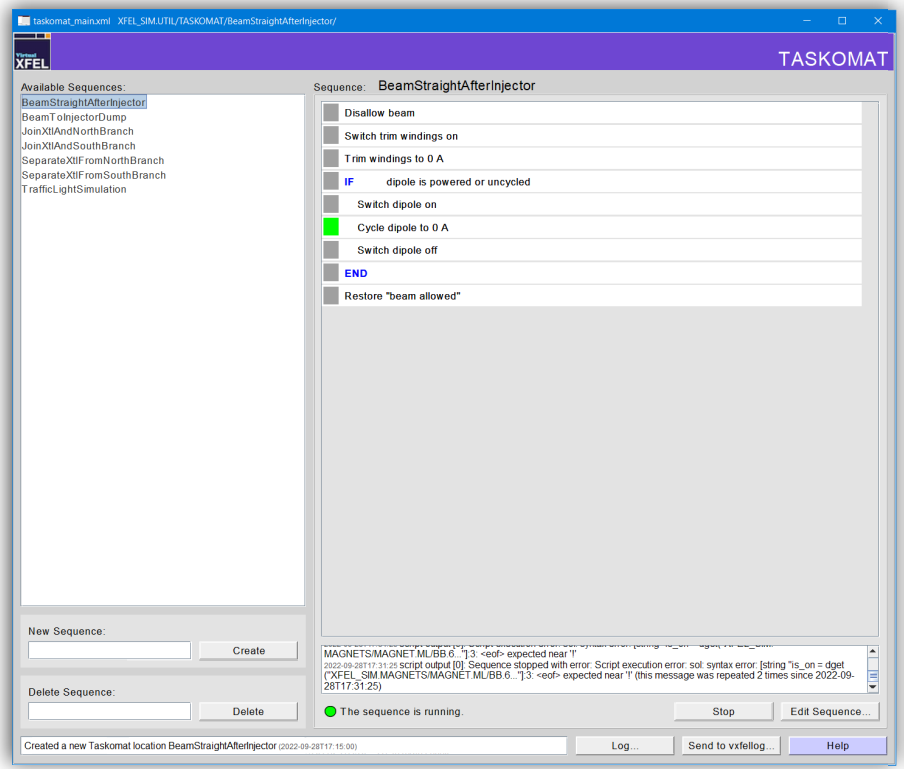
Graphical User Interface

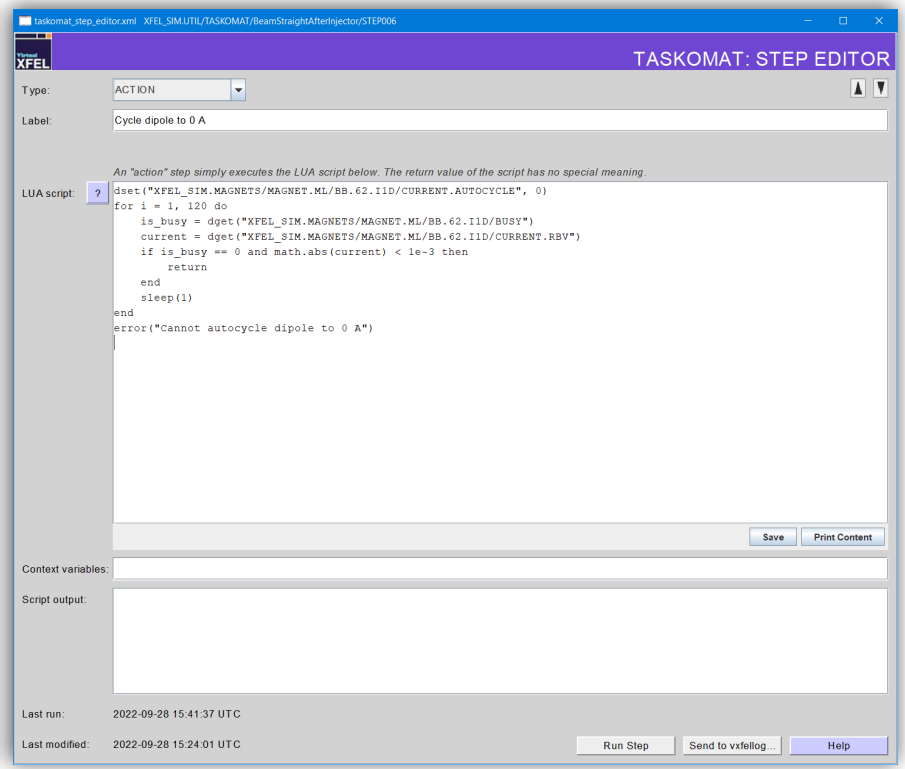
- User interface for creating, modifying, and executing sequences
- First implementation: **jddd** panels for DOOCS server



Graphical User Interface

Rapidly built with jddd UI builder for DOOCS: Rudimentary, but usable





Summary & Outlook

TASKOLIB Library

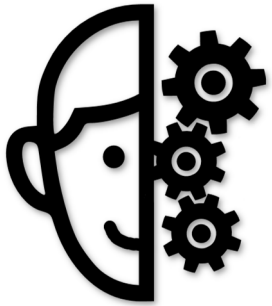
- Control system independent C++ library for creating, modifying, and executing sequences built out of Lua steps
- Open Source (LGPL-2.1): https://github.com/taskolib/taskolib

TASKOMAT Server for DOOCS

- First implementation of a TASKOLIB-based sequencer
- Headless DOOCS server application
- jddd “thin client” user interface panels

Status

- “Minimum viable product” stage: First instances deployed with friendly users
- Gathering feedback and experience for next extensions



Outlook

- More features based on user feedback (e.g. automatic version control)
- Develop full-fledged GUI?
  - Syntax highlighting, improved Lua editor
- Collaboration: Other control systems?