Static code Analysing Workflow on the Basis of CppCheck

Mariam Pirtskhalava

Georgian Technical University

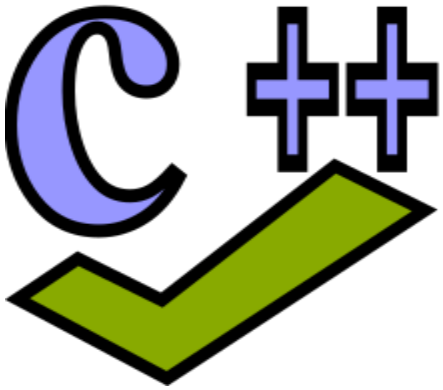
ATHENA

ATHENA is C++ control framework, in which data processing and analysis is performed.
In the ATLAS development workflow, this repository is also the merge target for developments from individual users or groups.

The Athena GitLab repository contains all code that could be built into an ATLAS software release.

Coverity

Coverity is a static code analysis tool and dynamic code analysis service
The tools enable engineers and security teams to find defects in custom source code written in C, C++, Java, C#, JavaScript and more.



CPPCHECK

Cppcheck is a static analysis tool for C/C++ code, that provides code analysis to detect bugs, undefined behaviour and dangerous coding constructs.

Currently we are working on Cppcheck.

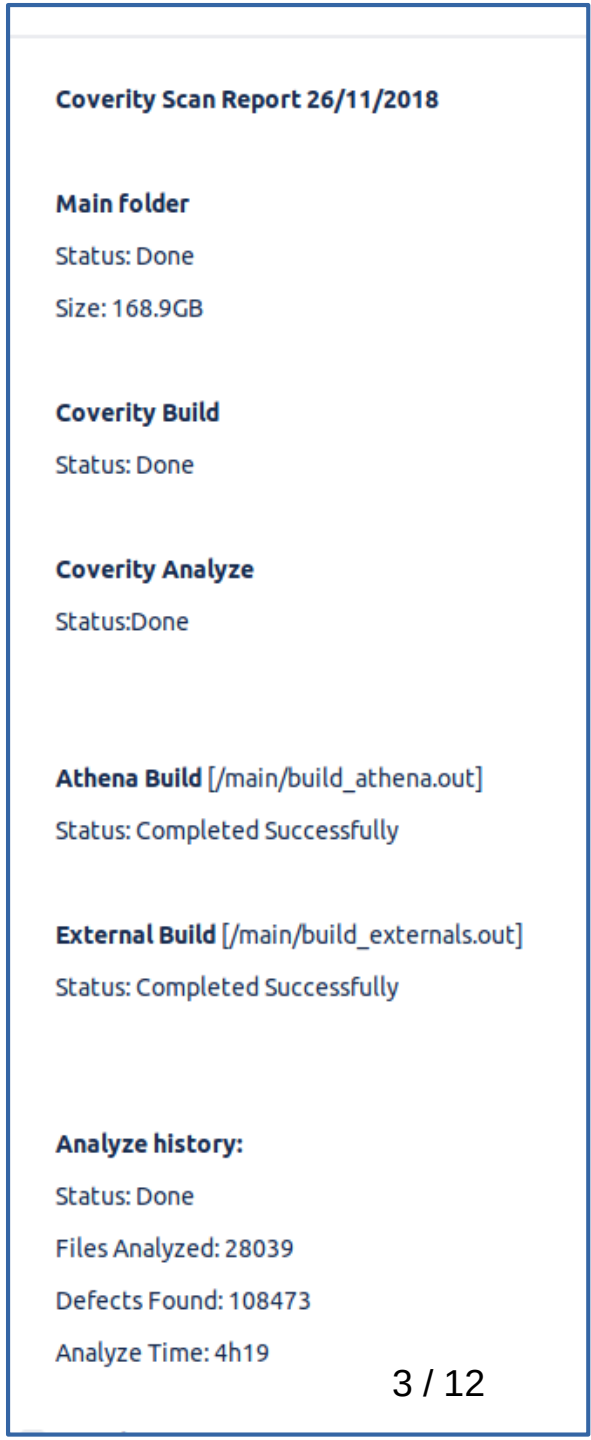
COVERITY workflow

Since June 2018 Georgian team was working with Andew Washbrook and Emil Obreshkov on Coverity and was providing Jira Tickets every week, but usage of Coverity in ATLAS temporarily stopped, so in January, 2019, activity has been postponed.

- We could provide scan process of ATHENA repository, but we were just making reports.
- Our duties were to analyze status of ATHENA scan and make reports, also we were checking defect list, providing reports and JIRA Tickets for defects.

After creating JIRA Tickets for scan, we were manually sending them to authors and were getting feedbacks.

Successful COVERITY scans were taking up to 30 hours.



- Example of Jira Ticket for Successful Scan

Coverity Scan Report	24/09/2018			
	Overal:			
	Name	Status	Size	Items
	Main folder	Complete	210,5 GB	417'81
	Coverity Build	Done		
	Coverity Analyze	Done		

Build History:					
Name	Instalation	Faults	File		
Status	Strings	Status	Strings		
Athena Build			Completed Successfully		/main/build_athena.out
External Build			Completed Successfully		/main/build_externals.out

Analyze history:	
Status	Done
Files Analyzed	35788
Defects Found	60084
Analyze Time	13h00

https://its.cern.ch/jira/projects/ATLASSQ/issues/ATLASSQ-28?filter=allopenissues&orderby=cf[13000]+ASC,+priority+DESC,+updated+DESC

- Example of Jira Ticket for Unsuccessful Scan

Name	Status	Size	Items
Main folder	Not Complete	7.0Gb	163’932
Coverity Build	Not Done		
Coverity Analyze	Not Done		

Coverity Scan Report 21/06/2018

Build History:					
Name	Instalation	Faults	File		
Status	Strings	Status	Strings		
Athena Build			Configuring incomplete, errors occurred!	#35	/main/build_athena.out
External Build			CMake Error at Coin3D-stamp/download-Coin3D.cmake:159 (message): make[2]: *** [src/Coin3D-stamp/Coin3D-download] Error 1 make[2]: Target `External/Coin3D/CMakeFiles/Coin3D.dir/build' not remade because of errors. make[1]: *** [External/Coin3D/CMakeFiles/Coin3D.dir/all] Error 2	#7'607	/main/build_externals.out
		[94%] Built target Package_Gdb make[1]: Target `all' not remade because of errors. make: *** [all] Error 2 make: Target `default_target' not remade because of errors.	#247'771	/main/build_externals.out	

Analyze history:	
Status	Not Done
Files Analyzed	0
Defects Found	0
Analyze Time	-

https://its.cern.ch/jira/projects/ATLASSQ/issues/ATLASSQ-3?filter=allopenissues&orderby=cf[13000]+ASC,+priority+DESC,+updated+DESC

ATLAS Software Quality / ATLASSQ-99

Coverity Defects Report MuonSpectrometer package

Issue Type: Bug

Assignee: Andrew John Washbrook

Created: 30/Nov/18 1:32 PM

Priority: Minor

Reporter: Mariam Pirtskhalava

Coverity Defects Report
Package: Muon Spectometer

1. CID: 121369

Defect Type: Resource leak
First Detected: 27/11/2018
File: /MuonSpectrometer/MuonGeoModel/src/MuonChamber.cxx
Author: Jochen Meyer <Jochen.Meyer@cern.ch>
Last Modified:13/11/2018

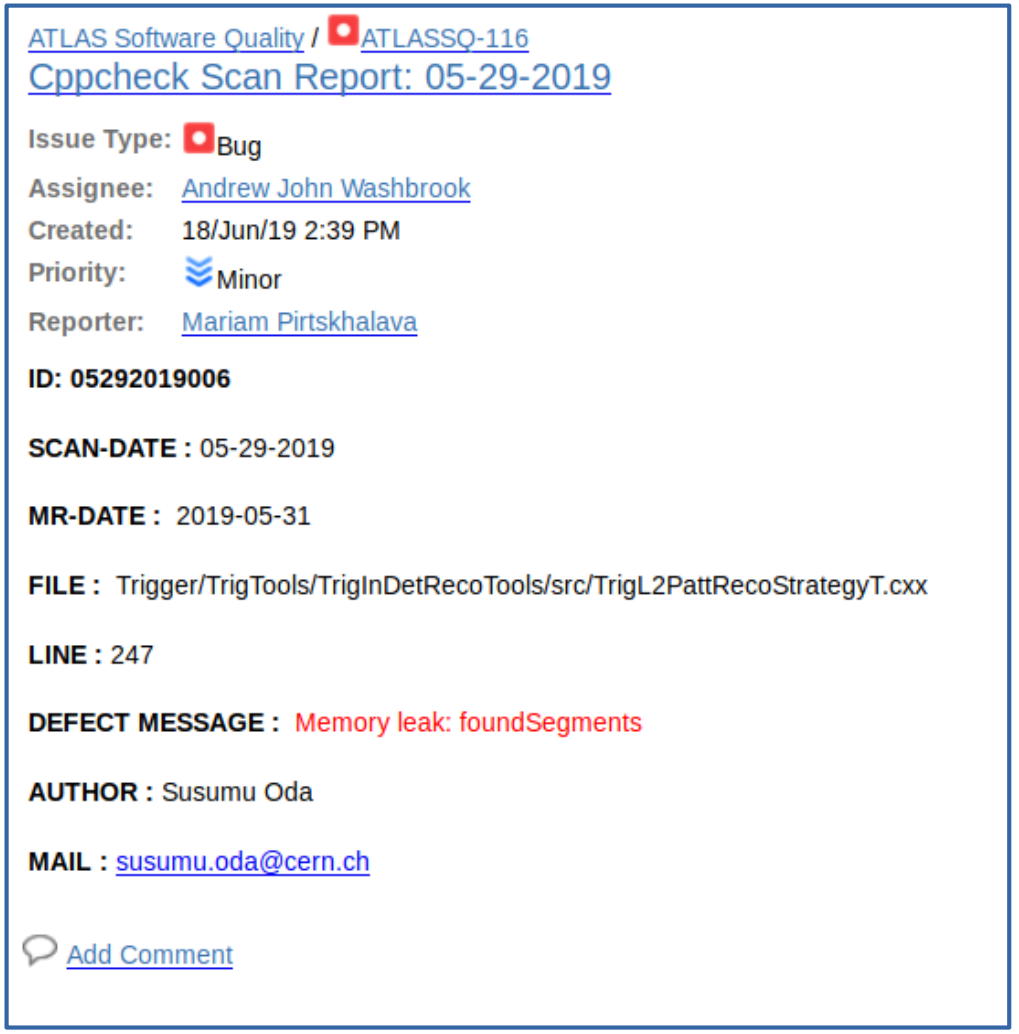
1. CID: 121366

Defect Type: Resource leak
First Detected: 27/11/2018

Cppcheck workflow steps

We started scanning Simulation folder with cppcheck. Simulation folder scans were successful, so we moved to whole ATHENA repository scanning, which means:

1. Cloning ATHENA repository from GitLab using git clone command and getting Athena folder locally
2. Scanning ATHENA repository with Cppcheck and getting defect list in .xml file format
3. Find authors of defects on GitLab with git log command in command line
4. Reading and parsing .xml manually in Excel
5. Grouping latest defects with merge-request date
6. Creating JIRA TICKET
7. Generating defects on cppcheck-list.web.cern.ch



fullathenascan.xml
~/Desktop/cppcheck

Save

```
<?xml version="1.0" encoding="UTF-8"?>
<results version="2">
    <cppcheck version="1.82"/>
    <errors>
        <error id="memleak" severity="error" msg="Memory leak: attrSpec" verbose="Memory leak: attrSpec" cwe="401">
            <location file="athena/AtlasTest/DatabaseTest/IOVDbTestAlg/src/IOVDbTestAlg.cxx" line="389"/>
        </error>
        <error id="syntaxError" severity="error" msg="syntax error" verbose="syntax error">
            <location file="athena/AtlasTest/GoogleTestTools/test/gt_GoogleTestTools.cxx" line="26"/>
        </error>
        <error id="shiftTooManyBitsSigned" severity="error" msg="Shifting signed 32-bit value by 31 bits is undefined behaviour" verbose="Shifting signed 32-bit value by 31 bits is
undefined behaviour" cwe="758">
            <location file="athena/Calorimeter/CaloClusterCorrection/src/CaloClusterTimeTool.cxx" line="232" info="Shift"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: tool.order" verbose="Uninitialized struct member: tool.order" cwe="908">
            <location file="athena/Calorimeter/CaloClusterCorrection/src/CaloRunClusterCorrections.cxx" line="314"/>
        </error>

        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_nx" verbose="Uninitialized struct member: onebin.m_nx" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="296"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_ny" verbose="Uninitialized struct member: onebin.m_ny" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="296"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_side" verbose="Uninitialized struct member: onebin.m_side" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="296"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_OSRatio" verbose="Uninitialized struct member: onebin.m_OSRatio" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="296"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_nx" verbose="Uninitialized struct member: onebin.m_nx" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="352"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_ny" verbose="Uninitialized struct member: onebin.m_ny" cwe="908">
            <location file="athena/Calorimeter/CaloMonitoring/rootMacros/CellClusterLinkTool.C" line="352"/>
        </error>
        <error id="uninitStructMember" severity="error" msg="Uninitialized struct member: onebin.m_side" verbose="Uninitialized struct member: onebin.m_side" cwe="908">
```

Defect list in .xml file format

Fast Navigation

```
888    double dP1 = (P1-M[1])+PV1*d*r0;
889
890    if(fabs(dP1) > m_halflenght) {
891
892      double r1;
893      dP1 > m_halflenght ? r1 = m_halflenght-P1 : r1 = -(m_halflenght+P1);
894
895      double v1 = PV1;
896      double v2 = PV2;
897      d = v0*v2-v1*v1  ; if(d<=0.) continue;
898      x = (r0*(r0*v2-r1*v1)+r1*(r1*v0-r0*v1))/d;
899      if(x>Xc) continue;
900    }
901
```

defect line

STATUS	ID	SCAN-DATE	MR DATE	FILE	LINE	DEFECT MESSAGE	AUTHOR	MAIL
	05292019001	06-09-2019	05-31-2019	InnerDetector/InDetRecTools /SiCombinatorialTrackFinderTool_xk /src/SiTrajectoryElement_xk.cxx	898	Uninitialized variable: r1	Susumu Oda	susumu.oda@cern.c

JIRA Ticket

ATLAS Software Quality / ATLASSQ-111

Cppcheck Scan Report: 05-29-2019

Details

Type: Bug — Status: TO DO

Priority: Minor — Resolution: Unresolved

Labels: None

Description

ID: 05292019001

SCAN-DATE : 05-29-2019

MR-DATE : 2019-05-31

FILE : InnerDetector/InDetRecTools/SiCombinatorialTrackFinderTool_xk/src/SiTrajectoryElement_xk.cxx

LINE : 898

DEFECT MESSAGE : Uninitialized variable: r1

AUTHOR : Susumu Oda

MAIL : susumu.oda@cern.ch

Gitlab

SiTrajectoryElement_xk.cxx 60.5 KB

```
1  /*
2    Copyright (C) 2002-2017 CERN for the benefit of the ATLAS collaboration
3  */
4
5
6  #include "TrkSurfaces/PerigeeSurface.h"
7  #include "TrkSurfaces/AnnulusBounds.h"
8  #include "TrkMaterialOnTrack/ScatteringAngles.h"
9  #include "TrkMaterialOnTrack/MaterialEffectsOnTrack.h"
10 #include "TrkEventPrimitives/FitQualityOnSurface.h"
11 #include "TrkRIO_OnTrack/RIO_OnTrack.h"
12 #include "SiCombinatorialTrackFinderTool_xk/SiTrajectoryElement_xk.h"
13
14 #include "InDetRIO_OnTrack/PixelClusterOnTrack.h"
15 #include "InDetRIO_OnTrack/SCT_ClusterOnTrack.h"
16 #include <stdexcept>
17 #include <math.h>//for sincos function
18
19 ////////////////////////////////////////////////////////////////////
20 // Set trajectory element
21 ////////////////////////////////////////////////////////////////////
22
23 void InDet::SiTrajectoryElement_xk::set
24 (int                                    st,
25  const InDet::SiDetElementBoundaryLink_xk*&  dl,
26  const InDet::SiClusterCollection::const_iterator& sb,
27  const InDet::SiClusterCollection::const_iterator& se,
28  const InDet::SiCluster*                 si
29  )
30 {
31   m_fieldMode    = false;
32   if(m_tools->fieldTool().magneticFieldMode()!=0) m_fieldMode = true;
33   m_status       = 0                      ;
34   m_detstatus    = st                     ;
35   m_ndist        = 0                      ;
```

cppcheck-list.web.cern.ch

Problems

JIRA Ticket Group administration

We had JIRA Ticket group called “ATLASSQ” with Andrew Washbrook, while we were working on COVERITY, but since Andrew left CERN, we don’t have permissions to add authors or become administrators of the group.

Scan Frequency

We don’t know how frequently we should scan ATHENA repository and create JIRA Tickets and reports.

Merge Request

For now, we don’t know if we should attach Cppcheck scanning to Merge Requests. We did scan testing and scanned merge request files separately and maximum time for scan was seconds.

Automatization

Whole process described above is mostly manual, so one of our main goal is to make automatization of whole process.

If automatization will be successful, we can propose it for Continuous Integration.

Future Steps

Our future steps are:

Get information about frequency of scan (Weekly, Monthly, etc.)

Get information about JIRA Ticket group administration and member selection.

Write scripts for automatization.

Scripts needed for automatization:

1. Cloning ATHENA repository
2. Scan ATHENA with Cppcheck and get defects in .xml
3. Reading, inserting tags into xml and writing information in these tags
4. Automatization of JIRA Ticket creation
5. Reading xml and transfer into web-page

some of these scripts are already written

```
for ((i=0; i<$amount; i++))
do



ccpath=${arr_path[$i]}
ccid=${arr_cid[$i]}
cdt=${arr_fdd[$i]}
cpath=${ccpath:1:${#ccpath}}
cpackage=$(echo $cpath | cut -d'/' -f 1)
ctype=${arr_type[$i]}

owner_name=$(git log -1 --pretty=format:'%an' -- $cpath )
owner_mail=$(git log -1 --pretty=format:'%ae' -- $cpath )
mod_date=$(git log -1 --pretty=format:'%cd' --date=short -- $cpath )

file=../out/$cpackage
touch $file || exit

num=$(grep -c "N:" $file)
((num++))
echo $file
echo N: $num >> $file
echo CID: $ccid >> $file
echo type: $ctype >> $file
echo first detected: $cdt >> $file
echo path: $cpath >> $file
echo owner: $owner_name >> $file
echo owner mail: $owner_mail >> $file
echo last modified: $mod_date >> $file
#echo package: $cpackage >> $file
echo newline >> $file

sed -i '' 's/newline/\
/g' $file
echo -ne "loading: $i/$amount\r"
sleep 1
done

cd ..


#open $file
```

Conclusion

1. We have done identical workflows for both tools: Coverity and Cppcheck, while workflow for COVERITY has been approved already one year.
2. Based on these experiences, soon we will be able to automate whole scanning and delivering process.
3. After automatization, we can propose it for Continuous Integration on GitLab.

Thanks for Attention!

mariam.pirtskhalava@cern.ch